

Handbook of Logic in Computer Science

Volume 1

Background: Mathematical Structures

Edited by

S. ABRAMSKY

Professor of Computing Science

DOV M. GABBAY

Professor of Computing Science

and

T. S. E. MAIBAUM

*Professor of Foundations of
Software Engineering*

*Imperial College of Science, Technology and Medicine
London*

Volume Co-ordinator

DOV M. GABBAY

CLARENDON PRESS · OXFORD

1992

Oxford University Press, Walton Street, Oxford OX2 6DP

*Oxford New York Toronto
Delhi Bombay Calcutta Madras Karachi
Kuala Lumpur Singapore Hong Kong Tokyo
Nairobi Dar es Salaam Cape Town
Melbourne Auckland Madrid
and associated companies in
Berlin Ibadan*

Oxford is a trade mark of Oxford University Press

*Published in the United States
by Oxford University Press Inc., New York*

© Chapter authors, 1992

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, without the prior permission in writing of Oxford University Press. Within the UK, exceptions are allowed in respect of any fair dealing for the purpose of research or private study, or criticism or review, as permitted under the Copyright, Designs and Patents Act, 1988, 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 and in other countries should be sent to the Rights Department, Oxford University Press, at the address above.

This book is sold subject to the condition that it shall not, by way of trade or otherwise, be lent, re-sold, hired out, or otherwise circulated without the publisher's prior consent in any form of binding or cover other than that in which it is published and without a similar condition including this condition being imposed on the subsequent purchaser.

A catalogue record for this book is available from the British Library

*Library of Congress Cataloging in Publication Data
(Data available on request)*

ISBN 0-19-853735-2

*Printed in Great Britain by
Biddles Ltd,
Guildford and King's Lynn*

HANDBOOK OF LOGIC IN COMPUTER SCIENCE

Editors

S. Abramsky, Dov M. Gabbay, and T. S. E. Maibaum

HANDBOOKS OF LOGIC IN COMPUTER SCIENCE
and
ARTIFICIAL INTELLIGENCE AND LOGIC PROGRAMMING

Executive Editor

Dov M. Gabbay

Administrator

Jane Spurr

Handbook of Logic in Computer Science

- | | |
|-----------------|---|
| Volume 1 | Background: Mathematical structures |
| Volume 2 | Background: Computational structures |
| Volume 3 | Semantic structures |
| Volume 4 | Semantic modelling |
| Volume 5 | Theoretical methods in specification and verification |
| Volume 6 | Logical methods in computer science |

Handbook of Logic in Artificial Intelligence and
Logic Programming

- | | |
|-----------------|--|
| Volume 1 | Logical foundations |
| Volume 2 | Deduction methodologies |
| Volume 3 | Nonmonotonic reasoning and uncertain reasoning |
| Volume 4 | Epistemic and temporal reasoning |
| Volume 5 | Logic programming |

Preface

We are happy to present the first volumes of the *Handbook of Logic in Computer Science*. Logic is now widely recognized to be one of the foundational disciplines of computing and has found applications in virtually all aspects of the subject, from software engineering and hardware to programming language and artificial intelligence. There is a growing need for an in-depth survey of the application of logic in computer science and AI. The *Handbook of Logic in Computer Science* and its companion, the *Handbook of Logic in Artificial Intelligence and Logic Programming* have been created in response to this need.

We see the creation of the Handbook as a combination of authoritative exposition, comprehensive survey, and fundamental research exploring the underlying unifying themes in the various areas. The intended audience is graduate students and researchers in the areas of computing and logic, as well as other people interested in the subject. We assume as background some mathematical sophistication. Much of the material will also be of interest to logicians and mathematicians.

The tables of contents of the volumes were finalized after extensive discussions between handbook authors and second readers. The first two volumes present the background logic and mathematics extensively used in computer science. The point of view is application oriented. The other four volumes present major areas in which the methods are used. These include Volume 3 — Semantic Structures; Volume 4 — Semantic Modelling; Volume 5 — Specification and Verification; and Volume 6 — Logical Methods.

