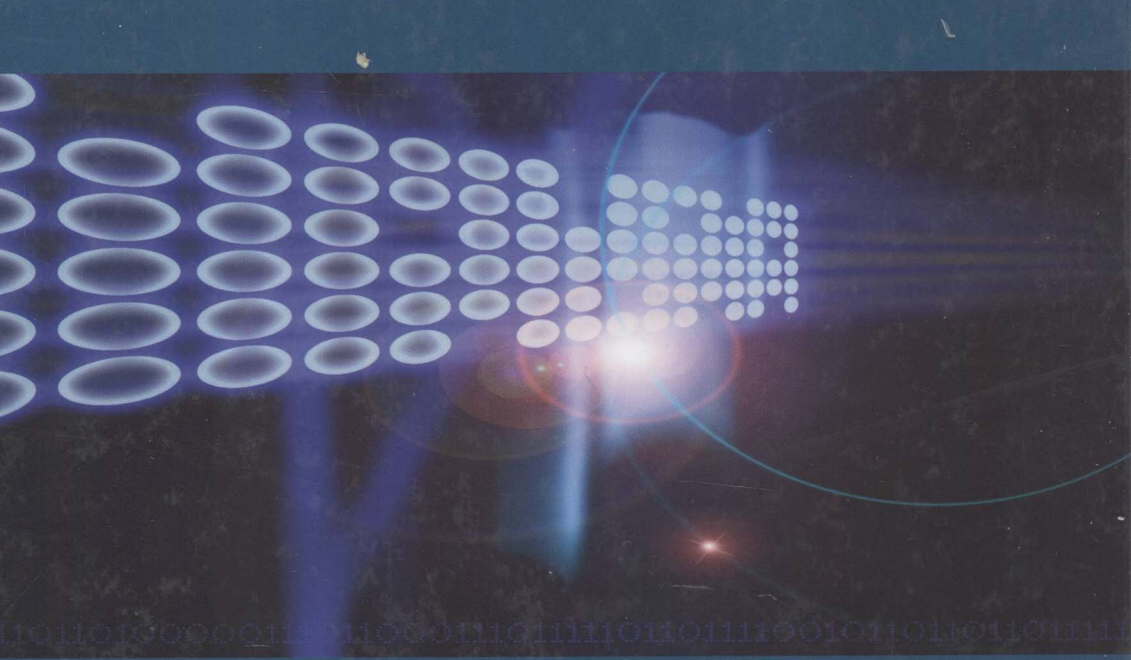




Modeling and Verification of Real-Time Systems

**Edited by
Stephan Merz and Nicolas Navet**

STE

 **WILEY**

Modeling and Verification of Real-Time Systems

TP274
M689

Modeling and Verification of Real-Time Systems

Formalisms and Software Tools

Edited by
Stephan Merz
Nicolas Navet



ISTE



 WILEY

First published in Great Britain and the United States in 2008 by ISTE Ltd and John Wiley & Sons, Inc.

Apart from any fair dealing for the purposes of research or private study, or criticism or review, as permitted under the Copyright, Designs and Patents Act 1988, this publication may only be reproduced, stored or transmitted, in any form or by any means, with the prior permission in writing of the publishers, or in the case of reprographic reproduction in accordance with the terms and licenses issued by the CLA. Enquiries concerning reproduction outside these terms should be sent to the publishers at the undermentioned address:

ISTE Ltd
6 Fitzroy Square
London W1T 5DX
UK

John Wiley & Sons, Inc.
111 River Street
Hoboken, NJ 07030
USA

www.iste.co.uk

www.wiley.com

© ISTE Ltd, 2008

The rights of Stephan Merz and Nicolas Navet to be identified as the authors of this work have been asserted by them in accordance with the Copyright, Designs and Patents Act 1988.

Library of Congress Cataloging-in-Publication Data

Modeling and verification of real-time systems : formalisms and software tools / edited by Nicolas Navet, Stephan Merz.

p. cm.

Includes bibliographical references and index.

