

Texts in Theoretical Computer Science

An EATCS Series

Editors: W. Brauer G. Rozenberg A. Salomaa

On behalf of the European Association
for Theoretical Computer Science (EATCS)

Advisory Board: G. Ausiello M. Broy C. Calude
S. Even J. Hartmanis J. Hromkovič N. Jones T. Leighton
M. Nivat C. Papadimitriou D. Scott

Springer-Verlag Berlin Heidelberg GmbH

Klaus Schneider

Verification of Reactive Systems

Formal Methods and Algorithms

With 149 Figures



Springer

Author

Prof. Dr. Klaus Schneider
FB Informatik
AG Reaktive Systeme
Universität Kaiserslautern
67653 Kaiserslautern
Germany
klaus.schneider@informatik.uni-kl.de

Series Editors

Prof. Dr. Wilfried Brauer
Institut für Informatik
Technische Universität München
Arcisstrasse 21, 80333 München, Germany
brauer@informatik.tu-muenchen.de

Prof. Dr. Grzegorz Rozenberg
Leiden Institute of Advanced Computer Science
University of Leiden
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
rozenber@liacs.nl

Prof. Dr. Arto Salomaa
Data City
Turku Centre for Computer Science
20 500 Turku, Finland
asalomaa@utu.fi

Library of Congress Cataloging-in-Publication Data applied for

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche
Nationalbibliographie; detailed bibliographic data is available in
the Internet at <<http://dnd.dd.de>>

ACM Computing Classification (1998): F.3.1, D.2.4, F.4.1

ISBN 978-3-642-05555-3 ISBN 978-3-662-10778-2 (eBook)
DOI 10.1007/978-3-662-10778-2

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag Berlin Heidelberg GmbH. Violations are liable for prosecution under the German Copyright Law.

springeronline.com

© Springer-Verlag Berlin Heidelberg 2004
Originally published by Springer-Verlag Berlin Heidelberg New York in 2004
Softcover reprint of the hardcover 1st edition 2004

The use of general descriptive names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Cover Design: KünkelLopka, Heidelberg
Typesetting: Camera-ready by the author
45/3142 – 5 4 3 2 1 0 – Printed on acid-free paper

To Tim, Katja and Kai

Foreword

Computer systems are becoming ubiquitous. Many of the most important and prevalent ones are reactive systems. Reactive systems include microprocessors, computer operating systems, air traffic control systems, as well as on-board avionics and other embedded systems. These systems are characterized technically by their ongoing, ideally infinite behavior; termination is impossible or aberrant behavior, in contrast to classical theories of computation. Reactive systems tend to be characterized in practice by having failure modes that can severely compromise safety, even leading to loss of life. Alternatively, errors can have serious financial repercussions such as expensive recalls. Reactive systems need to be correct before being deployed.

To determine whether such reactive systems do behave correctly, a rich mathematical theory of verification of reactive systems has been developed over the last two decades or so. In contrast to earlier work emphasizing the role of proofs in deductive systems to establish correctness, the alternative suggestion is to take a model-theoretic view. It turns out that this permits the process of reasoning about program correctness to be fully automated in principle and partially automated to a high degree in practice.

It is my pleasure to introduce Klaus Schneider's excellent book *Verification of Reactive Systems: Formal Methods and Algorithms*. This book is the story of reactive systems verification, reflecting Klaus's broad expertise on the subject. It addresses both applications and theory, providing especially strong coverage of basic as well as advanced theory not otherwise available in book form. Key topics include Kripke and related transition structures, temporal logics, automata on infinite strings including Safra's determinization construction, expressiveness and Borel hierarchies of ω -languages, as well as monadic predicate logics. An underlying theme is the use of the vectored μ -calculus to provide an elegant "Theory of Everything". *Verification of Reactive Systems* belongs on the bookshelf of every serious researcher into the topic. It should also serve as a valuable text for graduate students and advanced undergraduates.

April 2003

E. Allen Emerson,
Endowed Professor of Computer Sciences
University of Texas at Austin

Preface

The design of modern information processing systems like digital circuits or protocols is becoming more and more difficult. A large part of the design costs and time (about 70%) is currently spent on methods that try to guarantee the absence of design errors. For this reason, designing systems is now more and more a synonym for verifying systems.