The chapters, which in many cases are of monographic length and scope, are written with emphasis on possible unifying themes. The chapters have an overview, introduction, and a main body. A final part is dedicated to more specialized topics.

Chapters are written by internationally renowned researchers in the respective areas. The chapters are co-ordinated and their contents were discussed in joint meetings.

Each chapter has been written using the following procedures:

1. A very detailed table of contents was discussed and co-ordinated at several meetings between authors and editors of related chapters. The discussion was in the form of a series of lectures by the authors to everyone present. Once an agreement was reached on the detailed table of contents the authors wrote a draft and sent it to the editors and to other related authors. For each chapter there is a second reader (the first reader is the author) whose job it has been to scrutinize the

chapter together with the editors. The second reader's role is very important and has required effort and serious involvement with the authors.

Second readers for this volume are:

Chapter 1: Valuation Systems and Consequence Relations — M. Fourman

Chapter 2: Recursion Theory — M. Hennessy and J. Tucker

Chapter 3: Universal Algebra — A. Poigne and M. Fourman

Chapter 4: Category Theory — E. Wagner and M. Fourman

Chapter 5: Topology — H. Barendregt

Chapter 6: Model Theory — I. Hodkinson and D. Gabbay.

2. Once this process was completed (i.e. drafts seen and read by a large enough group of authors), there were other meetings on several chapters in which authors lectured on their chapters and faced the criticism of the editors and audience. The final drafts were prepared after these meetings.
3. We attached great importance to group effort and co-ordination in the writing of chapters. The first two parts of each chapter, namely the Introduction–Overview and Main Body, are not completely under the discretion of the author, as he/she had to face the general criticism of all the other authors. Only the third part of the chapter is entirely for the authors' own tastes and preferences.

The Handbook meetings were generously financed by OUP, by SERC contract SO/809/86, by the Department of Computing at Imperial College, and by several anonymous private donations.

We would like to thank our colleagues, authors, second readers, and students for their effort and professionalism in producing the manuscripts for the Handbook. We would particularly like to thank the staff of OUP for their continued and enthusiastic support, and Mrs Jane Spurr, our OUP Administrator for her dedication and efficiency.

London
June 1992

The Editors

Contents

Valuation systems and consequence relations	1
1 Introduction	2
1.1 Logics and computer science	2
1.2 Summary	8
2 Valuation systems	9
2.1 Satisfaction	9
2.2 Valuation systems	13
2.3 Modal logic and possible worlds	21
2.4 Predicate languages	25
2.5 Summary	30
3 Consequence relations and entailment relations	30
3.1 Consequence relations	31
3.2 Entailment relations	34
3.3 The systems C and $S4$	35
3.4 Levels of implication	38
3.5 Consequence operator	39
3.6 Summary	40
4 Proof theory and presentations	40
4.1 Hilbert presentations	41
4.2 Natural deduction presentations	45
4.3 Natural deduction in sequent style	51
4.4 Intuitionistic logic	56
4.5 Gentzen sequent calculus for I	57
4.6 Gentzen sequent calculus for C and $S4$	62
4.7 Properties of presentations	64
5 Some further topics	66
5.1 Valuation systems for I	67
5.2 Maps between logics	69
5.3 Correspondence theory	71
5.4 Consistency	74
Recursion theory	79
0 Introduction	80
0.1 Opening remarks	80
0.2 A taster	81
0.3 Contents of the chapter	84