ISBN-13: 978-1-84821-013-4

1. Real-time data processing. 2. Computer software--Verification. 3.
Formal methods (Computer science) I. Navet, Nicolas. II. Merz, Stephan.
QA76.54.M635 2008
004.01'51--dc22

2007045063

British Library Cataloguing-in-Publication Data

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

ISBN: 978-1-84821-013-4

Printed and bound in Great Britain by Antony Rowe Ltd, Chippenham, Wiltshire.



Contents

Preface	15
Stephan MERZ and Nicolas NAVET	
 Chapter 1. Time Petri Nets – Analysis Methods and Verification with TINA	19
Bernard BERTHOMIEU, Florent PERES and François VERNADAT	
1.1. Introduction	19
1.2. Time Petri nets	20
1.2.1. Definition	20
1.2.2. States and the state reachability relation	20
1.2.3. Illustration	22
1.2.4. Some general theorems	23
1.3. State class graphs preserving markings and <i>LTL</i> properties	24
1.3.1. State classes	24
1.3.2. Illustration	25
1.3.3. Checking the boundedness property on-the-fly	26
1.3.4. Variations	27
1.3.4.1. Multiple enabledness	27
1.3.4.2. Preservation of markings (only)	28
1.4. State class graphs preserving states and <i>LTL</i> properties	28
1.4.1. Clock domain	28
1.4.2. Construction of the <i>SSCG</i>	29
1.4.3. Variants	30
1.5. State class graphs preserving states and branching properties	32
1.6. Computing firing schedules	33
1.6.1. Schedule systems	33
1.6.2. Delays (relative dates) versus dates (absolute)	34
1.6.3. Illustration	35
1.7. An implementation: the Tina environment	35

1.8. The verification of $SE-LTL$ formulae in Tina	37
1.8.1. The temporal logic $SE-LTL$	37
1.8.2. Preservation of LTL properties by tina constructions	39
1.8.3. selt: the $SE-LTL$ checker of Tina	39
1.8.3.1. Verification technique	39
1.8.3.2. The selt logic	40
1.9. Some examples of use of selt	42
1.9.1. John and Fred	42
1.9.1.1. Statement of problem	42
1.9.1.2. Are the temporal constraints appearing in this scenario consistent?	42
1.9.1.3. Is it possible that Fred took the bus and John the carpool? . .	44
1.9.1.4. At which time could Fred have left home?	44
1.9.2. The alternating bit protocol	44
1.10. Conclusion	47
1.11. Bibliography	48
Chapter 2. Validation of Reactive Systems by Means of Verification and Conformance Testing	51
Camille CONSTANT, Thierry JÉRON, Hervé MARCHAND and Vlad RUSU	
2.1. Introduction	51
2.2. The IOSTS model	54
2.2.1. Syntax of IOSTS	54
2.2.2. Semantics of IOSTS	56
2.3. Basic operations on IOSTS	57
2.3.1. Parallel product	57
2.3.2. Suspension	58
2.3.3. Deterministic IOSTS and determinization	60
2.4. Verification and conformance testing with IOSTS	60
2.4.1. Verification	60
2.4.1.1. Verifying safety properties	62
2.4.1.2. Verifying possibility properties	63
2.4.1.3. Combining observers	63
2.4.2. Conformance testing	64
2.5. Test generation	64
2.6. Test selection	68
2.7. Conclusion and related work	70
2.8. Bibliography	73
Chapter 3. An Introduction to Model Checking	77
Stephan MERZ	
3.1. Introduction	77
3.2. Example: control of an elevator	78

3.3. Transition systems and invariant checking	79
3.3.1. Transition systems and their runs	81
3.3.2. Verification of invariants	82
3.4. Temporal logic	84
3.4.1. Linear-time temporal logic	84
3.4.2. Branching-time temporal logic	87
3.4.3. ω -automata	89
3.4.4. Automata and PTL	92
3.5. Model checking algorithms	94
3.5.1. Local PTL model checking	95
3.5.2. Global CTL model checking	97
3.5.3. Symbolic model checking algorithms	99
3.6. Some research topics	103
3.7. Bibliography	105

