Preface

This book is based on an interdisciplinary course on logic offered to upper-level undergraduates at Duke University over a period of more than ten years. Why an interdisciplinary course on logic? Although logic has been a discipline of study in philosophy since ancient times, in recent decades it has played an important role in other disciplines as well. For example, logic is at the core of two programming languages, is used in program verification, has enriched philosophy (and computer science) with non-classical logics that can deal constructively with contradictions, and has shaken the foundations of mathematics with insight into non-computable functions and non-provability. Several of these ideas are treated in this book.

We developed a one-semester course suitable for undergraduates that presents some of these more recent, exciting ideas in logic as well as some of the traditional, core ideas. Undergraduate students generally have limited time to pursue logic courses and we found that the course we offered gave them some understanding of both the breadth of logic and the depth of ideas in logic.

This book addresses select topics drawn from three different areas of logic: proof theory, computability theory, and philosophical logic. A common thread throughout is the application of logic to computers and computation.

Part 1 on Proof Theory introduces a deductive system (resolution logic) that comes from an area of research known as automated deduction.

Part 2 on Computability Theory explores the limits of computation using an abstract model of computers called register machines.

Part 3 on Philosophical Logic presents a certain non-classical logic (relevance logic) and a semantics for it that is useful for automated reasoning systems that must deal with the possibility of inconsistent information.

The book can serve a variety of needs. For the first time there is now available a text for an instructor who would like to offer a course that teaches the role of logic in several disciplines. The book could also be used as a supplementary text for a logic course that emphasizes the more traditional topics of logic but also wishes to include a few special topics. The book is also designed to be a valuable resource for researchers and academics who want an accessible yet substantial introduction to the three topics.

The three areas from which the special topics are drawn — proof theory, computability theory, and philosophical logic — exhibit the different roles that logic plays in three different disciplines: computer science, mathematics, and philosophy. The three parts of the book were written by a computer scientist, a mathematician, and a philosopher, respectively, and each part was reviewed by the other two authors for accessibility to students in their fields. The three parts of the book are roughly of equal length. The second part, on computability theory, is largely independent of the first, but the third part, on philosophical logic, is best presented after the first two parts.

Although it is helpful to have had a previous course in logic, we present the topics in such a way that this is not necessary. However, some mathematical background is useful, especially if no logic background is offered. We had a number of freshmen and sophomores take this course with success, but they had a strong analytic preparation. In particular, prior exposure to proofs by induction is important. (We do offer a summary review of the induction methods employed in an appendix.)

The three topics covered are both timely and important. Although the use of automated theorem proving in artificial intelligence (AI) is often associated with the early decades of AI, it is also of great value in some current AI research programs. Watson, IBM's question-answering computer, made famous in 2011 by an impressive performance on the quiz show *Jeopardy!*, employed resolution logic (presented in **Part 1**) through the resolution-based programming language Prolog.

Part 2, on computability theory, presents one of the great success stories of mathematical logic. We now have a methodology for proving that certain problems cannot be solved by an algorithm. The ideas required to reach this goal can be traced back to Gödel and Turing and moreover played an important role in the development of the modern-day computer; it is for this reason that Gödel's and Turing's names are on *Time* magazine's list of the twenty most influential scientists and thinkers of the twentieth century.

Classical logic was motivated by considerations in mathematics. The important role that logic plays in other disciplines has given rise to logics that extend or differ from classical logic; examples include modal logic, intuitionistic logic, fuzzy logic, and relevance logic. **Part 3** explores the ideas of extensions and alternatives to classical logic, with an in-depth treatment of one of these, relevance logic.

The book begins with proof theory for both propositional logic and first-order logic. In each case, there is a quick review of the semantics of that logic. This has the advantage of serving as background for the subsequent parts on computability theory and non-classical logics.

A computer-oriented deductive logic based on the resolution inference rule is employed. At the propositional level all proofs are given, including both soundness and completeness proofs. In resolution logic the completeness theorem proof has an intuitive graphical form that makes the proof easier to comprehend. At the first-order level proofs are deferred to a set of problems to be undertaken by the mathematically oriented students. They cover most of the major results, including the steps to the completeness theorem. Plausibility arguments are used instead. This pedagogical strategy works well without losing the important content because first-order proof theory based on resolution employs lifting proofs almost verbatim from the propositional counterpart proof. The lifting process is discussed in detail. There is an extensive treatment of restrictions of resolution logic based on linear resolution that serves as the basis of Prolog, a computer programming language based on deduction. No programming experience is required.

The second part of the book introduces the student to computability theory, an area of mathematical logic that should be of interest to a broad audience due to its influence on the development of the computer. There are two major goals: clarify the intuitive notion of an algorithm; and develop a methodology for proving that certain problems cannot be solved by an algorithm. Four famous problems whose solution requires an algorithm are emphasized: Hilbert's Decision Problem, Hilbert's Tenth Problem, the Halting Problem, and Thue's Word Problem. A wide range of explicit algorithms are described, after which attention is restricted to the set of natural numbers. In this setting three informal concepts are defined (each in terms of an algorithm): computable function, decidable relation, and semi-decidable relation. The first three problems mentioned above are semi-decidable (in a more general sense), but are they decidable? Two models of computation are described in considerable detail, each with the motivation of giving a precise counterpart to the three informal concepts. The first model is a machine model, namely the register machine and RM-computable functions. Turing's diagonal argument that the Halting Problem is unsolvable is given, together with an outline of his application of that result, namely that Hilbert's Decision Problem is unsolvable. The Post-Markov result that the Word Problem is unsolvable is also proved. The second model of computation is a mathematical model, the recursive functions. Precise counterparts of the three informal concepts are defined: recursive functions, recursive relations, and recursively enumerable relations. There is a detailed proof that the two models give the same class of functions. The relationship between the informal concepts and their formal counterparts, together with the important role of the Church-Turing Thesis, is emphasized.

The third part of the book consists of topics from philosophical logic, with an emphasis on the propositional calculus of a particular non-classical logic known as relevance logic. We follow Anderson and Belnap's own presentation of it here. The topic is presented by first considering some well-known theorems of classical propositional logic that clash with intuitions about the use of "if . . . then . . . ," which have been known as "paradoxes of implication." The student is invited to reflect on the features of classical logic that give rise to them. This is approached by presenting the rules for a natural deduction system for classical logic and examining which features of these rules permit derivation of the non-intuitive theorems or (so-called) paradoxes. This motivates considering alternative rules for deriving theorems, which is an occasion for a discussion of the analysis of the conditional (if . . . then . . .) and its relation to deduction and derivation. The non-classical logic known as relevance logic is presented as one such alternative. Both a natural deduction style proof system and a four-valued semantics (told true, told false, told both, told neither) for this logic are given. This is important, as some philosophers present relevance logic as a paraconsistent logic. The pedagogical approach we take here shows that is by no means mandatory: by the use of the engaging example of its application in a question-answering computer, we present a practical application in which this non-classical logic accords well with intuitions about what one would want in a logic to deal with situations in which we are faced with conflicting information. This example broadens the student's ideas of the uses and capabilities of logic. The inferential semantics is presented using a mathematical structure called a lattice. A brief introduction to mathematical lattices is provided. Then, drawing on the points in the classic paper "How a Computer Should Think," it is shown that, in certain contexts in which automated deduction is employed, relevance logic is to be preferred over classical logic. Some connections with the two earlier parts of the course on computer deduction and computability theory are made. Part 3 closes with some remarks on the impact of relevance logic in various disciplines.