1	Languages and notions of computability	84
1.1	Data types and coding	85
1.2	The imperative paradigm	87
1.3	The functional paradigm	99
1.4	Recursive functions	109
1.5	Universality	113
2	Computability and non-computability	124
2.1	Non-computability	124
2.2	Computability	127
2.3	Recursive and recursively enumerable sets	129
2.4	The S - m - n theorem and partial evaluation	133
2.5	More undecidable problems	135
2.6	Problem reduction and r.e. completeness	136
3	Inductive definitions	139
3.1	Operators and fixed points	140
3.2	The denotational semantics of the functional language FL	149
3.3	Ordinals	156
3.4	The general case	169
4	Recursion theory	172
4.1	Fixed point theorems	173
4.2	Acceptable programming systems	176
4.3	Recursive operators	181
4.4	Inductive definitions and logics	183
	Universal algebra	189
1	Introduction	190
1.1	What is universal algebra?	190
1.2	Universal algebra in mathematics and computer science	191
1.3	Overview of the chapter	192
1.4	Historical notes	192
1.5	Acknowledgements	195
1.6	Prerequisites	196
2	Examples of algebras	196
2.1	Some basic algebras	197
2.2	Some simple constructions	207
2.3	Syntax and semantics of programs	210
2.4	Synchronous concurrent algorithms	215
2.5	Algebras and the modularisation of software	220
3	Algebras and morphisms	220
3.1	Signatures and algebras	220
3.2	Subalgebras	234

3.3	Congruences and quotient algebras	244
3.4	Homomorphisms and isomorphisms	260
3.5	Direct products	275
3.6	Abstract data types	287
4	Constructions	287
4.1	Subdirect products, residual and local properties	288
4.2	Direct and inverse limits	298
4.3	Reduced products and ultraproducts	321
4.4	Local and residual properties and approximation	332
4.5	Remarks on references	336
5	Classes of algebras	337
5.1	Free, initial and final algebras	338
5.2	Equational logic	351
5.3	Equational Horn logic	369
5.4	Specification of abstract data types	392
5.5	Remarks on references	396
6	Further reading	397
6.1	Universal algebra	398
6.2	Model theory	398

Basic category theory 413

1	Categories, functors and natural transformations	416
1.1	Types, composition and identities	416
1.2	Categories	418
1.3	Relating functional calculus and category theory	424
1.4	Compositionality is functorial	427
1.5	Natural transformations	433
2	On universal definitions: products, disjoint sums and higher types	437
2.1	Product types	437
2.2	Coproducts	443
2.3	Higher types	446
2.4	Reasoning by universal arguments	451
2.5	Another ‘universal definition’: primitive recursion	456
2.6	The categorical abstract machine	457
3	Universal problems and universal solutions	460
3.1	On observation and abstraction	461
3.2	A more categorical point of view	465
3.3	Universal morphisms	470
3.4	Adjunction	473
3.5	On generation	475
3.6	More examples for separation and generation	481

4	Elements and beyond	492
4.1	Variable elements, variable subsets and representable functors	492
4.2	Yoneda's heritage	498
4.3	Towards an enriched category theory	500
5	Data structures	511
5.1	Subtypes	511
5.2	Limits	516
5.3	Colimits	526
6	Universal constructions	535
6.1	The adjoint functor theorem	536
6.2	Generation as partial evaluation	541
6.3	Left Kan extensions, tensors and coends	545
6.4	Separation by testing	551
6.5	On bimodules and density	553
7	Axiomatizing programming languages	559
7.1	Relating theories of λ -calculus	559
7.2	Type equations and recursion	566
7.3	Solving recursive equations	574
8	Algebra categorically	578
8.1	Functorial semantics	578
8.2	Enriched functorial semantics	588
8.3	Monads	593
9	On the categorical interpretation of calculi	603
9.1	Category theory as proof theory	603
9.2	Substitution as predicate transformation	605
9.3	Theories of equality	612
9.4	Type theories	619
10	A sort of conclusion	632
11	Literature	633
11.1	Textbooks	633
11.2	References	634
	Topology	641
1	Observable properties	642
2	Examples of topological spaces	646
2.1	Sierpinski space	646
2.2	Scott Topology	646
2.3	Spaces of maximal elements. Cantor space	647
2.4	Alexandroff topology	647
2.5	Stone spaces	648
2.6	Spectral spaces	649
2.7	The reals	650

