Universitext

Guerino Mazzola
Gérard Milmeister
Jody Weissmann

Comprehensive Mathematics for Computer Scientists 1

Sets and Numbers, Graphs and Algebra, Logic and Machines, Linear Geometry

Springer

Guerino Mazzola • Gérard Milmeister •
Jody Weissmann

Comprehensive Mathematics for Computer Scientists 1

Sets and Numbers, Graphs and Algebra, Logic and Machines, Linear Geometry

With 82 Figures

Springer

Guerino Mazzola
Gérard Milmeister
Jody Weissmann

Department of Informatics
University of Zurich
Winterthurerstr. 190
8057 Zurich, Switzerland

The text has been created using LaTeX 2_{ε}. The graphics were drawn using Dia, an open-source diagramming software. The main text has been set in the Y&Y Lucida Bright type family, the headings in Bitstream Zapf Humanist 601.

Library of Congress Control Number: 2004102307

Mathematics Subject Classification (1998): 00A06

ISBN 3-540-20835-6 Springer-Verlag Berlin Heidelberg New York

Springer-Verlag
is a part of Springer Science+Business Media
springer.de

Cover Design: Erich Kirchner, Heidelberg
Typesetting: Camera ready by the authors
40/3142SR - 5 4 3 2 1 0 - Printed on acid-free paper

Universitext

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Preface

The need for better formal competence as it is generated by a sound mathematical education has been confirmed by recent investigations by professional associations, but also by IT opinion leaders such as Niklaus Wirth or Peter Wegner. It is rightly argued that programming skills are a necessary but by far not sufficient qualification for designing and controlling the conceptual architecture of valid software. Often, the deficiency in formal competence is compensated by trial and error programming. This strategy may lead to uncontrolled code which neither formally nor effectively meets the given objectives. According to the global view such bad engineering practice leads to massive quality breakdowns with corresponding economical consequences.

Improved formal competence is also urged by the object-oriented paradigm which progressively requires a programming style and a design strategy of high abstraction level in conceptual engineering. In this context, the arsenal of formal tools must apply to completely different problem situations. Moreover, the dynamics and life cycle of hard- and software projects enforce high flexibility of theorists and executives on all levels of the computer science academia and IT industry. This flexibility can only be guaranteed by a propaedeutical training in a number of typical styles of mathematical argumentation.

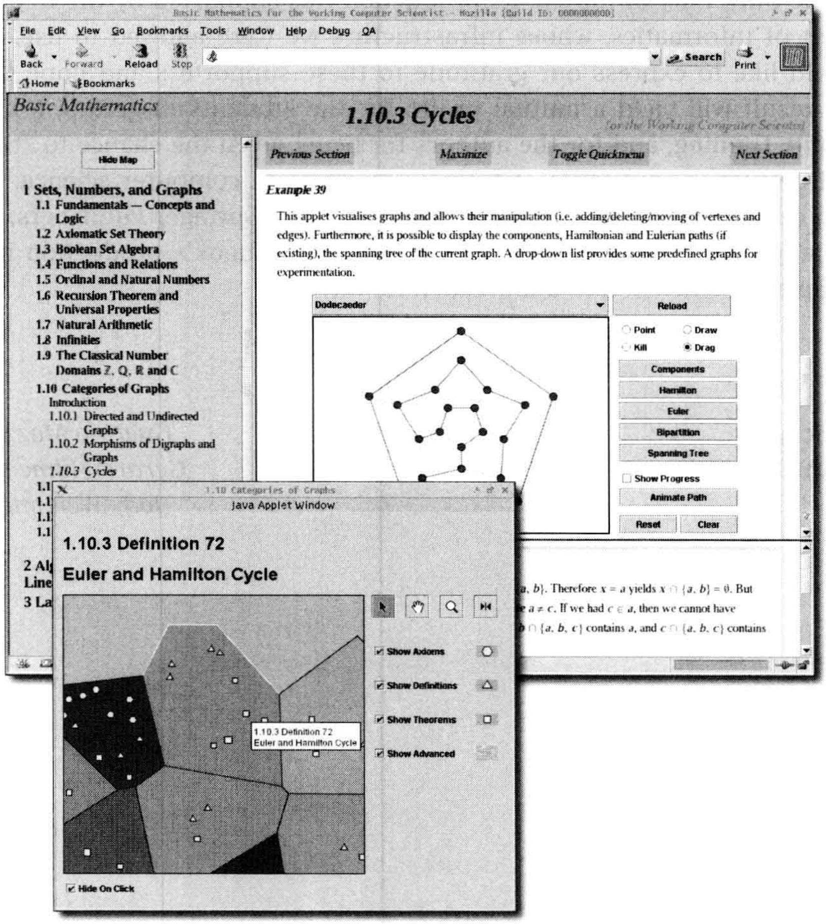
With this in mind, writing an introductory book on mathematics for computer scientists is a somewhat delicate task. On the one hand, computer science delves into the most basic machinery of human thought, such as it is traced in the theory of Turing machines, rewriting systems and grammars, languages, and formal logic. On the other hand, numerous applications of core mathematics, such as the theory of Galois fields (e.g., for coding theory), linear geometry (e.g., for computer graphics), or differential equations (e.g., for simulation of dynamic systems) arise in any relevant topic of computational science. In view of this wide field of mathematical subjects the common practice is to focus one's attention on a particular bundle of issues and to presuppose acquaintance with the background theory, or else to give a short summary thereof without any further details.

