

Mathematical Logic and Computation

This new book on mathematical logic by Jeremy Avigad gives a thorough introduction to the fundamental results and methods of the subject from the syntactic point of view, emphasizing logic as the study of formal languages and systems and their proper use. Topics include proof theory, model theory, the theory of computability, and axiomatic foundations, with special emphasis given to aspects of mathematical logic that are fundamental to computer science, including deductive systems, constructive logic, the simply typed lambda calculus, and type-theoretic foundations.

Clear and engaging, with plentiful examples and exercises, it is an excellent introduction to the subject for graduate students and advanced undergraduates who are interested in logic in mathematics, computer science, and philosophy, and an invaluable reference for any practicing logician's bookshelf.

JEREMY AVIGAD is Professor in the Department of Philosophy and the Department of Mathematical Sciences at Carnegie Mellon University. His research interests include mathematical logic, formal verification, automated reasoning, and the philosophy and history of mathematics. He is Director of the Charles C. Hoskinson Center for Formal Mathematics at Carnegie Mellon University.

Mathematical Logic and Computation

Jeremy Avigad
Carnegie Mellon University





Shaftesbury Road, Cambridge CB2 8EA, United Kingdom
One Liberty Plaza, 20th Floor, New York, NY 10006, USA
477 Williamstown Road, Port Melbourne, VIC 3207, Australia
314–321, 3rd Floor, Plot 3, Splendor Forum, Jasola District Centre,
New Delhi – 110025, India
103 Penang Road, #05–06/07, Visioncrest Commercial, Singapore 238467

Cambridge University Press is part of Cambridge University Press & Assessment,
a department of the University of Cambridge.

We share the University's mission to contribute to society through the pursuit of
education, learning and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/9781108478755

DOI: 10.1017/9781108778756

© Jeremy Avigad 2023

This publication is in copyright. Subject to statutory exception and to the provisions
of relevant collective licensing agreements, no reproduction of any part may take
place without the written permission of Cambridge University Press.

First published 2023

A catalogue record for this publication is available from the British Library.

Library of Congress Cataloging-in-Publication Data

Names: Avigad, Jeremy, author.

Title: Mathematical logic and computation / Jeremy Avigad, Carnegie Mellon
University, Pennsylvania.

Description: First edition. | Cambridge, United Kingdom ; Boca Raton :
Cambridge University Press, 2023. | Includes bibliographical
references and index.

Identifiers: LCCN 2022006053 | ISBN 9781108478755 (hbk) | ISBN
9781108778756 (ebook)

Subjects: LCSH: Logic, Symbolic and mathematical.

Classification: LCC QA9 .A92 2023 | DDC 511.3–dc23 / eng20220726

LC record available at <https://lcn.loc.gov/2022006053>

ISBN 978-1-108-47875-5 Hardback

Cambridge University Press has no responsibility for the persistence
or accuracy of URLs for external or third-party internet websites referred to in this
publication and does not guarantee that any content on such websites is, or will
remain, accurate or appropriate.

Contents

<i>Preface</i>	<i>page</i> ix
<i>Acknowledgments</i>	xii
1 Fundamentals	1
1.1 Languages and Structures	2
1.2 Inductively Defined Sets	4
1.3 Terms and Formulas	8
1.4 Trees	11
1.5 Structural Recursion	15
1.6 Bound Variables	21
2 Propositional Logic	28
2.1 The Language of Propositional Logic	28
2.2 Axiomatic Systems	30
2.3 The Provability Relation	34
2.4 Natural Deduction	36
2.5 Some Propositional Validities	42
2.6 Normal Forms for Classical Logic	47
2.7 Translations Between Logics	50
2.8 Other Deductive Systems	53
3 Semantics of Propositional Logic	59
3.1 Classical Logic	60
3.2 Algebraic Semantics for Classical Logic	65
3.3 Intuitionistic Logic	70
3.4 Algebraic Semantics for Intuitionistic Logic	75
3.5 Variations	79
4 First-Order Logic	83
4.1 The Language of First-Order Logic	83
4.2 Quantifiers	85
4.3 Equality	90
4.4 Equational and Quantifier-Free Logic	94
4.5 Normal Forms for Classical Logic	96