3	Alternative formulations of topology	653
3.1	Closed sets	653
3.2	Neighbourhoods	654
3.3	Examples	655
3.4	Closure operators	656
3.5	Convergence	660
4	Separation, continuity and sobriety	663
4.1	Separation conditions	663
4.2	Continuous functions	665
4.3	Predicate transformers and sobriety	668
4.4	Many-valued functions	675
5	Constructions: new spaces from old	681
5.1	Postscript: effectiveness and representation	690
6	Metric Spaces	698
6.1	Basic definitions	699
6.2	Examples	702
6.3	Completeness	706
6.4	Topology and metric	713
6.5	Constructions	719
6.6	A note on uniformities	724
7	Compactness	727
7.1	Compactness and finiteness	727
7.2	Spectral spaces	730
7.3	Positive and negative information: patch topology	733
7.4	Hyperspaces	736
7.5	Tychonoff's theorem	738
7.6	Locally compact spaces	739
7.7	Function spaces	744
8	Appendix	751

Model theory and computer science:

	An appetizer	763
1	Introduction	764
2	The set theoretic modelling of syntax and semantics	766
2.1	First order structures	767
2.2	The choice of the vocabulary	768
2.3	Logics	769
3	Model theory and computer science	770
3.1	Computer science	770
3.2	The birth of model theory	771
3.3	Definability questions	772
3.4	Preservation theorems	773
3.5	Disappointing ultraproducts	774

3.6	Complete theories and elimination of quantifiers	774
3.7	Spectrum problems	775
3.8	Beyond first order logic	777
3.9	The hidden method	778
3.10	0–1 Laws	779
4	Preservation theorems	780
4.1	Horn formulas	780
5	Fast growing functions	782
5.1	Non-provability results in second order arithmetic	782
5.2	Non-provability in complexity theory	782
5.3	Model theory of fast growing functions	783
6	Elimination of quantifiers	784
6.1	Computer aided theorem proving in classical mathematics	785
6.2	Tarski's theorem	787
6.3	Elementary geometry	789
6.4	Other theories with elimination of quantifiers	790
7	Computable logics over finite structures	790
7.1	Computable logics	790
7.2	Computable quantifiers	793
7.3	Computable predicate transformers	794
7.4	L -Reducibility	796
7.5	Logics capturing complexity classes	797
8	Ehrenfeucht–Fraïssé games	798
8.1	The games	798
8.2	Completeness of the game	801
8.3	Second order logic and its sublogics	802
8.4	More non-definability results	802
8.5	The games and pumping lemmas	805
9	Conclusions	805
Author index		815
Index		817

Valuation Systems and Consequence Relations

Mark Ryan and Martin Sadler

Contents

1	Introduction	2
1.1	Logics and computer science	2
1.2	Summary	8
2	Valuation systems	9
2.1	Satisfaction	9
2.2	Valuation systems	13
2.3	Modal logic and possible worlds	21
2.4	Predicate languages	25
2.5	Summary	30
3	Consequence relations and entailment relations	30
3.1	Consequence relations	31
3.2	Entailment relations	34
3.3	The systems C and $S4$	35
3.4	Levels of implication	38
3.5	Consequence operator	39
3.6	Summary	40
4	Proof theory and presentations	40
4.1	Hilbert presentations	41
4.2	Natural deduction presentations	45
4.3	Natural deduction in sequent style	51
4.4	Intuitionistic logic	56
4.5	Gentzen sequent calculus for I	57
4.6	Gentzen sequent calculus for C and $S4$	62
4.7	Properties of presentations	64
5	Some further topics	66
5.1	Valuation systems for I	67
5.2	Maps between logics	69
5.3	Correspondence theory	71
5.4	Consistency	74

1 Introduction

This chapter is an introduction to some of the basic concepts and machinery of logic, concentrating on the contemporary uses of logic in computer science. It is not comprehensive, since many topics (such as equational reasoning, algebras, categories and computability) are introduced in other chapters. In particular, we concentrate on the applications of logics as reasoning systems about computing rather than on the foundational aspects of computer science which logic addresses. This is not a standard introduction to classical first order logic and its model theory, such treatments being widely available elsewhere [Hamilton, 1978; Hodges, 1983; Makowsky,]. Rather, we attempt to bring together some topics from the logics currently used for reasoning about programs and about computer systems, and see them within a unifying framework at an appropriate level of abstraction.

