Undergraduate Topics in Computer Science

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

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

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