4.6	Translations Between Logics	97
4.7	Definite Descriptions	101
4.8	Sorts and Undefined Terms	105
5	Semantics of First-Order Logic	110
5.1	Classical Logic	110
5.2	Equational and Quantifier-Free Logic	117
5.3	Intuitionistic Logic	118
5.4	Algebraic Semantics	123
5.5	Definability	127
5.6	Some Model Theory	132
6	Cut Elimination	138
6.1	An Intuitionistic Sequent Calculus	138
6.2	Classical Sequent Calculi	143
6.3	Cut-Free Completeness of Classical Logic	147
6.4	Cut Elimination for Classical Logic	151
6.5	Cut Elimination for Intuitionistic Logic	156
6.6	Equality	158
6.7	Variations on Cut Elimination	161
6.8	Cut-Free Completeness of Intuitionistic Logic	162
7	Properties of First-Order Logic	166
7.1	Herbrand's Theorem	166
7.2	Explicit Definability and the Disjunction Property	172
7.3	Interpolation Theorems	174
7.4	Indefinite Descriptions	180
7.5	Skolemization in Classical Theories	186
8	Primitive Recursion	192
8.1	The Primitive Recursive Functions	192
8.2	Some Primitive Recursive Functions and Relations	195
8.3	Finite Sets and Sequences	199
8.4	Other Recursion Principles	204
8.5	Recursion along Well-Founded Relations	208
8.6	Diagonalization and Reflection	210
9	Primitive Recursive Arithmetic	214
9.1	A Quantifier-Free Axiomatization	214
9.2	Bootstrapping PRA	218
9.3	Finite Sets and Sequences	224
9.4	First-Order PRA	229
9.5	Equational PRA	231

Contents

vii

10	First-Order Arithmetic	237
10.1	Peano Arithmetic and Heyting Arithmetic	237
10.2	The Arithmetic Hierarchy	240
10.3	Subsystems of First-Order Arithmetic	247
10.4	Interpreting PRA	252
10.5	Truth and Reflection	258
10.6	Conservation Results via Cut Elimination	266
10.7	Conservation Results via Model Theory	272
11	Computability	279
11.1	The Computable Functions	279
11.2	Computability and Arithmetic Definability	281
11.3	Undecidability and the Halting Problem	284
11.4	Computably Enumerable Sets	290
11.5	The Recursion Theorem	293
11.6	Turing Machines	296
11.7	The Lambda Calculus	301
11.8	Relativized Computation	308
11.9	Computability and Infinite Binary Trees	314
12	Undecidability and Incompleteness	322
12.1	Computability and Representability	323
12.2	Incompleteness via Undecidability	329
12.3	Incompleteness via Self-Reference	334
12.4	The Second Incompleteness Theorem	336
12.5	Some Decidable Theories	341
13	Finite Types	351
13.1	The Simply Typed Lambda Calculus	351
13.2	Strong Normalization	356
13.3	Confluence	362
13.4	Combinatory Logic	364
13.5	Equational Theories	368
13.6	First-Order Theories and Models	371
13.7	Primitive Recursive Functionals	376
13.8	Propositions as Types	379
14	Arithmetic and Computation	383
14.1	Realizability	383
14.2	Metamathematical Applications	389
14.3	Modified Realizability	394
14.4	Finite-Type Arithmetic	399
14.5	The Dialectica Interpretation	401

15	Second-Order Logic and Arithmetic	409
15.1	Second-Order Logic	410
15.2	Semantics of Second-Order Logic	413
15.3	Cut Elimination	417
15.4	Second-Order Arithmetic	422
15.5	The Analytic Hierarchy	426
15.6	The Second-Order Typed Lambda Calculus	430
15.7	Higher-Order Logic and Arithmetic	437
16	Subsystems of Second-Order Arithmetic	440
16.1	Arithmetic Comprehension	441
16.2	Recursive Comprehension	446
16.3	Formalizing Analysis	449
16.4	Weak König's Lemma	457
16.5	Π_1^1 Comprehension and Inductive Definitions	461
16.6	Arithmetic Transfinite Recursion	464
17	Foundations	472
17.1	Simple Type Theory	472
17.2	Mathematics in Simple Type Theory	475
17.3	Set Theory	475
17.4	Mathematics in Set Theory	477
17.5	Dependent Type Theory	478
17.6	Inductive Types	481
17.7	Mathematics in Dependent Type Theory	485
Appendix	Background	488
A.1	Naive Set Theory	488
A.2	Orders and Equivalence Relations	490
A.3	Cardinality and Zorn's Lemma	491
A.4	Topology	493
A.5	Category Theory	495
	<i>References</i>	497
	<i>Notation</i>	504
	<i>Index</i>	508

Preface

In the phrase *mathematical logic*, the word “mathematical” is ambiguous. It can be taken to specify the methods used, so that the phrase refers to the mathematical study of the principles of reasoning. It can be taken to demarcate the type of reasoning considered, so that the phrase refers to the study of specifically mathematical reasoning. Or it can be taken to indicate the purpose, so that the phrase refers to the study of logic with an eye toward mathematical applications.

