Macmillan Computer Science Series

Consulting Editor

Professor F. H. Sumner, University of Manchester

S. T. Allworth and R. N. Zobel, *Introduction to Real-time Software Design*, second edition

Ian O. Angell and Gareth Griffith, *High-resolution Graphics Using Fortran 77*

R. E. Berry and B. A. E. Meekings, *A Book on C*

G. M. Birtwistle, *Discrete Event Modelling on Simula*

T. B. Boffey, *Graph Theory in Operations Research*

Richard Bornat, *Understanding and Writing Compilers*

J. K. Buckle, *Software Configuration Management*

W. D. Burnham and A. R. Hall, *Prolog Programming and Applications*

J. C. Cluley, *Interfacing to Microprocessors*

Robert Cole, *Computer Communications*, second edition

Derek Coleman, *A Structured Programming Approach to Data*

Andrew J. T. Colin, *Fundamentals of Computer Science*

Andrew J. T. Colin, *Programming and Problem-solving in Algol 68*

S. M. Deen, *Fundamentals of Data Base Systems*

S. M. Deen, *Principles and Practice of Database Systems*

Tim Denvir, *Introduction to Discrete Mathematics for Software Engineering*

P. M. Dew and K. R. James, *Introduction to Numerical Computation in Pascal*

M. R. M. Dunsmuir and G. J. Davies, *Programming the UNIX System*

K. C. E. Gee, *Introduction to Local Area Computer Networks*

J. B. Gosling, *Design of Arithmetic Units for Digital Computers*

Roger Hutton, *Z80 Assembly Language Programming for Students*

Roland N. Ibbett, *The Architecture of High Performance Computers*

Patrick Jaulent, *The 68000—Hardware and Software*

J. M. King and J. P. Pardoe, *Program Design Using JSP—A Practical Introduction*

H. Kopetz, *Software Reliability*

E. V. Krishnamurthy, *Introductory Theory of Computer Science*

V. P. Lane, *Security of Computer Based Information Systems*

Graham Lee, *From Hardware to Software—an introduction to computers*

A. M. Lister, *Fundamentals of Operating Systems*, third edition

G. P. McKeown and V. J. Rayward-Smith, *Mathematics for Computing*

Brian Meek, *Fortran, PL/I and the Algols*

Barry Morrell and Peter Whittle, *CP/M 80 Programmer's Guide*

Derrick Morris, *System Programming Based on the PDP11*

Pim Oets, *MS-DOS and PC-DOS—A Practical Guide*

Christian Queinnec, *LISP*

(continued overleaf)

W. P. Salman, O. Tisserand and B. Toulout, *FORTH*
L. E. Scales, *Introduction to Non-linear Optimization*
Peter S. Sell, *Expert Systems—A Practical Introduction*
Colin J. Theaker and Graham R. Brookes, *A Practical Course on
Operating Systems*
J-M. Trio, *8086–8088 Architecture and Programming*
M. J. Usher, *Information Theory for Information Technologists*
B. S. Walker, *Understanding Microprocessors*
Peter J. L. Wallis, *Portable Programming*
Colin Walls, *Programming Dedicated Microprocessors*
I. R. Wilson and A. M. Addyman, *A Practical Introduction to Pascal—with
BS6192, second edition*

Introduction to Discrete Mathematics for Software Engineering

Tim Denvir

Praxis Systems plc

M
MACMILLAN

© Tim Denvir 1986

All rights reserved. No reproduction, copy or transmission of this publication may be made without written permission.

No paragraph of this publication may be reproduced, copied or transmitted save with written permission or in accordance with the provision of the Copyright Act 1956 (as amended).

Any person who does any unauthorised act in relation to this publication may be liable to criminal prosecution and civil claims for damages.

First published 1986

Published by
MACMILLAN EDUCATION LTD
Houndmills, Basingstoke, Hampshire RG21 2XS
and London
Companies and representatives
throughout the world

British Library Cataloguing in Publication Data
Denvir, Tim

Introduction to discrete mathematics for
software engineering.—(Macmillan computer
science series)

1. Electronic data processing—Mathematics
510 QA76.9.M35

ISBN 978-0-333-40737-0 ISBN 978-1-349-18496-5 (eBook)
DOI 10.1007/978-1-349-18496-5

Contents

<i>Foreword</i>	<i>vii</i>
<i>Preface</i>	<i>viii</i>
Chapter 1 Introduction	1
Part I: Foundations	13
Chapter 2 Sets	15
Chapter 3 Logic	28
Chapter 4 Sets and their Operations	55
Part II: Data Type Construction	93
Chapter 5 Tuples, Products and Sums	95
Chapter 6 Mappings and Functions	113
Chapter 7 Relations	147
Part III: Algebraic Concepts	157
Chapter 8 Algebras	159
Chapter 9 Homomorphisms	181
Chapter 10 Functions and their Specifications	214
Part IV: Specification and Correctness of Programs	229
Chapter 11 Relations and Lattices	231
Chapter 12 Specification and Correctness of Programs	252
<i>References</i>	<i>281</i>
<i>Answers to Exercises</i>	<i>283</i>
<i>Index</i>	<i>307</i>

*To Brenda, James and Eleanor;
to my colleagues at Praxis Systems plc;
and to the memories of Alan H. Day and A. Richard Tate*

Foreword

Notations from areas of discrete mathematics can be used to record the specifications of software systems; reasoning based on the properties of such notation can be used to show that designs satisfy such specifications. The sufficiency of discrete mathematics shows that, in some ways, it should be easier to verify software designs than to ensure safety in those engineering disciplines which have to cope with continuous variation, stress, decay, etc. In other ways, software design is made difficult by the inability to increase safety by adding redundancy.

