

Undergraduate Topics in Computer Science

Undergraduate Topics in Computer Science' (UTiCS) delivers high-quality instructional content for undergraduates studying in all areas of computing and information science. From core foundational and theoretical material to final-year topics and applications, UTiCS books take a fresh, concise, and modern approach and are ideal for self-study or for a one- or two-semester course. The texts are all authored by established experts in their fields, reviewed by an international advisory board, and contain numerous examples and problems. Many include fully worked solutions.

For other titles published in this series, go to
<http://www.springer.com/7592>

David Makinson

Sets, Logic and Maths for Computing

 Springer

David Makinson, DPhil
London School of Economics, UK

Series editor

Ian Mackie, École Polytechnique, France and King's College London, UK

Advisory board

Samson Abramsky, University of Oxford, UK
Chris Hankin, Imperial College London, UK
Dexter Kozen, Cornell University, USA
Andrew Pitts, University of Cambridge, UK
Hanne Riis Nielson, Technical University of Denmark, Denmark
Steven Skiena, Stony Brook University, USA
Iain Stewart, University of Durham, UK
David Zhang, The Hong Kong Polytechnic University, Hong Kong

Undergraduate Topics in Computer Science ISSN 1863-7310
ISBN: 978-1-84628-844-9 e-ISBN: 978-1-84628-845-6
DOI: 10.1007/978-1-84628-845-6

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2008927215

© Springer-Verlag London Limited 2008

Apart from any fair dealing for the purposes of research or private study, or criticism or review, as permitted under the Copyright, Designs and Patents Act 1988, this publication may only be reproduced, stored or transmitted, in any form or by any means, with the prior permission in writing of the publishers, or in the case of reprographic reproduction in accordance with the terms of licences issued by the Copyright Licensing Agency. Enquiries concerning reproduction outside those terms should be sent to the publishers.

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

The publisher makes no representation, express or implied, with regard to the accuracy of the information contained in this book and cannot accept any legal responsibility or liability for any errors or omissions that may be made.

Printed on acid-free paper

Springer Science+Business Media
springer.com

Preface

The first part of this preface is for the student; the second for the instructor. But whoever you are, welcome to both parts.

For the Student

You have finished secondary school, and are about to begin at a university or technical college. You want to study computing. The course includes some mathematics – and that was not necessarily your favourite subject. But there is no escape: some finite mathematics is a required part of the first year curriculum.

That is where this book comes in. Its purpose is to provide the basics – the essentials that you need to know to understand the mathematical language that is used in computer and information science.

It does not contain all the mathematics that you will need to look at through the several years of your undergraduate career. There are other very good, massive volumes that do that. At some stage you will probably find it useful to get one and keep it on your shelf for reference. But experience has convinced this author that no matter how good the compendia are, beginning students tend to feel intimidated, lost, and unclear about what parts to focus on. This short book, on the other hand, offers just the basics which you need to know from the beginning, and on which you can build further when needed.

It also recognizes that you may not have done much mathematics at school, may not have understood very well what was going on, and may even have grown

to detest it. No matter: you can re-learn the essentials here, and perhaps even have fun doing so.

So what is the book about? It is about certain mathematical tools that we need to apply over and over again when working with computations. They include:

- *Collecting* things together. In the jargon of mathematics, this is *set theory*.
- *Comparing* things. This is the theory of *relations*.
- *Associating* one item with another. This is the theory of *functions*.
- *Recycling outputs as inputs*. We introduce the ideas of *induction* and *recursion*.
- *Counting*. The mathematician's term is *combinatorics*.
- *Weighing the odds*. This is done with the notion of *probability*.
- *Squirrel math*. Here we look at the use of *trees*.
- *Yea and Nay*. Just two truth-values underlie *propositional logic*.
- *Everything and nothing*. That is what *quantificational logic* is about.

Frankly, without an understanding of the basic concepts in these areas, you will not be able to acquire more than a vague idea of what is going on in computer science, nor be able to make computing decisions with discrimination or creatively. Conversely, as you begin to grasp them, you will find that their usefulness extends far beyond computing into many other areas of thought.

The good news is that there is not all that much to commit to memory. Your sister studying medicine, or brother going for law, will envy you terribly for this. In our subject, the essential thing in our kind of subject is to *understand*, and be able to *apply*.

But that is a much more subtle affair than you might imagine. Understanding and application are interdependent. Application without understanding is blind, and quickly leads to ghastly errors. On the other hand, comprehension remains poor without practice in application. In particular, you do not understand a definition until you have seen how it takes effect in specific situations: positive examples reveal its range, negative examples show its limits. It also takes time to recognize *when* you have really understood something, and when you have done no more than recite the words, or call upon it in hope of blessing.

