

Cambridge University Press

978-0-521-54310-1 - Logic in Computer Science: Modelling and Reasoning about Systems

Michael Huth and Mark Ryan

Frontmatter

[More information](#)

LOGIC IN COMPUTER SCIENCE

Modelling and Reasoning about Systems

Cambridge University Press
978-0-521-54310-1 - Logic in Computer Science: Modelling and Reasoning about Systems
Michael Huth and Mark Ryan
Frontmatter
[More information](#)

Cambridge University Press

978-0-521-54310-1 - Logic in Computer Science: Modelling and Reasoning about Systems

Michael Huth and Mark Ryan

Frontmatter

[More information](#)

LOGIC IN COMPUTER SCIENCE

Modelling and Reasoning about Systems

MICHAEL HUTH

*Department of Computing
Imperial College London, United Kingdom*

MARK RYAN

*School of Computer Science
University of Birmingham, United Kingdom*



CAMBRIDGE
UNIVERSITY PRESS

Cambridge University Press
978-0-521-54310-1 - Logic in Computer Science: Modelling and Reasoning about Systems
Michael Huth and Mark Ryan
Frontmatter
[More information](#)

CAMBRIDGE
UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

Published in the United States of America by Cambridge University Press, New York

Cambridge University Press is part of the University of Cambridge.

It furthers the University’s mission by disseminating knowledge in the pursuit of education, learning and research at the highest international levels of excellence.

www.cambridge.org

Information on this title: www.cambridge.org/9780521543101

© Cambridge University Press 2004

This publication is in copyright. Subject to statutory exception and to the provisions of relevant collective licensing agreements, no reproduction of any part may take place without the written permission of Cambridge University Press.

First published 2004
13th printing 2015

Printed in the United States of America by Sheridan Books, Inc.

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

ISBN 978-0-521-54310-1 Paperback

Cambridge University Press has no responsibility for the persistence or accuracy of URLs for external or third-party internet websites referred to in this publication, and does not guarantee that any content on such websites is, or will remain, accurate or appropriate. Information regarding prices, travel timetables and other factual information given in this work are correct at the time of first printing but Cambridge University Press does not guarantee the accuracy of such information thereafter.

Contents

<i>Foreword to the first edition</i>	<i>page</i> ix
<i>Preface to the second edition</i>	xi
<i>Acknowledgements</i>	xiii
1 Propositional logic	1
1.1 Declarative sentences	2
1.2 Natural deduction	5
1.2.1 Rules for natural deduction	6
1.2.2 Derived rules	23
1.2.3 Natural deduction in summary	26
1.2.4 Provable equivalence	29
1.2.5 An aside: proof by contradiction	29
1.3 Propositional logic as a formal language	31
1.4 Semantics of propositional logic	36
1.4.1 The meaning of logical connectives	36
1.4.2 Mathematical induction	40
1.4.3 Soundness of propositional logic	45
1.4.4 Completeness of propositional logic	49
1.5 Normal forms	53
1.5.1 Semantic equivalence, satisfiability and validity	54
1.5.2 Conjunctive normal forms and validity	58
1.5.3 Horn clauses and satisfiability	65
1.6 SAT solvers	68
1.6.1 A linear solver	69
1.6.2 A cubic solver	72
1.7 Exercises	78
1.8 Bibliographic notes	91
2 Predicate logic	93
2.1 The need for a richer language	93