The research into the verification of reactive systems, in particular, into model checking, is one of the most impressive successes of theoretical computer science. Two decades after the publication of the basic papers on the formal foundation, the methods became mature enough for industrial usage. Nowadays, the hardware industry employs hundreds of highly specialized researchers working with formal methods to detect design bugs.

When I entered this field, it was an enormous effort to read hundreds of papers to understand the relationships between the different formal methods that are currently in use. It was surprising to me that there was no book covering all these methods, even the basic ones, although there is such a huge interest in them. For this reason, I decided to write this book to provide newcomers and researchers with a textbook that covers most of the relevant logics, with a particular emphasis on (verification and translation) algorithms.

The book is intended for graduate students as well as for researchers already working in this area. It is self-contained and gives proofs and algorithms for all important constructions. For a less detailed and formal introduction, I want to recommend the book of Clarke, Grumberg, and Peled [111]. Supplemental material on actual tools is found in [38], and further topics on the μ -calculus and infinite games are found in [221].

There are many persons I have to thank for helping me to write this book. In particular, I want to thank Detlef Schmid and the hardware verification group at the University of Karlsruhe, in particular Jorgos Logothetis, Tobias Schüle, and Roberto Ziller. Many discussions with Moshe Vardi moved me to improve the book. Allen Emerson was soon interested in the project and also gave fruitful comments. Moreover, I want to thank Amir Pnueli, Wolfgang Thomas, and Peter Schmitt for comments on early versions of the manuscript. Last, but not least, it should be mentioned that the editors of the EATCS series, in particular, Prof. Brauer, and the team at Springer-Verlag helped me to publish this book.

Contents

1	Introduction	1
1.1	Formal Methods in System Design	1
1.1.1	General Remarks and Taxonomy	1
1.1.2	Classification of Formal Methods	4
1.1.3	Classification of Systems	10
1.2	Genealogy of Formal Verification	16
1.2.1	Early Beginnings of Mathematical Logic	16
1.2.2	Automated Theorem Proving	20
1.2.3	Beginnings of Program Verification	23
1.2.4	Dynamic Logics and Fixpoint Calculi	24
1.2.5	Temporal Logics	28
1.2.6	Decidable Theories and ω -Automata	33
1.2.7	Summary	38
1.3	Outline of the Book	40
2	A Unified Specification Language	45
2.1	Kripke Structures as Formal Models of Reactive Systems	45
2.1.1	Simulation and Bisimulation of Kripke Structures	53
2.1.2	Quotient Structures	61
2.1.3	Products of Kripke Structures	66
2.2	Syntax of the Specification Logic $\mathcal{L}_{\text{spec}}$	68
2.3	Semantics of the Specification Logic $\mathcal{L}_{\text{spec}}$	77
2.4	Normal Forms	84
3	Fixpoint Calculi	89
3.1	Partial Orders, Lattices and Fixpoints	90
3.2	The Basic μ -Calculus	98
3.3	Monotonicity of State Transformers	103
3.4	Model Checking of the Basic μ -Calculus	108
3.4.1	A Naive Model Checking Procedure	108
3.4.2	Optimization by the Alternation Depth	111