In this book, we have chosen a different presentation. The idea was to set forth and prove the entire core theory, from axiomatic set theory to numbers, graphs, algebraic and logical structures, linear geometry—in the present first volume, and then, in the second volume, topology and calculus, differential equations, and more specialized and current subjects such as neural networks, fractals, numerics, Fourier theory, wavelets, probability and statistics, manifolds, and categories.

There is a price to pay for this comprehensive journey through the overwhelmingly extended landscape of mathematics: We decided to omit any not absolutely necessary ramification in mathematical theorization. Rather it was essential to keep the global development in mind and to avoid an unnecessarily broad approach. We have therefore limited explicit proofs to a length which is reasonable for the non-mathematician. In the case of lengthy and more involved proofs, we refer to further readings. For a more profound reading we included a list of references to original publications. After all, the student should realize as early as possible in his or her career that science is vitally built upon a network of links to further knowledge resources.

We have, however, chosen a a modern presentation: We introduce the language of commutative diagrams, universal properties and intuitionistic logic as advanced by contemporary theoretical computer science in its topos-theoretic aspect. This presentation serves the economy and elegance of abstraction so urgently requested by opinion leaders in computer science. It also shows some original restatements of well-known facts, for example in the theory of graphs or automata. In addition, our presentation offers a glimpse of the unity of science: Machines, formal concept architectures, and mathematical structures are intimately related with each other.

Beyond a traditional "standalone" textbook, this text is part of a larger formal training project hosted by the Department of Informatics at the University of Zurich. The online counterpart of the text can be found on `http://math.ifi.unizh.ch`. It offers access to this material and includes interactive tools for examples and exercises implemented by Java applets and script-based dynamic HTML. Moreover, the online presentation allows switching between textual navigation via classical links and a quasi-geographical navigation on a "landscape of knowledge". In the latter, parts, chapters, axioms, definitions, and propositions are visualized by continents, nations, cities, and paths. This surface structure describes the top layer of a three-fold stratification (see the following screenshot of some windows of the online version).



1.10.3 Cycles

1 Sets, Numbers, and Graphs
1.1 Fundamentals — Concepts and Logic
1.2 Axiomatic Set Theory
1.3 Boolean Set Algebra
1.4 Functions and Relations
1.5 Ordinal and Natural Numbers
1.6 Recursion Theorem and Universal Properties
1.7 Natural Arithmetic
1.8 Infinities
1.9 The Classical Number Domains Z, Q, R and C
1.10 Categories of Graphs
Introduction
1.10.1 Directed and Undirected Graphs
1.10.2 Morphisms of Digraphs and Graphs
1.10.3 Cycles
1.1
1.1
1.1
1.1
2 Al
Line
3 La

Example 39

This applet visualises graphs and allows their manipulation (i.e. adding/deleting/moving of vertexes and edges). Furthermore, it is possible to display the components, Hamiltonian and Eulerian paths (if existing), the spanning tree of the current graph. A drop-down list provides some predefined graphs for experimentation.