Chapter 4. Model Checking Timed Automata 111

Patricia BOUYER and François LAROUSSINIE

4.1. Introduction	111
4.2. Timed automata	112
4.2.1. Some notations	112
4.2.2. Timed automata, syntax and semantics	113
4.2.3. Parallel composition	114
4.3. Decision procedure for checking reachability	115
4.4. Other verification problems	118
4.4.1. Timed languages	118
4.4.2. Branching-time timed logics	118
4.4.3. Linear-time timed logics	120
4.4.4. Timed modal logics	121
4.4.5. Testing automata	121
4.4.6. Behavioral equivalences	121
4.5. Some extensions of timed automata	121
4.5.1. Diagonal clock constraints	122
4.5.2. Additive clock constraints	123
4.5.3. Internal actions	124
4.5.4. Updates of clocks	125
4.5.5. Linear hybrid automata	126
4.6. Subclasses of timed automata	127
4.6.1. Event-recording automata	127
4.6.2. One-clock timed automata	128
4.6.3. Discrete-time models	129
4.7. Algorithms for timed verification	130
4.7.1. A symbolic representation for timed automata: the zones	130
4.7.2. Backward analysis in timed automata	131

4.7.3. Forward analysis of timed automata	132
4.7.4. A data structure for timed systems: DBMs	133
4.8. The model-checking tool Uppaal	134
4.9. Bibliography	135
Chapter 5. Specification and Analysis of Asynchronous Systems using CADP	141
Radu MATEESCU	
5.1. Introduction	141
5.2. The CADP toolbox	142
5.2.1. The LOTOS language	143
5.2.2. Labeled transition systems	143
5.2.3. Some verification tools	144
5.3. Specification of a drilling unit	147
5.3.1. Architecture	150
5.3.2. Physical devices and local controllers	152
5.3.2.1. Turning table	152
5.3.2.2. Clamp	153
5.3.2.3. Drill	154
5.3.2.4. Tester	154
5.3.3. Main controller – sequential version	155
5.3.4. Main controller – parallel version	157
5.3.5. Environment	158
5.4. Analysis of the functioning of the drilling unit	159
5.4.1. Equivalence checking	159
5.4.2. Model checking	161
5.5. Conclusion and future work	164
5.6. Bibliography	166
Chapter 6. Synchronous Program Verification with Lustre/Lesar	171
Pascal RAYMOND	
6.1. Synchronous approach	171
6.1.1. Reactive systems	171
6.1.2. The synchronous approach	172
6.1.3. Synchronous languages	172
6.2. The Lustre language	173
6.2.1. Principles	173
6.2.2. Example: the beacon counter	174
6.3. Program verification	174
6.3.1. Notion of temporal property	175
6.3.2. Safety and liveness	175
6.3.3. Beacon counter properties	175
6.3.4. State machine	175

6.3.5. Explicit automata	176
6.3.6. Principles of model checking	176
6.3.7. Example of abstraction	177
6.3.8. Conservative abstraction and safety	178
6.4. Expressing properties	178
6.4.1. Model checking: general scheme	178
6.4.2. Model checking synchronous program	179
6.4.3. Observers	180
6.4.4. Examples	180
6.4.5. Hypothesis	180
6.4.6. Model checking of synchronous programs	181
6.5. Algorithms	182
6.5.1. Boolean automaton	182
6.5.2. Explicit automaton	182
6.5.3. The “pre ” and “post ” functions	183
6.5.4. Outstanding states	183
6.5.5. Principles of the exploration	184
6.6. Enumerative algorithm	184
6.7. Symbolic methods and binary decision diagrams	185
6.7.1. Notations	185
6.7.2. Handling predicates	186
6.7.3. Representation of the predicates	186
6.7.3.1. Shannon’s decomposition	186
6.7.3.2. Binary decision diagrams	187
6.7.4. Typical interface of a BDD library	188
6.7.5. Implementation of BDDs	188
6.7.6. Operations on BDDs	189
6.7.6.1. Negation	189
6.7.6.2. Binary operators	189
6.7.6.3. Cofactors and quantifiers	190
6.7.7. Notes on complexity	191
6.7.8. Typed decision diagrams	192
6.7.8.1. Positive functions	192
6.7.8.2. TDG	192
6.7.8.3. TDG implementation	193
6.7.8.4. Interest in TDGs	193
6.7.9. Care set and generalized cofactor	194
6.7.9.1. “Knowing that” operators	194
6.7.9.2. Generalized cofactor	194
6.7.9.3. Restriction	194
6.7.9.4. Algebraic properties of the generalized cofactor	195
6.8. Forward symbolic exploration	195
6.8.1. General scheme	196