For this reason, doing exercises is a indispensable part of the learning process. That is part of what is meant by the old proverb 'there is no royal road in mathematics'. It is also why we give so many problems and provide sample answers to some. Skip them at your peril: no matter how simple and straightforward a concept seems, you will not fully understand it unless you practice using it. So, even when an exercise is accompanied by a solution, you will benefit a great

deal if you place a sheet over the answer and first try to work it out for yourself. That requires self-discipline and patience, but it brings real rewards.

In addition, the exercises have been chosen so that for many of them, the result is just what we need to make a step somewhere later in the book. They are thus integral to the development of the general theory.

By the same token, don't get into the habit of skipping the proofs when you read the text. In mathematics, you have never fully understood a fact unless you have also grasped *why* it is true, i.e. have assimilated at least one proof of it. The well-meaning idea that mathematics can be democratized by teaching the 'facts' and forgetting about the proofs has, in some countries, wrought disaster in secondary and university education in recent decades.

In practice, the mathematical tools that we bulleted above are rarely applied in isolation from each other. They gain their real power when used in combination, setting up a crossfire that can bring tough problems to the ground. The concept of a set, once explained in the first chapter, is used absolutely everywhere in what follows. Relations reappear in graphs and trees. The familiar arithmetical operations of addition and multiplication, heavily employed in combinatorics and probability, are of course particular examples of functions. And so on.

For the Instructor

Any book of this kind needs to find a delicate balance between the competing demands of intrinsic mathematical order and those of intuition. Mathematically, the most elegant and coherent way to proceed is to begin with the most general concepts, and gradually unfold them so that the more specific and familiar ones appear as special cases. Pedagogically, this sometimes works, but it can also be disastrous. There are situations where the reverse is often required: begin with some of the more familiar special cases, and then show how they may naturally be broadened into cover much wider terrain.

There is no perfect solution to this problem; we have tried to find a least imperfect one. Insofar as we begin the book with sets, relations and functions in that order, we are following the first path. But in some chapters we have followed the second one. For example, when explaining induction and recursion we begin with the most familiar special case, simple induction/recursion over the positive integers; passing to their cumulative forms over the same domain; broadening to their qualitatively formulated structural versions; and finally presenting the most general forms on arbitrary well-founded sets. Again, in the chapter on trees, we have taken the rather unusual step of beginning with rooted trees, where intuition is strongest, then abstracting to unrooted trees.

In the chapters on counting and probability we have had to strike another balance – between traditional terminology and notation, which antedates the modern era, and its translation into the language of sets, relations and functions. Most textbook presentations do everything the traditional way, which has its drawbacks. It leaves the student in the dark about the relation of this material to what was taught in earlier chapters on sets, relations and functions. And, frankly, it is not always very rigorous or transparent. Our policy is to familiarize the reader with *both* kinds of presentation – using the language of sets and functions for a clear understanding of the material itself, and the traditional languages of combinatorics and probability to permit communication in the local dialect.

The place of logic in the story is delicate. We have left its systematic exposition to the end, a decision that may seem rather strange. For surely one uses logic whenever reasoning mathematically, even about such elementary things as sets, relations and functions, covered in the first three chapters. Don't we need a chapter on logic at the very beginning? The author's experience in the classroom tells him that in practice that does not work well. Despite its simplicity – indeed because of it – logic can appear intangible for beginning students. It acquires intuitive meaning only as its applications are revealed. Moreover, it turns out that a really clear explanation of the basic concepts of logic requires some familiarity with the mathematical notions of sets, relations, functions and trees.

For these reasons, the book takes a different tack. In its early chapters, notions of logic are identified briefly as they arise in the discussion of more 'concrete' material. This is done in 'logic boxes'. Each box introduces just enough to get on with the task in hand. Much later, in the last two chapters, all this is brought together and extended in a systematic presentation. By then, the student will have little trouble appreciating what the subject is all about, and how natural it all is.

From time to time there are boxes of a different nature – 'Alice boxes'. This little trouble-maker comes from the pages of Lewis Carroll, to ask embarrassing questions in all innocence. Often they are questions that bother students, but which they have trouble articulating clearly or are too shy to pose. In particular, it is a fact that the house of mathematics and logic can be built in many different ways, and sometimes the constructions of one text appear to be in conflict with those of another. Perhaps the most troublesome example of this comes up in quantificational logic, with different ways of reading the quantifiers and even different ways of using the terms 'true' and 'false'. But there are also plenty of others, in all the chapters. It is hoped that the Mad Hatter's responses are of assistance.

Overall, our choice of topics is fairly standard, as the chapter titles indicate. If strapped for class time, an instructor could omit some of the later sections of Chapters 5–9, perhaps even from the end of Chapter 4. But it is urged that

Chapters 1–3 be kept intact, as everything in them is subsequently needed. Some instructors may wish to add or deepen topics; as this text makes no pretence to cover all possible areas of focus, such extensions are left to their discretion.