The correctness of a software system must be established by mathematical proof. There is a growing acceptance that 'formal methods' are essential in achieving computer systems which are safe in use and economical in creation and maintenance costs.

There already exist a number of books which show how formal methods can be applied to software development. Most of these books assume a prior knowledge of discrete mathematics. There are, of course, many books which teach the relevant mathematical ideas, but these are not related to software problems.

Tim Denvir's book bridges the gap. This is a book with well-chosen examples which takes the reader from the elementary details of set and logic notation to their use in specification and proof. The book has evolved from industrial courses and should provide the sort of 'new-readers-start-here' material which will enable many more software engineers to begin to apply formal methods.

Marktoberdorf
August 1986

Cliff Jones

Preface

The title of this book has been chosen with some deliberation. The subject of the book is *Discrete Mathematics*. It does not aim to teach any techniques or methods of software engineering *per se*, but rather the mathematics which lies at their foundations. It is an *Introduction* only, because the elements of the subject are all that are required for this purpose. Finally, it is about Discrete Mathematics for the activity of *Software Engineering*, whether that activity is carried out by software engineers, accountants, civil engineers, astronomers, clerks or anyone else. Thus the title is intended to invite all who propose to 'engineer software', whatever their principal occupation.

This book has four primary aims: to teach to a depth adequate for its appreciation and practice the mathematical theory which underpins a scientific approach to software engineering; to impart sufficient skills in Discrete Mathematics to enable readers to practise software engineering in a scientific, disciplined manner; to convince readers of the desirability and applicability of Discrete Mathematics to software engineering; and to provide a cognitive stepping stone such that readers will be able to deal with more advanced works in computer science and related topics, as found in the ever-increasing range of scientific journals and books.

To support these aims, examples are drawn from software engineering which model the discrete mathematical concepts. These examples are thematically developed and elaborated as further concepts are introduced. To develop readers' skills, exercises are also presented. Some of these are formalistic and provide practice in the notational symbolism. Others apply the mathematical concepts to practical domains such as data processing or telecommunications. Some of the exercises are designated 'workshop'. These are intended for supervised group discussion and solution.

One of the workshop exercises concerns a control system for a nuclear power station. As I write this preface, the tragic explosion at Chernobyl has just occurred. This particular choice of workshop exercise may therefore be considered an unhappy one. While I believe that a scientific, mathematically based approach can only increase the safety of any piece of engineering, alone it cannot guarantee it; many other principles and practices need to be applied, and some risks will almost certainly remain.

This book is relevant for courses in discrete mathematics which form part of an undergraduate curriculum in computer science or software engineering. It can also be used to support courses for software engineers in industry. In fact, it was out of such an industrial course in STC plc that this book had its origins. Practising software engineers found that a grounding in discrete mathematics was of substantial advantage when

trying to apply techniques such as VDM (Jones, 1980, 1986), CCS (Milner, 1980) or Dijkstra's 'discipline of programming' (Dijkstra, 1976). This book thus owes a debt to over 120 students from the ranks of STC plc who attended the courses and helped the author and other lecturers to remedy its initial deficiencies.

Since the initial motivation provided by the course, the production of this book has struggled up a long and stony path. Over this time the support and encouragement of many colleagues has given my fragile enthusiasm and determination the impetus needed to continue the work. I am especially grateful to Paul Taylor, who, as a colleague at STL took part in giving the Discrete Mathematics course to STC engineers, and later when at Brighton Polytechnic read the manuscript and suggested many improvements to the quality of my proofs of theorems and other mathematical presentations. Several of the illustrative examples are his. I am also particularly grateful to Cliff Jones who gave me encouragement at a critical time and made helpful suggestions on my final draft.

A. McGettrick, J. Van Leuwen and a number of anonymous reviewers made substantial comments on the first two drafts, and Chris George, Roger Shaw and Mel Jackson supported me in giving the STC course which catalysed the work. Kathy Evans persevered with my handwriting in producing the whole of the first draft, and Standard Telecommunication Laboratories kindly allowed me the use of their facilities for preparing the various drafts.

Any serious attempt to communicate ideas always reveals to the communicator new insights into the subject matter. When I started writing this book I thought it would take less than a year, but it has in fact taken more than four. During that time my motives, while always being to communicate, have shifted their focus. I started by trying to communicate a skill, but now I hope I may communicate a further kind of understanding. Computer programs are built of abstractions at all levels. They are like poems whose language is pure thought, whose form is of science, and whose power, if controlled by an engineering discipline, can be put to extending ourselves and our environment or destroying them.

Mathematics is perhaps the most abstract form of thought the human mind can contrive. The pleasure I derive from mathematics has always disproportionately exceeded my skill in manipulating it. At the end of these four years I now hope that what I have produced will kindle in others, not merely a skill but more essentially, an understanding of the joy to be found in mathematics, and with that, a perception which will stir software engineers to direct their skills to the service of humanity and our environment and not to their degradation.

TIM DENVIR

London
April 1986