6.8.2. Detailed implementation	196
6.8.3. Symbolic image computing	198
6.8.4. Optimized image computing	198
6.8.4.1. Principles	198
6.8.4.2. Universal image	199
6.8.4.3. Case of a single transition function	199
6.8.4.4. Shannon's decomposition of the image	200
6.9. Backward symbolic exploration	201
6.9.1. General scheme	201
6.9.2. Reverse image computing	202
6.9.3. Comparing forward and backward methods	203
6.10. Conclusion and related works	203
6.11. Demonstrations	204
6.12. Bibliography	205

Chapter 7. Synchronous Functional Programming with Lucid Synchron 207

Paul CASPI, Grégoire HAMON and Marc POUZET

7.1. Introduction	207
7.1.1. Programming reactive systems	207
7.1.1.1. The synchronous languages	208
7.1.1.2. Model-based design	210
7.1.1.3. Converging needs	211
7.1.2. Lucid Synchron	211
7.2. Lucid Synchron	213
7.2.1. An ML dataflow language	213
7.2.1.1. Infinite streams as basic objects	213
7.2.1.2. Temporal operations: delay and initialization	213
7.2.2. Stream functions	214
7.2.3. Multi-sampled systems	216
7.2.3.1. The sampling operator when	217
7.2.3.2. The combination operator merge	218
7.2.3.3. Oversampling	219
7.2.3.4. Clock constraints and synchrony	220
7.2.4. Static values	222
7.2.5. Higher-order features	222
7.2.6. Datatypes and pattern matching	224
7.2.7. A programming construct to share the memory	225
7.2.8. Signals and signal patterns	227
7.2.8.1. Signals as clock abstractions	227
7.2.8.2. Testing presence and pattern matching over signals	228
7.2.9. State machines and mixed designs	229
7.2.9.1. Weak and strong preemption	229
7.2.9.2. ABRO and modular reset	230

7.2.9.3. Local definitions to a state	231
7.2.9.4. Communication between states and shared memory	232
7.2.9.5. Resume or reset a state	233
7.2.10. Parametrized state machines	233
7.2.11. Combining state machines and signals	234
7.2.12. Recursion and non-real-time features	236
7.2.13. Two classical examples	236
7.2.13.1. The inverted pendulum	236
7.2.13.2. A heater	237
7.3. Discussion	240
7.3.1. Functional reactive programming and circuit description languages	240
7.3.2. Lucid Synchrone as a prototyping language	241
7.4. Conclusion	242
7.5. Acknowledgment	243
7.6. Bibliography	243

Chapter 8. Verification of Real-Time Probabilistic Systems 249

Marta KWIATKOWSKA, Gethin NORMAN, David PARKER
and Jeremy SPROSTON

8.1. Introduction	249
8.2. Probabilistic timed automata	250
8.2.1. Preliminaries	250
8.2.2. Syntax of probabilistic timed automata	252
8.2.3. Modeling with probabilistic timed automata	254
8.2.4. Semantics of probabilistic timed automata	254
8.2.5. Probabilistic reachability and invariance	255
8.3. Model checking for probabilistic timed automata	258
8.3.1. The region graph	258
8.3.2. Forward symbolic approach	261
8.3.2.1. Symbolic state operations	261
8.3.2.2. Computing maximum reachability probabilities	263
8.3.3. Backward symbolic approach	266
8.3.3.1. Symbolic state operations	266
8.3.3.2. Probabilistic until	267
8.3.3.3. Computing maximum reachability probabilities	268
8.3.3.4. Computing minimum reachability probabilities	270
8.3.4. Digital clocks	273
8.3.4.1. Expected reachability	274
8.3.4.2. Integral semantics	276
8.4. Case study: the IEEE FireWire root contention protocol	277
8.4.1. Overview	277
8.4.2. Probabilistic timed automata model	278
8.4.3. Model checking statistics	281

