

Mathematics of Discrete Structures for Computer Science

Gordon J. Pace

Mathematics of Discrete Structures for Computer Science

 Springer

Gordon J. Pace
Department of Computer Science,
Faculty of Information and
Communication Technology
University of Malta
Msida, Malta

ISBN 978-3-642-29839-4

ISBN 978-3-642-29840-0 (eBook)

DOI 10.1007/978-3-642-29840-0

Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012940954

ACM Computing Classification (1998): F.4, F.2, G.2

© Springer-Verlag Berlin Heidelberg 2012

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

*Dedicated to Claudia
and to my parents*

Foreword

Profound knowledge and skills in discrete mathematics are mandatory for a well educated computer scientist. Therefore a corresponding curriculum typically requires at least a basic course on this subject. This textbook, which is based on the lectures given by the author at the University of Malta, is a perfect companion for every student taking such a course.

When teaching a formal computer science topic one can either follow a so-called descriptive or algorithmic approach. Traditionally most textbooks on discrete mathematics follow the descriptive approach meaning that the properties of mathematical structures are listed first while the algorithms to compute corresponding results are of secondary interest. This textbook follows the algorithmic approach which seems to be better suited for computer science students: Mathematical structures are introduced and defined using algorithms which are then characterized by their properties in a second step.

The book covers the material that is essential knowledge for a computer science student. Starting with propositional and first order logic it discusses sets, relations and further discrete mathematical structures. It contains a chapter on type theory before introducing the natural numbers. In the final chapter it discusses how to reason about programs and gives the basics of computability theory.

Besides a thorough formal presentation the book always gives a convincing motivation for studying the corresponding structures and explains the main ideas using illustrations. A well chosen set of exercises rounds off each topic. After studying the presented material a computer science student will be well prepared for further, more specialized courses.

Lübeck, Germany

Martin Leucker

Preface

In computer science, as in other sciences, mathematics plays an important role. In fact, computer science has more in common with mathematics than with the traditional sciences. Consider physics, for example. Clearly, mathematics is right at the core of the subject, used to describe models which explain and predict the physical world. Without mathematics, most of modern physics would have been impossible to develop, since a purely qualitative analysis does not allow one to reason and deduce what logically follows from an observation or a hypothesised model. No manner of qualitative reasoning would have enabled Newton to formulate his law of gravity based on Kepler's claims about planetary motion. Computer science also exploits mathematical reasoning in a similar manner. Given a computer system, one can apply mathematical reasoning to prove that it will always calculate your paycheck correctly.

However, the link is even tighter than this. Computer science is a direct descendant of mathematics, and computers are nothing but the physical embodiment of mathematical systems developed long before the first modern computer was even conceived, let alone built.

In this book, we will be exploring the foundational mathematics which is necessary for the understanding of more advanced courses in computer science. Whether you are designing a digital circuit, a computer program or a new programming language, mathematics is needed to reason about the design—its correctness, robustness and dependability.

There are two distinct approaches used to present mathematical concepts and operators. In the first approach concepts and operators are defined in terms of properties which they satisfy. Based on these definitions, ways of computing the result of applying these operators are then developed and proved to be correct. In contrast, in computer science frequently one takes the opposite approach—one starts by defining ways of calculating the result of an operator, and then proves that the operator one knows how to compute satisfies a number of properties. The two approaches are, in the end, equivalent. However, given that this book is aimed at computer science students, the latter approach is usually adopted.

Finally, most sections are accompanied by exercises which are usually necessary as part of the learning process. Learning mathematics is surprisingly like learning to ride a bicycle. You must try to write formal expressions and proofs before you really understand the concepts you read about. Some of the concepts covered in this book may initially appear to be difficult. However, as you progress through the book and familiarise yourself with the concepts, you will hopefully start requiring less effort to follow the definitions and proofs, and begin to enjoy the beauty of how various concepts in mathematics can be built and reasoned about, based on just a small number of basic concepts.

Buenos Aires, Argentina

Gordon J. Pace

Acknowledgements

The original inspiration which led to this book dates back almost two decades, when I followed a course on mathematics for system specification lectured by Joe Stoy at the University of Oxford. His approach of building up the necessary mathematical machinery from the ground up, akin to building a software system, provided the spark which years later led to the design of the structure and content of a course in mathematics of discrete structures, and consequently to this book. Without Joe Stoy's spark, this book would never have been.

This book has been long in the making. Life kept getting in the way. Many students would have benefitted had the book been finished earlier. Instead, they were exposed to the intended content in my lectures—questions and blank faces exposing the parts of the book which I had to rethink. Thanks go out to those hundreds of students without whom the book would have been much poorer.

Thanks also go out to those people who used initial drafts of the book to deliver and support taught courses in mathematics of discrete structures: Christian Colombo, Adrian Francalanza, Ingram Bondin, Mark Bonnici, Andrew Calleja, Karlston Demanuele, Kevin Falzon, Andrew Gauci and Christine Vella. Their feedback was invaluable.

Finally, the shortest thanks go out to those whose support can never be fully acknowledged, no matter how long I write about it—Claudia, Dodie and Josie. Thank you.