We have not included a chapter on the theory of graphs. This was a difficult call to make, and the reasons were as follows. Although trees are a particular kind of graph, there is no difficulty in covering everything we want to say about trees, without entering into the more general theory. Moreover, an adequate treatment of graphs, even if squeezed into one chapter of about the same length as the others, takes a good two weeks of additional class time to cover properly. The general theory of graphs is a rather messy area, with options about how wide to cast the net (graphs with or without loops, multiple edges etc as well as the basic distinction between directed and undirected graphs), and a rather high definition/theorem ratio. The author's experience is that students gain little from a high-speed run through these distinctions and definitions, memorized for the examinations and then promptly forgotten.

Finally, a decision had to be made whether to include specific algorithms and, if so, in what form: ordinary English, pseudo-code outline, or a real-life programming language in full detail? Our decision has been to leave options open as much as possible. In principle, most first year students of computing will be taking, in parallel, courses on principles of programming and some specific programming language. But the programming languages chosen will differ from one institution to another. The policy in this text is to sketch the essential idea of basic algorithms in plain but carefully formulated English. In some cases (particularly the chapter on trees), we give optional exercises in expressing them in pseudo-code. Instructors wishing to make more systematic use of pseudo-code, or to link material with specific programming languages, should feel free to do so.

Acknowledgements

The author would like to thank Anatoli Degtyarev, Franz Dietrich, Valentin Goranko and George Kourousias for helpful discussions. George also helped prepare the diagrams. Thanks as well to the London School of Economics (LSE), and particularly department heads Colin Howson and Richard Bradley, for the wonderful working environment that they provided while this book was being written. Thanks as well to the London School of Economics (LSE), and particularly department heads Colin Howson and Richard Bradley, for the wonderful working environment that they provided while this book was being written.

Contents

1. Collecting Things Together: Sets.	1
1.1 The Intuitive Concept of a Set	1
1.2 Basic Relations between Sets	2
1.2.1 Inclusion	2
1.2.2 Identity	4
1.2.3 Proper Inclusion	5
1.2.4 Euler Diagrams	6
1.2.5 Venn Diagrams	6
1.2.6 Ways of Defining a Set	7
1.3 The Empty Set	11
1.3.1 Emptiness	11
1.3.2 Disjoint Sets	12
1.4 Boolean Operations on Sets	12
1.4.1 Intersection	12
1.4.2 Union	14
1.4.3 Difference and Complement	17
1.5 Generalised Union and Intersection	20
1.6 Power Sets	22
1.7 Some Important Sets of Numbers	25
2. Comparing Things: Relations.	29
2.1 Ordered Tuples, Cartesian Products, Relations	29
2.1.1 Ordered Tuples	30
2.1.2 Cartesian Products	31
2.1.3 Relations	34
2.2 Tables and Digraphs for Relations	36
2.2.1 Tables for Relations	36
2.2.2 Digraphs for Relations	37
2.3 Operations on Relations	38
2.3.1 Converse	39
2.3.2 Join of Relations	40
2.3.3 Composition of Relations	42
2.3.4 Image	44
2.4 Reflexivity and Transitivity	46

2.4.1	Reflexivity	46
2.4.2	Transitivity	47
2.5	Equivalence Relations and Partitions	48
2.5.1	Symmetry.	48
2.5.2	Equivalence Relations	49
2.5.3	Partitions.	51
2.5.4	The Correspondence between Partitions and Equivalence Relations	52
2.6	Relations for Ordering	54
2.6.1	Partial Order	54
2.6.2	Linear Orderings	55
2.6.3	Strict Orderings	56
2.7	Closing with Relations	58
2.7.1	Transitive Closure of a Relation	58
2.7.2	Closure of a Set under a Relation	59
3.	Associating One Item with Another: Functions.	63
3.1	What is a Function?	63
3.2	Operations on Functions	66
3.2.1	Domain and Range	66
3.2.2	Image, Restriction, Closure	67
3.2.3	Composition.	69
3.2.4	Inverse	70
3.3	Injectons, Surjections, Bijections.	71
3.3.1	Injectivity.	71
3.3.2	Surjectivity	72
3.3.3	Bijective Functions	74
3.4	Using Functions to Compare Size	75
3.4.1	The Equinumerosity Principle.	75
3.4.2	The Principle of Comparison.	76
3.4.3	The Pigeonhole Principle.	77
3.5	Some Handy Functions.	79
3.5.1	Identity Functions.	79
3.5.2	Constant Functions	80
3.5.3	Projection Functions	81
3.5.4	Characteristic Functions	81
3.5.5	Families of Sets	81
3.5.6	Sequences.	82
4.	Recycling Outputs as Inputs: Induction and Recursion.	87
4.1	What are Induction and Recursion?.	88
4.2	Proof by Simple Induction on the Positive Integers.	89
4.2.1	An Example	89
4.2.2	The Principle behind the Example	90
4.3	Definition by Simple Recursion on the Natural Numbers	93