8.4.4. Performance analysis	282
8.5. Conclusion	285
8.6. Bibliography	285

Chapter 9. Verification of Probabilistic Systems Methods and Tools 289
Serge HADDAD and Patrice MOREAUX

9.1. Introduction	289
9.2. Performance evaluation of Markovian models	290
9.2.1. A stochastic model of discrete event systems	290
9.2.2. Discrete-time Markov chains	292
9.2.2.1. Presentation	292
9.2.2.2. Transient and steady-state behaviors of DTMC	293
9.2.3. Continuous-time Markov chains	294
9.2.3.1. Presentation	294
9.2.3.2. Transient and steady-state behaviors of CTMC	295
9.3. High level stochastic models	297
9.3.1. Stochastic Petri nets with general distributions	297
9.3.1.1. Choice policy	298
9.3.1.2. Service policy	298
9.3.1.3. Memory policy	298
9.3.2. GLSPN with exponential distributions	299
9.3.3. Performance indices of SPN	300
9.3.4. Overview of models and methods in performance evaluation	300
9.3.5. The GreatSPN tool	301
9.3.5.1. Supported models	302
9.3.5.2. Qualitative analysis of Petri nets	302
9.3.5.3. Performance analysis of stochastic Petri nets	302
9.3.5.4. Software architecture	302
9.4. Probabilistic verification of Markov chains	303
9.4.1. Limits of standard performance indices	303
9.4.2. A temporal logic for Markov chains	303
9.4.3. Verification algorithms	305
9.4.4. Overview of probabilistic verification of Markov chains	306
9.4.5. The ETMCC tool	307
9.4.5.1. Language of system models	307
9.4.5.2. Language of properties	307
9.4.5.3. Computed results	308
9.4.5.4. Software architecture	308
9.5. Markov decision processes	308
9.5.1. Presentation of Markov decision processes	308
9.5.2. A temporal logic for Markov decision processes	309
9.5.3. Verification algorithms	309
9.5.4. Overview of verification of Markov decision processes	313

9.5.5. The PRISM tool	314
9.5.5.1. Language of system models	314
9.5.5.2. Properties language	314
9.5.5.3. Computed results	314
9.5.5.4. Software architecture	314
9.6. Bibliography	315

Chapter 10. Modeling and Verification of Real-Time Systems

using the IF Toolset	319
---------------------------------------	------------

Marius BOZGA, Susanne GRAF, Laurent MOUNIER and Iulian OBER

10.1. Introduction	320
10.2. Architecture	322
10.3. The IF notation	324
10.3.1. Functional features	324
10.3.2. Non-functional features	326
10.3.3. Expressing properties with observers	328
10.4. The IF tools	329
10.4.1. Core components	329
10.4.2. Static analysis	332
10.4.3. Validation	333
10.4.4. Translating UML to IF	334
10.4.4.1. UML modeling	334
10.4.4.2. The principles of the mapping from UML to IF	334
10.5. An overview on uses of IF in case studies	336
10.6. Case study: the Ariane 5 flight program	337
10.6.1. Overview of the Ariane 5 flight program	337
10.6.2. Verification of functional properties	339
10.6.3. Verification of non-functional properties	343
10.6.4. Modular verification and abstraction	344
10.7. Conclusion	345
10.8. Bibliography	347