vi	Contents	
2.2	Predicate logic as a formal language	98
2.2.1	Terms	99
2.2.2	Formulas	100
2.2.3	Free and bound variables	102
2.2.4	Substitution	104
2.3	Proof theory of predicate logic	107
2.3.1	Natural deduction rules	107
2.3.2	Quantifier equivalences	117
2.4	Semantics of predicate logic	122
2.4.1	Models	123
2.4.2	Semantic entailment	129
2.4.3	The semantics of equality	130
2.5	Undecidability of predicate logic	131
2.6	Expressiveness of predicate logic	136
2.6.1	Existential second-order logic	139
2.6.2	Universal second-order logic	140
2.7	Micromodels of software	141
2.7.1	State machines	142
2.7.2	Alma – re-visited	146
2.7.3	A software micromodel	148
2.8	Exercises	157
2.9	Bibliographic notes	170
3	Verification by model checking	172
3.1	Motivation for verification	172
3.2	Linear-time temporal logic	175
3.2.1	Syntax of LTL	175
3.2.2	Semantics of LTL	178
3.2.3	Practical patterns of specifications	183
3.2.4	Important equivalences between LTL formulas	184
3.2.5	Adequate sets of connectives for LTL	186
3.3	Model checking: systems, tools, properties	187
3.3.1	Example: mutual exclusion	187
3.3.2	The NuSMV model checker	191
3.3.3	Running NuSMV	194
3.3.4	Mutual exclusion revisited	195
3.3.5	The ferryman	199
3.3.6	The alternating bit protocol	203
3.4	Branching-time logic	207
3.4.1	Syntax of CTL	208

Contents	vii
3.4.2 Semantics of CTL	211
3.4.3 Practical patterns of specifications	215
3.4.4 Important equivalences between CTL formulas	215
3.4.5 Adequate sets of CTL connectives	216
3.5 CTL* and the expressive powers of LTL and CTL	217
3.5.1 Boolean combinations of temporal formulas in CTL	220
3.5.2 Past operators in LTL	221
3.6 Model-checking algorithms	221
3.6.1 The CTL model-checking algorithm	222
3.6.2 CTL model checking with fairness	230
3.6.3 The LTL model-checking algorithm	232
3.7 The fixed-point characterisation of CTL	238
3.7.1 Monotone functions	240
3.7.2 The correctness of SAT _{EG}	242
3.7.3 The correctness of SAT _{EU}	243
3.8 Exercises	245
3.9 Bibliographic notes	254
4 Program verification	256
4.1 Why should we specify and verify code?	257
4.2 A framework for software verification	258
4.2.1 A core programming language	259
4.2.2 Hoare triples	262
4.2.3 Partial and total correctness	265
4.2.4 Program variables and logical variables	268
4.3 Proof calculus for partial correctness	269
4.3.1 Proof rules	269
4.3.2 Proof tableaux	273
4.3.3 A case study: minimal-sum section	287
4.4 Proof calculus for total correctness	292
4.5 Programming by contract	296
4.6 Exercises	299
4.7 Bibliographic notes	304
5 Modal logics and agents	306
5.1 Modes of truth	306
5.2 Basic modal logic	307
5.2.1 Syntax	307
5.2.2 Semantics	308
5.3 Logic engineering	316
5.3.1 The stock of valid formulas	317

viii	Contents	
	5.3.2 Important properties of the accessibility relation	320
	5.3.3 Correspondence theory	322
	5.3.4 Some modal logics	326
5.4	Natural deduction	328
5.5	Reasoning about knowledge in a multi-agent system	331
	5.5.1 Some examples	332
	5.5.2 The modal logic $KT45^n$	335
	5.5.3 Natural deduction for $KT45^n$	339
	5.5.4 Formalising the examples	342
5.6	Exercises	350
5.7	Bibliographic notes	356
6	Binary decision diagrams	358
6.1	Representing boolean functions	358
	6.1.1 Propositional formulas and truth tables	359
	6.1.2 Binary decision diagrams	361
	6.1.3 Ordered BDDs	366
6.2	Algorithms for reduced OBDDs	372
	6.2.1 The algorithm reduce	372
	6.2.2 The algorithm apply	373
	6.2.3 The algorithm restrict	377
	6.2.4 The algorithm exists	377
	6.2.5 Assessment of OBDDs	380
6.3	Symbolic model checking	382
	6.3.1 Representing subsets of the set of states	383
	6.3.2 Representing the transition relation	385
	6.3.3 Implementing the functions pre_\exists and pre_\forall	387
	6.3.4 Synthesising OBDDs	387
6.4	A relational mu-calculus	390
	6.4.1 Syntax and semantics	390
	6.4.2 Coding CTL models and specifications	393
6.5	Exercises	398
6.6	Bibliographic notes	413
	<i>Bibliography</i>	414
	<i>Index</i>	418