Prerequisites. Most of the constructions in this chapter are based on ‘naïve set theory’, whose use we take for granted. We also assume some exposure to the use and manipulation of the usual propositional operators (\wedge , \rightarrow , \vee , \neg), and to predicates. Some familiarity with discrete mathematics, including the use of λ -notation, and the idea of proofs both *within* and *about* a formal system would be helpful.

1.1 Logics and computer science

It has been recognized for a long time that there is an intimate connection between logic and computer science:

‘It is reasonable to hope that the relationship between computation and mathematical logic will be as fruitful in the next century as that between analysis and physics in the last. The development of this relationship demands a concern for both applications and mathematical elegance.’ (J. McCarthy, 1965)

We begin this chapter by describing some aspects of the interconnections between logic and computer science. We will introduce the machinery of logic later, here we rely on the reader’s intuitions about *truth*, *proofs* and *propositions*.

First we can distinguish between a ‘foundational’ aspect and a ‘reasoning system’ aspect of the connection. At the foundational aspect, logic is used to provide a model of the phenomenon of computation. For example: the *Curry–Howard isomorphism* is a description of the parallel between computation and proof transformation in intuitionistic logic. An idealized view of computation is adopted: it is taken to mean reducing typed lambda

terms to normal form. We can justify this idealization by appealing to the formal power of the typed λ -calculus and of Turing machines; see for example [Barendregt,] and [Phillips,] in this volume.

The key idea is the identification of *propositions* with *types*, and *proofs* with *terms*. A proposition is a type whose terms are the proofs of the proposition. This programme is called *intuitionistic type theory*, and started with P. Martin-Löf [Martin-Löf, 1975]. Reducing lambda terms to normal form (which is our model of computation) is the same as eliminating redundant steps in proofs. We return to this topic briefly later on.

Girard's *Linear logic* is a further step in this foundational direction. It abandons certain structural properties present in both classical and intuitionistic logic and by doing so provides a new approach to many issues in functional programming, such as lazy evaluation, side-effects, memory allocation and parallelism.

This is not, however, the level of interconnections between logic and computer science with which we are primarily concerned in this chapter; the interested reader should see other chapters in this Handbook. Instead, we concentrate on *logics as reasoning systems*—on their claim to be systems for making deductions from premises. Logic is used for describing and implementing systems which reason about a particular domain (e.g. in *specification theory* and *artificial intelligence*). Temporal logic is used to reason about domains in which time plays a key role, deontic logic for domains involving normative behaviour, default logic when only partial information is available, and so on. Specification theory provides many examples of this relationship between logic and computer science. The expressions in specification languages usually denote logical theories, often based on *multi-modal logics* (see [Stirling,] in this volume). Multi-modal logics have also been used to reason about concurrent systems and non-deterministic systems.

Already in this chapter we have referred to several different 'logics': intuitionistic logic, temporal logic, linear logic, deontic logic and others. There is a recurrent debate between *absolutism* and *pluralism*, whether there is 'one true logic' for reasoning about all domains or whether different logics should be studied for different domains. We attempt in this chapter to argue that computer science demands a pluralist view.

It is often stated that classical logic (the logic which is usually taken to be the 'one true logic' by absolutists) is capable of expressing arguments phrased in any 'non-standard' logic, and so removes the need for studying them. Indeed, we shall see in the last section that many logics can be embedded in classical logic in a precise sense. But to argue along these lines misses an important point. As V. Pratt says,

'There is a tradition in logic, carried over into computer science, to think of pure first order logic as a universal language. In fact