In the title of this book, the word “mathematical” is intended in the first two senses but not in the third. In other words, mathematical logic is viewed here as a mathematical study of the methods of mathematical reasoning. The subject is interesting in its own right, and it has mathematical applications. But it also has applications in computer science, for example, to the verification of hardware and software and to the mechanization of mathematical reasoning. It can inform the philosophy of mathematics as well, by providing idealized models of what it means to do mathematics.

One thing that distinguishes logic as a discipline is its focus on language. The subject starts with formal expressions that are supposed to model the informal language we use to define objects, state claims, and prove them. At that point, two distinct perspectives emerge. From a *semantic* perspective, the formal expressions are used to say things about abstract mathematical objects and structures. They can be used to characterize classes of structures like groups, rings, and fields; to characterize particular structures, like the Euclidean plane or the real numbers; or to describe relationships within a particular structure. From that point of view, mathematical logic is the science of reference, definability, and truth, clarifying the semantic notions that determine the relationship between mathematical language and the mathematical structures it describes.

This book adopts a more *syntactic* perspective, in which the primary objects of interest are the expressions themselves. From that point of view, formal languages are used to reason and calculate, and what we care about are the rules that govern their proper use. We will use formal systems to understand patterns of mathematical inference and the structure of mathematical definitions and proofs, and we will be interested in the things that we can do with these syntactic representations. We won’t shy away from the use of semantic methods, but our goal is to use semantics to illuminate syntax rather than the other way around.

There are a number of reasons why a syntactic approach is valuable. The mathematical theory of syntax is independently interesting and informative. A focus on syntactic objects is also more closely aligned with computer science, since these objects can be represented as data and acted on by algorithms. Finally, there are philosophical benefits. Because a general theory of finite strings of symbols is all that is needed to work with expressions, a syntactic perspective provides a means of studying mathematical reasoning – including the use of

infinitary objects and methods – without importing strong mathematical presuppositions at the outset.

Another notable feature of this book is its focus on computation. On the one hand, we expect mathematics to give us a broad conceptual understanding. At the empirical borders of the subject, this serves to organize and explain our scientific observations, but our desire for understanding is not limited to empirical phenomena. On the other hand, we also expect mathematics to tell us how to calculate trajectories and probabilities so that we can make better predictions and decisions, and act rationally toward securing our pragmatic goals. There is a tension between conceptual understanding and calculation: computation is important, but we often see further, and reason more efficiently, by suppressing computational detail.

The tension is partially captured by the logician’s distinction between *classical* logic, on the one hand, and *intuitionistic* or *constructive* logic, on the other. Classical logic is meant to support a style of reasoning that supports abstraction and idealization, whereas intuitionistic logic is more directly suited to computational interpretation. Mathematics today is resolutely classical, but mathematics without calculation is almost a contradiction in terms. And while the end goal of computer science is practical computation, abstraction is essential to designing and reasoning about computational systems. Here we will study both classical and constructive logic, with an eye toward understanding the relationships between the two.

The connections between logic and computation run deep: we can compute with formal expressions, we can reason formally about computation, and we can extract computational content from formal derivations. The style of presentation I have adopted here runs the risk of being judged too mathematical by computer scientists and too computationally minded for pure mathematicians, but I have aimed to strike a balance and, hopefully, bring the two communities closer together.

Audience The material here should be accessible at an advanced undergraduate or introductory graduate level. I have tried to keep the presentation self-contained, but the emphasis is on proving things about logical systems rather than illustrating and motivating their use. Readers who have had a prior introduction to logic will be able to navigate the material here more quickly and comfortably.