Contents

- 1 Why Mathematics?** 1
 - 1.1 What Is Mathematics? 2
 - 1.2 A Historical Perspective 2
 - 1.3 On the Superiority of Formal Reasoning 4
 - 1.4 The Mathematics of Computing 4
 - 1.4.1 The Analysis of Programs 5
 - 1.4.2 Formal Specification of Requirements 6
 - 1.4.3 Reasoning About Programs 6
 - 1.4.4 The Limits of Computing 7
 - 1.4.5 A Physical Embodiment of Mathematics 8
 - 1.5 Terminology 8
- 2 Propositional Logic** 9
 - 2.1 What Is a Proposition? 9
 - 2.2 The Language of Discourse 10
 - 2.3 Model Theory 15
 - 2.3.1 Truth Tables: Propositional Operators 16
 - 2.3.2 Properties of Propositional Sentences 22
 - 2.3.3 Conclusion 28
 - 2.4 Proof Theory 29
 - 2.4.1 What Is a Proof? 29
 - 2.4.2 Example Axioms and Rules of Inference 30
 - 2.4.3 Why Proofs? 47
 - 2.5 Comparing Truth Tables and Proofs 47
 - 2.5.1 Completeness 48
 - 2.5.2 Soundness 48
 - 2.6 More About Propositional Logic 48
 - 2.6.1 Boolean Algebra 48
 - 2.6.2 Doing with Less 50
 - 2.6.3 Doing with Even Less 51
 - 2.6.4 What if the Axioms Are Wrong? 52

2.7	Some Notes	53
2.7.1	Notation and Symbols	53
2.7.2	Proof Elegance	54
2.7.3	How to Come up with a Proof	55
2.7.4	Not All Proofs Are Equal	55
3	Predicate Calculus	57
3.1	The Limits of Propositional Logic	57
3.2	The Language of Discourse	58
3.3	The Use of Variables	61
3.3.1	Free Variables	62
3.3.2	Substitution of Variables	64
3.4	Axiomatisation of Predicate Logic	67
3.4.1	Universal Quantification	67
3.4.2	Existential Quantification	69
3.4.3	Existential Quantification and Equality	76
3.5	Beyond Predicate Calculus	76
4	Sets	79
4.1	What Are Sets?	79
4.2	Queries About Sets	80
4.3	Comparing Sets	81
4.3.1	Subsets	81
4.3.2	Set Equality	83
4.4	Constructing Sets	84
4.4.1	Finite Sets	84
4.4.2	The Empty Set	85
4.4.3	Set Comprehensions	86
4.5	Set Operators	90
4.5.1	Set Union	92
4.5.2	Set Intersection	93
4.5.3	Set Complement	95
4.5.4	Set Difference	97
4.5.5	Other Properties of the Set Operators	98
4.5.6	Generalised Set Operators	99
4.6	Sets as a Boolean Algebra	102
4.7	More About Types	104
4.7.1	Types and Sets	104
4.7.2	On Why Types Are Desirable: Russell's Paradox	108
4.8	Summary	110
5	Relations	111
5.1	An Informal View of Relations	111
5.2	Formalising Relations	112
5.2.1	The Type of a Relation	113
5.2.2	Some Basic Relations	114

5.2.3	Relational Equality	115
5.2.4	Combining Relations as Sets	115
5.3	Domain and Range	118
5.4	Building New Relations from Old Ones	119
5.4.1	Composition of Relations	119
5.4.2	Relational Inverse	123
5.4.3	Repeated Relations	127
5.4.4	Closure	131
5.5	Other Relational Operators	134
5.6	Beyond Binary Relations	138
5.7	Summary	139
6	Classifying Relations	141
6.1	Classes of Relations	141
6.1.1	Totality	141
6.1.2	Surjectivity	143
6.1.3	Injectivity	144
6.1.4	Functionality	146
6.1.5	Combining the Results	148
6.2	Relating a Type to Itself	149
6.2.1	Properties of These Relations	149
6.2.2	Order-Inducing Relations	152
6.2.3	Equivalence Relations	153
6.3	Summary	154
7	More Discrete Structures	157
7.1	Multisets	157
7.2	Sequences	160
7.3	Graph Theory	164
8	Defining New Structured Types	175
8.1	Notions and Notation	176
8.1.1	Simple Enumerated Types	176
8.1.2	More Elaborate Types	177
8.1.3	Self-Referential Types	179
8.1.4	Parametrised Types	181
8.2	Reasoning About New Types	184
8.2.1	Axiomatising New Types	184
8.2.2	A General Inductive Principle	188
8.2.3	Structural Induction	193
8.3	Using Structured Types	194
8.3.1	Three-Valued Logic	194
8.3.2	Processing Data	198
8.3.3	Lists	202
8.3.4	Binary Trees	206
8.4	Summary	210

9	Numbers	211
9.1	Natural Numbers	211
9.1.1	Defining the Counting Numbers	212
9.1.2	Defining the Arithmetic Operators	212
9.1.3	Strong Induction	222
9.1.4	Division and Prime Numbers	224
9.2	Beyond the Natural	234
9.2.1	Integers	235
9.2.2	The Rational Numbers	237
9.2.3	The Real Numbers	237
9.3	Cardinality	239
9.3.1	Counting with Finite Sets	239
9.3.2	Extending Cardinality to Infinite Sets	249
9.4	Summary	255
10	Reasoning About Programs	257
10.1	Correctness of Algorithms	258
10.1.1	Euclid's Algorithm	259
10.1.2	Sorted Binary Trees	263
10.2	Assigning Meaning to Programs	268
10.2.1	Numeric Expressions	268
10.2.2	Program Semantics	275
10.3	The Uncomputable	280
10.3.1	Counting Computer Programs	280
10.3.2	Sets of Numbers	281
10.3.3	Parsing Languages	282
10.3.4	The Halting Problem	283
10.4	Summary	285
	Index	287