Foreword to the first edition

by

Edmund M. Clarke
FORE Systems Professor of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Formal methods have finally come of age! Specification languages, theorem provers, and model checkers are beginning to be used routinely in industry. Mathematical logic is basic to all of these techniques. Until now textbooks on logic for computer scientists have not kept pace with the development of tools for hardware and software specification and verification. For example, in spite of the success of model checking in verifying sequential circuit designs and communication protocols, until now I did not know of a single text, suitable for undergraduate and beginning graduate students, that attempts to explain how this technique works. As a result, this material is rarely taught to computer scientists and electrical engineers who will need to use it as part of their jobs in the near future. Instead, engineers avoid using formal methods in situations where the methods would be of genuine benefit or complain that the concepts and notation used by the tools are complicated and unnatural. This is unfortunate since the underlying mathematics is generally quite simple, certainly no more difficult than the concepts from mathematical analysis that every calculus student is expected to learn.

Logic in Computer Science by Huth and Ryan is an exceptional book. I was amazed when I looked through it for the first time. In addition to propositional and predicate logic, it has a particularly thorough treatment of temporal logic and model checking. In fact, the book is quite remarkable in how much of this material it is able to cover: linear and branching time temporal logic, explicit state model checking, fairness, the basic fixpoint

theorems for computation tree logic (CTL), even binary decision diagrams and symbolic model checking. Moreover, this material is presented at a level that is accessible to undergraduate and beginning graduate students. Numerous problems and examples are provided to help students master the material in the book. Since both Huth and Ryan are active researchers in logics of programs and program verification, they write with considerable authority.

In summary, the material in this book is up-to-date, practical, and elegantly presented. The book is a wonderful example of what a modern text on logic for computer science should be like. I recommend it to the reader with greatest enthusiasm and predict that the book will be an enormous success.

(This foreword is re-printed in the second edition with its author's permission.)

Preface to the second edition

Our motivation for (re)writing this book

One of the leitmotifs of writing the first edition of our book was the observation that most logics used in the design, specification and verification of computer systems fundamentally deal with a *satisfaction relation*

$$\mathcal{M} \models \phi$$

where \mathcal{M} is some sort of *situation* or *model* of a system, and ϕ is a specification, a formula of that logic, expressing what should be true in situation \mathcal{M} . At the heart of this set-up is that one can often specify and implement algorithms for computing \models . We developed this theme for propositional, first-order, temporal, modal, and program logics. Based on the encouraging feedback received from five continents we are pleased to hereby present the second edition of this text which means to preserve and improve on the original intent of the first edition.

What’s new and what’s gone

Chapter 1 now discusses the design, correctness, and complexity of a SAT solver (a marking algorithm similar to Stålmarck’s method [SS90]) for full propositional logic.

Chapter 2 now contains basic results from model theory (Compactness Theorem and Löwenheim–Skolem Theorem); a section on the transitive closure and the expressiveness of existential and universal second-order logic; and a section on the use of the object modelling language Alloy and its analyser for specifying and exploring under-specified first-order logic models with respect to properties written in first-order logic with transitive closure. The Alloy language is executable which makes such exploration interactive and formal.

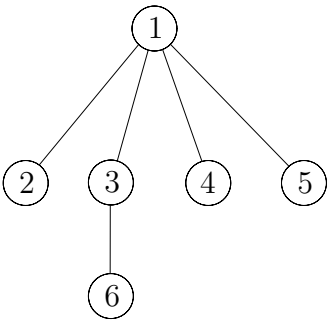
Chapter 3 has been completely restructured. It now begins with a discussion of linear-time temporal logic; features the open-source NuSMV model-checking tool throughout; and includes a discussion on planning problems, more material on the expressiveness of temporal logics, and new modelling examples.