Notation Most of the notation in this book is conventional, but challenges inevitably arise when juxtaposing material from fields where conventions differ. The most striking instance of such a challenge is the use of *binders* like quantifiers and lambda abstraction: mathematical logicians generally take such operations to bind tightly, while computer scientists typically give them the widest scope possible. Another mismatch arises with respect to function application, since theoretical computer scientists often write $f x$ instead of $f(x)$. So, where a mathematical logician would write

$$\exists x A(x) \rightarrow \exists x (A(x) \wedge \forall y (R(y, x) \rightarrow \neg A(y))),$$

a computer scientist might write

$$(\exists x. A x) \rightarrow \exists x. A x \wedge \forall y. R y x \rightarrow \neg A y.$$

As a compromise, this book uses the mathematician’s notation for symbolic logic but adopts the computer scientist’s conventions for computational systems like the simply typed lambda

calculus. The differences are most pronounced in Chapters 14 and 17, where logic and type theory come together.

Teaching The material in this book can be used to teach a number of different courses, and the dependencies between topics are mild. Chapters 1–7 provide a thorough introduction to the syntax and semantics of first-order logic. Classically minded mathematicians can easily tune out anything to do with intuitionistic logic, while computer scientists and other interested parties can spend more time with it. Ignoring intuitionistic logic might leave time for the cut elimination theorem and its applications, while skipping cut elimination might leave time to dabble in algebraic semantics.

For students already familiar with propositional and first-order logic, Chapters 8–14 provide a fairly self-contained presentation of formal arithmetic, computability, and the incompleteness theorems. A course on computability and incompleteness can start with Chapter 8, skip Section 8.5, continue to Chapter 11, refer back to Sections 10.2 and 10.5 for background on arithmetic definability, skip Sections 11.8 and 11.9, and then move on to Chapter 12.

A course on proof theory can focus on the cut elimination theorem and its applications (Chapters 6 and 7) and the simply typed lambda calculus (Chapter 13). Chapter 14 can serve as the focus of a course on computational interpretations of arithmetic, building on material in Chapters 8–11 and 13; for that purpose, Sections 8.5, 9.5, 10.6–10.7, and 11.6–11.9 can be omitted. Similarly, Chapter 16 can serve as the focus for an introduction to subsystems of second-order arithmetic and reverse mathematics, supported by a quick tour of Chapters 8–11 and Sections 15.4 and 15.5. Sections 11.8 and 11.9, in particular, were written with Chapter 16 in mind.

Most of the material after Chapter 7 is not strongly tied to any particular deductive system. Natural deduction is sometimes used by default, but that choice is not essential to the exposition.

Acknowledgments

I have had the privilege of learning logic with colleagues, mentors, and countless students over the years. Among those to whom I am indebted for comments on drafts of this book are Seulkee Baek, Krishan Canzian, Mario Carneiro, Lucas Clark, Ulrich Kohlenbach, Gustavo Lacerda, Fernando Langlois, Steffen Lempp, Oualid Merzouga, Paulo Oliva, Jason Rute, Alex Smith, Henry Towsner, and Jin Wei. This list is woefully incomplete. I owe a special thanks to Jasmin Blanchette, Madeleine Harlow, and Stephen Mackereth for exceptionally close readings and detailed corrections. The derivations in this book have been typeset in L^AT_EX by Samuel Buss's *Bussproofs* package.

The bibliographical notes at the end of each chapter implicitly acknowledge another debt. I myself learned the topics presented here from a number of excellent sources, including Shoenfield's *Mathematical Logic*, *The Handbook of Mathematical Logic*, Chang and Keisler's *Model Theory*, Troelstra and Van Dalen's *Constructive Mathematics*, Troelstra and Schwichtenberg's *Basic Proof Theory*, Hájek and Pudlák's *Metamathematics of First-Order Arithmetic*, Soare's *Recursively Enumerable Sets and Degrees*, Girard's *Proofs and Types*, Hindley and Seldin's *Lambda-Calculus and Combinators*, Troelstra's epic *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, and Simpson's *Subsystems of Second-Order Arithmetic*. I have drawn extensively on all of these sources and many others. I hope that my presentation does the material justice and helps keep it accessible to the next generation of logicians.

I am indebted to numerous colleagues, including Steve Awodey and Wilfried Sieg, for sharing their insights and expertise. I am especially indebted to Jack Silver and Solomon Feferman, who introduced me to logic and set me on my way. Finally, this book is dedicated to my wife, Elinor, and to my daughters, Rebecca, Jordana, and Ariella. They are the ones who make it all worthwhile.