3.5	Vectorized μ -Calculus	118
3.5.1	State Transformers of Vectorized Fixpoint Expressions ..	119
3.5.2	Decomposing Equation Systems	124
3.5.3	Model Checking Vectorized Fixpoint Expressions	129
3.5.4	Comparing Basic and Vectorized μ -Calculus Model Checking	138
3.5.5	Dependency-Triggered Evaluations	142
3.5.6	The Cleaveland-Steffen Algorithm	148
3.6	Reducing the Alternation Depth w.r.t. Structures	159
3.7	Computing Fair States	164
3.8	Final Remarks on Completeness and Expressiveness	169
3.8.1	Bisimilarity and the Future Fragment	169
3.8.2	Relationship to ω -Tree Automata and Games	173
3.8.3	Dynamic Logic	175
4	Finite Automata	183
4.1	Regular Languages, Regular Expressions and Automata	186
4.2	The Logic of Automaton Formulas	189
4.3	Boolean Closure	194
4.4	Converting Automaton Classes	202
4.5	Determinization and Complementation	209
4.5.1	The Rabin-Scott Subset Construction	210
4.5.2	Determinization of NDet_F	213
4.5.3	Determinization of NDet_G	215
4.5.4	Determinization of NDet_{FG}	219
4.5.5	Reducing NDet_{GF} to $\text{Det}_{\text{Rabin}}$	223
4.6	The Hierarchy of ω -Automata and the Borel Hierarchy	236
4.7	Automata and Monoids	252
4.7.1	Finite Semigroups and Monoids	252
4.7.2	Automata and Their Monoids	257
4.8	Decision Procedures for ω -Automata	264
4.8.1	Flattening ω -Automata	265
4.8.2	Translating \mathcal{L}_ω Model Checking to \mathcal{L}_μ Model Checking	267
4.8.3	Translating Automata to Vectorized μ -Calculus	270
5	Temporal Logics	279
5.1	Introduction	279
5.2	Branching Time Logics – Sublanguages of CTL^*	284
5.2.1	CTL , LTL and CTL^*	285
5.2.2	Adding Syntactic Sugar to CTL	292
5.3	Translating Temporal Logics to the μ -Calculus	299
5.3.1	CTL and Fair CTL as Fragments of the μ -Calculus	300
5.3.2	CTL^2 as a Fragment of the μ -Calculus	302
5.3.3	Eliminating Quantified Boolean Expressions	304

5.3.4	Adding Path Quantifiers	310
5.3.5	Translating LeftCTL* to Vectorized μ -Calculus	313
5.4	Translating Temporal Logics to ω -Automata	329
5.4.1	The Basic Translation from LTL_p to $NDet_{Streett}$	331
5.4.2	Exploitation of Monotonicity	343
5.4.3	Borel Classes of Temporal Logic	348
5.4.4	Reducing Temporal Borel Classes to Borel Automata ..	355
5.4.5	Reductions to CTL/LeftCTL* Model Checking	365
5.5	Completeness and Expressiveness of Temporal Logic	375
5.5.1	Noncounting Automata and Temporal Logic	376
5.5.2	Completeness of the Borel Classes	383
5.5.3	Completeness of the Future Fragments	387
5.6	Complexities of the Model Checking Problems	393
5.7	Reductions by Simulation and Bisimulation Relations	400
6	Predicate Logic	405
6.1	Introduction	405
6.2	Predicate Logics	408
6.2.1	Syntax and Semantics	408
6.2.2	Basics of Predicate Logic	410
6.2.3	Fragments with Decidable Satisfiability Problem	415
6.2.4	Embedding Modal Logics in Predicate Logic	421
6.2.5	Predicate Logic on Linearly Ordered Domains (on \mathbb{N}) ..	424
6.3	Monadic Second Order Logic of Linear Order $MSO_{<}$	428
6.3.1	Equivalence of S1S and $MSO_{<}$	428
6.3.2	Translating $MSO_{<}$ to ω -Automata	434
6.3.3	Büchi's Decision Procedure: Normal Forms for S1S ...	439
6.4	Monadic First Order Logic of Linear Order $MFO_{<}$	442
6.5	Non-Monadic Characterizations	452
7	Conclusions	455
A	Binary Decision Diagrams	459
A.1	Basic Definitions	459
A.2	Basic Algorithms on BDDs	466
A.3	Minimization of BDDs Using Care Sets	471
A.4	Computing Successors and Predecessors	477
A.5	Variable Reordering	483
A.6	Final Remarks	486
B	Local Model Checking and Satisfiability Checking for the μ-Calculus	487
B.1	A Partial Local Model Checking Procedure	488
B.2	A Complete Local Model Checking Procedure	493
B.3	Satisfiability of μ -Calculus Formulas	500

C	Reduction of Structures	527
C.1	Galois Connections and Simulations	527
C.1.1	Basic Properties of Galois Connections	528
C.1.2	Galois Simulation	531
C.2	Abstract Structures and Preservation Results	534
C.3	Optimal and Faithful Abstractions	537
C.4	Data Abstraction	542
C.4.1	Abstract Interpretation of Structures	544
C.4.2	Abstract Specifications	549
C.5	Symmetry and Model Checking	551
C.5.1	Symmetries of Structures	552
C.5.2	Symmetries in the Specification	557
	References	561
	Index	591