Chapter 4 contains more material on total correctness proofs and a new section on the programming-by-contract paradigm of verifying program correctness.

Chapters 5 and 6 have also been revised, with many small alterations and corrections.

The interdependence of chapters and prerequisites

The book requires that students know the basics of elementary arithmetic and naive set theoretic concepts and notation. The core material of Chapter 1 (everything except Sections 1.4.3 to 1.6.2) is essential for all of the chapters that follow. Other than that, only Chapter 6 depends on Chapter 3 and a basic understanding of the static scoping rules covered in Chapter 2 – although one may easily cover Sections 6.1 and 6.2 without having done Chapter 3 at all. Roughly, the interdependence diagram of chapters is



WWW page

This book is supported by a Web page, which contains a list of errata; text files for all the program code; ancillary technical material and links; all the figures; an interactive tutor based on multiple-choice questions; and details of how instructors can obtain the solutions to exercises in this book which are marked with a *. The URL for the book’s page is www.cs.bham.ac.uk/research/lics/. See also www.cambridge.org/052154310x

Acknowledgements

Many people have, directly or indirectly, assisted us in writing this book. David Schmidt kindly provided several exercises for Chapter 4. Krysia Broda has pointed out some typographical errors and she and the other authors of [BEKV94] have allowed us to use some exercises from that book. We have also borrowed exercises or examples from [Hod77] and [FHMV95]. Susan Eisenbach provided a first description of the Package Dependency System that we model in Alloy in Chapter 2. Daniel Jackson made very helpful comments on versions of that section. Zena Matilde Ariola, Josh Hodas, Jan Komorowski, Sergey Kotov, Scott A. Smolka and Steve Vickers have corresponded with us about this text; their comments are appreciated. Matt Dwyer and John Hatcliff made useful comments on drafts of Chapter 3. Kevin Lucas provided insightful comments on the content of Chapter 6, and notified us of numerous typographical errors in several drafts of the book. Achim Jung read several chapters and gave useful feedback.

Additionally, a number of people read and provided useful comments on several chapters, including Moti Ben-Ari, Graham Clark, Christian Haack, Anthony Hook, Roberto Segala, Alan Sexton and Allen Stoughton. Numerous students at Kansas State University and the University of Birmingham have given us feedback of various kinds, which has influenced our choice and presentation of the topics. We acknowledge Paul Taylor’s L^AT_EX package for proof boxes. About half a dozen anonymous referees made critical, but constructive, comments which helped to improve this text in various ways. In spite of these contributions, there may still be errors in the book, and we alone must take responsibility for those.

Added for second edition

Many people have helped improve this text by pointing out typos and making other useful comments after the publication date. Among them,

we mention Wolfgang Ahrendt, Natasha Alechina, Yasuhiro Ajiro, Torben Amtoft, Stephan Andrei, Miguel Carrillo Barajas, Bernhard Beckert, Jonathan Bowen, Jonathan Brown, Filip Bruman, James Caldwell, B. Chimbo, Ruchira Datta, Xiao Fan, Amy Felty, Valentin Goranko, Dimitar Guelev, A. Burak Gurdag, Ernst Moritz Hahn, Pascal Honoré, Rod Howell, Lei Jinjiang, Hirotugu Kakugawa, Kamran Kashef, Anders Krogh, Markus Krötzsch, Jagun Kwon, Ranko Lazic, David Makinson, Alexander Miczo, Aart Middeldorp, Juan S. Morales, Robert Morelli, Lena Morgenroth, Prakash Panangaden, Aileen Paraguya, Frank Pfenning, Shekhar Pradhan, Hanspeter Schneider, Marek Sergot, Christian Sternagel, Koichi Takahashi, Petur Thors, Kazunori Ueda, Hiroshi Watanabe, Fuzhi Wang and Reinhard Wilhelm, Jin Yun, Harald Zankl.