1.10.3 Definition 72

Euler and Hamilton Cycle

$a, b\}$. Therefore $x = a$ yields $x \cap \{a, b\} = \emptyset$. But
$a \neq c$. If we had $c \in a$, then we cannot have
$b \cap \{a, b, c\}$ contains a, and $c \cap \{a, b, c\}$ contains

On top are the facts, below, in the middle layer, the user will find the proofs, and in the third, deepest stratum, one may access the advanced topics, such as floating point arithmetic, or coding theory. The online counterpart of the book includes two important addenda: First, a list of errata can be checked out. The reader is invited to submit any error encountered while reading the book or the online presentation. Second, the subject spectrum, be it in theory, examples, or exercises, is constantly updated and completed and, if appropriate, extended. It is therefore recommended and beneficial to work with both, the book and its online counterpart.

This book is a result of an educational project of the E-Learning Center of the University of Zurich. Its production was supported by the Department of Informatics, whose infrastructure we were allowed to use. We would like to express our gratitude to these supporters and hope that the result will yield a mutual profit: for the students in getting a high quality training, and for the authors for being given the chance to study and develop a core topic of formal education in computer science. We also deeply appreciate the cooperation with the Springer Publishers, especially with Clemens Heine, who managed the book's production in a completely efficient and unbureaucratic way.

Zurich,
February 2004

Guerino Mazzola
Gérard Milmeister
Jody Weissmann

Contents

Volume II

PART I

Sets, Numbers, and Graphs

CHAPTER 1

Fundamentals—Concepts and Logic

Die Welt ist alles, was der Fall ist.
Ludwig Wittgenstein

"The world is everything that is the case" — this is the first tractatus in Ludwig Wittgenstein's *Tractatus Logico-Philosophicus.*

In science, we want to know what is true, i.e., what is the case, and what is not. Propositions are the theorems of our language, they are to describe or denote what is the case. If they do, they are called true, otherwise they are called false. This sounds a bit clumsy, but actually it is pretty much what our common sense tells us about true and false statements. Perhaps an example would help to clarify things:

"This sentence contains five words" This proposition describes something which is the case, therefore it is a *true* statement.

"Every human being has three heads" Since I myself have only one head (and I assume this is the case with you as well), this proposition describes a situation which is not the case, therefore it is *false.*

In order to precisely handle propositions, science makes use of two fundamental tools of thought:

- Propositional Logic
- Architecture of Concepts

These tools aid a scientist to construct accurate concepts and to formulate new true propositions from old ones.

The following sections may appear quite diffuse to the reader; some things will seem to be obviously true, other things will perhaps not make much sense to start with. The problem is that we have to use our natural language for the task of defining things in a precise way. It is only by using these tools that we can define in a clear way what a set is, what numbers are, etc.

1.1 Propositional Logic

Propositional logic helps us to navigate in a world painted in black and white, a world in which there is only truth or falsehood, but nothing in between. It is a boiled down version of common sense reasoning. It is the essence of Sherlock Holmes' way of deducing that Professor Moriarty was the mastermind behind a criminal organization ("Elementary, my dear Watson"). Propositional logic builds on two propositions, which are declared to be true as basic principles (and they seem to make sense...):

Principle of contradiction (principium contradictionis) A proposition is never true and false at the same time.

Principle of the excluded third (tertium non datur) A proposition is either true or false—there is no third possibility.

In other words, in propositional logic we work with statements that are either true or false, no more and no less.

In propositional logic there are also some operations which are used to create new propositions from old ones:

Logical Negation The negation of a true proposition is a false proposition, the negation of a false proposition is a true proposition. This operation is also called 'NOT'.

Logical Conjunction The conjunction of two propositions is true if and only if both propositions are true. In all other cases it is false. This operation is also called 'AND'.

Logical Disjunction The disjunction of two propositions is true if at least one of the propositions is true. If both propositions are false, the disjunction is false, too. This operation is also known as 'OR'.