Chapter 11. Architecture Description Languages: An Introduction

to the SAE AADL	353
----------------------------------	------------

Anne-Marie DÉPLANCHE and Sébastien FAUCOU

11.1. Introduction	353
11.2. Main characteristics of the architecture description languages	356
11.3. ADLs and real-time systems	357
11.3.1. Requirement analysis	357
11.3.2. Architectural views	359
11.4. Outline of related works	360
11.5. The AADL language	362
11.5.1. An overview of the AADL	363

11.6. Case study	365
11.6.1. Requirements	365
11.6.2. Architecture design and analysis with AADL	366
11.6.2.1. High-level design	366
11.6.2.2. Thread and communication timing semantics	369
11.6.2.3. Technical overview of the flow latency analysis algorithm .	373
11.6.2.4. Modeling fault-tolerance mechanisms	374
11.6.3. Designing for reuse: package and refinement	377
11.7. Conclusion	380
11.8. Bibliography	381
List of Authors	385
Index	389

Preface

The study of real-time systems has been recognized over the past 30 years as a discipline of its own whose research community is firmly established in academia as well as in industry. This book aims at presenting some fundamental problems, methods, and techniques of this domain, as well as questions open for research.

The field is mainly concerned with the control and analysis of dynamically evolving systems for which requirements of timeliness are paramount. Typical examples include systems for the production or transport of goods, materials, energy or information. Frequently, controllers for these systems are “embedded” in the sense that they are physically implemented within the environment with which they interact, such as a computerized controller in a plane or a car. This characteristic imposes strong constraints on space, cost, and energy consumption, which limits the computational power and the available memory for these devices, in contrast with traditional applications of computer science where resources usually grow exponentially according to Moore’s law. The design of real-time systems relies on techniques that originate in several disciplines, including control theory, operations research, software engineering, stochastic process analysis and others.

Software supporting real-time systems needs not only to compute the correct value of a given function, but it must also deliver these values at the right moment in order to ensure the safety and the required performance level of the overall system. Usually, this is implemented by imposing constraints (or *deadlines*) on the termination of certain activities. The verification techniques presented in this volume can help to ensure that deadlines are respected.

The chapters of this book present basic concepts and established techniques for modeling real-time systems and for verifying their properties. They concentrate on functional and timing requirements; the analysis of non-functional properties such as schedulability and Quality of Service guarantees would be a useful complement, but would require a separate volume. Formal methods of system design are based on mathematical principles and abstractions; they are a cornerstone for a “zero-default” discipline. However, their use for the development of real-world systems requires the use of efficient support tools. The chapters therefore emphasize the presentation of verification techniques and tools associated with the different specification methods, as well as the presentation of case studies that illustrate the application of the formalisms and the tools. The focus lies on model checking approaches, which attempt to provide a “push-button” approach to verification and integrate well into standard development processes. The main obstacle for the use of model checking in industrial-sized developments is the state-explosion problem, and several chapters describe techniques based on abstraction, reduction or compression that stretch the limits of the size of systems that can be handled.

Before verification can be applied, the system must be modeled in a formal description language such as (timed) Petri nets, timed automata or process algebra. The properties expected of a system are typically expressed in temporal logic or using automata as observers. Two main classes of properties are *safety* properties that, intuitively, express that nothing bad ever happens, and *liveness* properties that assert that something good eventually happens. The third step is the application of the verification algorithm itself to decide whether the properties hold over the model of the system or not; in the latter case, model checking generates a counter-example exhibiting a run of the system that violates the property.

Beyond *verification*, which compares two formal objects, the model should also be *validated* to ensure that it faithfully represents the system under development. One approach to validation is to decide healthiness properties of the model (for example, ensure that each component action can occur in a system run), and model checking is again useful here. In general, it is helpful to narrow the gap between the system description and its formal model, for example by writing a model in a high-level executable language or in a notation familiar to designers such as UML. The chapters of this book, written by researchers active in the fields, present different possible approaches to the problems of modeling, verification and validation, as well as open research questions.

Chapter 1, written by Bernard Berthomieu, Florent Peres and François Vernadat, explains the analysis of real-time systems based on timed Petri nets. It illustrates the high expressiveness of that formalism and the different, complementary verification techniques that are implemented in the Tina tool.