4.4	Evaluating Functions Defined by Recursion	96
4.5	Cumulative Induction and Recursion.	97
4.5.1	Recursive Definitions Reaching Back more than One Unit. .	97
4.5.2	Proof by Cumulative Induction.	99
4.5.3	Simultaneous Recursion and Induction.	101
4.6	Structural Recursion and Induction.	103
4.6.1	Defining Sets by Structural Recursion	103
4.6.2	Proof by Structural Induction	107
4.6.3	Defining Functions by Structural Recursion on their Domains.	108
4.6.4	Condition for Defining a Function by Structural Recursion. .	109
4.6.5	When the Unique Decomposition Condition Fails?	112
4.7	Recursion and Induction on Well-Founded Sets	112
4.7.1	Well-Founded Sets.	112
4.7.2	The Principle of Proof by Well-Founded Induction	114
4.7.3	Definition of a Function by Well-Founded Recursion on its Domain	116
4.8	Recursive Programs	118
5.	Counting Things: Combinatorics	123
5.1	Two Basic Principles: Addition and Multiplication.	124
5.2	Using the Two Basic Principles Together	128
5.3	Four Ways of Selecting k Items out of n	128
5.4	Counting Formulae: Permutations and Combinations	133
5.4.1	The Formula for Permutations (O+R-)	134
5.4.2	The Formula for Combinations (O-R-)	136
5.5	Counting Formulae: Perms and Coms with Repetition.	140
5.5.1	The Formula for Permutations with Repetition Allowed (O+R+)	141
5.5.2	The Formula for Combinations with Repetition Allowed (O-R+)	142
5.6	Rearrangements and Partitions	144
5.6.1	Rearrangements.	144
5.6.2	Counting Partitions with a Given Numerical Configuration	146
6.	Weighing the Odds: Probability	153
6.1	Finite Probability Spaces	153
6.1.1	Basic Definitions	154
6.1.2	Properties of Probability Functions	155
6.2	Philosophy and Applications	158
6.3	Some Simple Problems	161
6.4	Conditional Probability	164
6.5	Interlude: Simpson's Paradox.	171
6.6	Independence	172
6.7	Bayes' Theorem	176

6.8	Random Variables and Expected Values	178
6.8.1	Random Variables	178
6.8.2	Expectation	179
6.8.3	Induced Probability Distributions	181
6.8.4	Expectation Expressed using Induced Probability Functions	183
7.	Squirrel Math: Trees	189
7.1	My First Tree	189
7.2	Rooted Trees	192
7.3	Labelled Trees	198
7.4	Interlude: Parenthesis-Free Notation	202
7.5	Binary Search Trees	204
7.6	Unrooted Trees	209
7.6.1	Definition of Unrooted Tree	210
7.6.2	Properties of Unrooted Trees	211
7.6.3	Finding Spanning Trees	214
8.	Yea and Nay: Propositional Logic	219
8.1	What is Logic?	220
8.2	Structural Features of Consequence	221
8.3	Truth-Functional Connectives	226
8.4	Tautologicality	230
8.4.1	The Language of Propositional Logic	230
8.4.2	Assignments and Valuations	231
8.4.3	Tautological Implication	231
8.4.4	Tautological Equivalence	234
8.4.5	Tautologies and Contradictions	237
8.5	Normal Forms, Least Letter-Sets, Greatest Modularity	240
8.5.1	Disjunctive Normal Form	240
8.5.2	Conjunctive Normal Form	242
8.5.3	Eliminating Redundant Letters	244
8.5.4	Most Modular Representation	245
8.6	Semantic Decomposition Trees	247
8.7	Natural Deduction	252
8.7.1	Enchainment	253
8.7.2	Second-Level (alias Indirect) Inference	255
9.	Something about Everything: Quantificational Logic	265
9.1	The Language of Quantifiers	265
9.1.1	Some Examples	266
9.1.2	Systematic Presentation of the Language	267
9.1.3	Freedom and Bondage	272
9.2	Some Basic Logical Equivalences	273
9.3	Semantics for Quantificational Logic	276
9.3.1	Interpretations	276

9.3.2	Valuating Terms under an Interpretation.	277
9.3.3	Valuating Formulae under an Interpretation: Basis	277
9.3.4	Valuating Formulae under an Interpretation: Recursion Step.	278
9.3.5	The x -Variant Reading of the Quantifiers.	278
9.3.6	The Substitutional Reading of the Quantifiers	281
9.4	Logical Consequence etc.	283
9.5	Natural Deduction with Quantifiers.	289
Index	297