Real-Time Systems

Real-time systems need to react to certain input stimuli within given time bounds. For example, an airbag in a car has to unfold within 300 milliseconds in a crash. There are many embedded safety-critical applications and each requires real-time specification techniques. This textbook introduces three of these techniques, based on logic and automata: Duration Calculus, Timed Automata, and PLC-Automata.

The techniques are brought together to form a seamless design flow, from real-time requirements specified in the Duration Calculus, via designs specified by PLC-Automata, and into source code for hardware platforms of embedded systems. The syntax, semantics, and proof methods of the specification techniques are introduced; their most important properties are established; and real-life examples illustrate their use. Detailed case studies and exercises conclude each chapter.

Ideal for students of real-time systems or embedded systems, this text will also be of great interest to researchers and professionals in transportation and automation.

E.-R. OLDEROG is Professor of Computer Science at the University of Oldenburg, Germany. In 1994 he was awarded the Leibniz Prize of the German Research Council (DFG).

H. DIERKS is a researcher currently working with OFFIS, a technology transfer institute for computer science in Oldenburg, Germany.

REAL-TIME SYSTEMS

Formal Specification and Automatic Verification

ERNST-RÜDIGER OLDEROG[1] AND HENNING DIERKS[2]

[1] Department of Computing Science, University of Oldenburg, Germany
[2] OFFIS, Oldenburg, Germany

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE
UNIVERSITY PRESS

Shaftesbury Road, Cambridge CB2 8EA, United Kingdom

One Liberty Plaza, 20th Floor, New York, NY 10006, USA

477 Williamstown Road, Port Melbourne, VIC 3207, Australia

314–321, 3rd Floor, Plot 3, Splendor Forum, Jasola District Centre, New Delhi – 110025, India

103 Penang Road, #05–06/07, Visioncrest Commercial, Singapore 238467

Cambridge University Press is part of Cambridge University Press & Assessment,
a department of the University of Cambridge.

We share the University's mission to contribute to society through the pursuit of
education, learning and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/9780521883337

First published 2008

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

ISBN 978-0-521-88333-7 Hardback

Contents

Preface

Computers are used more and more to provide high-quality and reliable products and services, and to control and optimise production processes. Such computers are often embedded into the products and thus hidden to the human user. Examples are computer-controlled washing machines or gas burners, electronic control units in cars needed for operating airbags and braking systems, signalling systems for high-speed trains, or robots and automatic transport vehicles in industrial production lines.

In these systems the computer continuously interacts with a physical environment or plant. Such systems are thus called reactive systems. Moreover, common to all these applications is that the computer reactions should obey certain timing constraints. For example, an airbag has to unfold within milliseconds, not too early and not too late. Reactive systems with such constraints are called real-time systems. They often appear in safety-critical applications where a malfunction of the controller will cause damage and risk the lives of people. This is immediately clear for all applications in the transport sector where computers control cars, trains and planes.

Therefore the design of real-time systems requires a high degree of precision. Here formal methods based on mathematical models of the system under design are helpful. They allow the designer to specify the system at different levels of abstraction and to formally verify the consistency of these specifications before implementing them. In recent years significant advances have been made in the maturity of formal methods that can be applied to real-time systems.

Structure of this book

In this advanced textbook we shall present three such formal approaches:

- Duration Calculus (DC for short), a logic and calculus for specifying high-level requirements of real-time systems;
- timed automata (TA for short), a state-transition model of real-time systems with the advantage of elaborate tool support for the automatic verification of real-time properties;
- PLC-Automata, a state-transition model of real-time systems with the advantage of being implementable, for example in the programming language C or on Programmable Logic Controllers (PLCs for short), a hardware platform that is widespread in the automation industry.

This book is the first one that presents the above three approaches to the specification of real-time systems in a coherent way. This is achieved by combining the approaches into a design method for real-time systems, reaching from requirements down to executable code as illustrated in Figure 0.1. Here:

- Real-time requirements are specified in the Duration Calculus or subsets thereof.
- Designs are specified by PLC-Automata.
- Implementations are written as C programs with timers or as programs that are executable on PLCs.
- Automatic verification of requirements is performed using the model-checking tool UPPAAL for timed automata.
- A tool Moby/RT, built for PLC-Automata, allows the user to invoke algorithms for generating C or PLC code from such automata, and to automatically verify properties specified in a subset of Duration Calculus by using UPPAAL as a back-end verification engine.

The connection is that PLC-Automata have both a semantics in terms of the Duration Calculus and an equivalent one in terms of timed automata. To verify that a PLC-Automaton satisfies a given real-time requirement expressed in the Duration Calculus, there are two possibilities: either a proof can be conducted in the Duration Calculus exploiting the corresponding semantics of the PLC-Automaton, or, for certain types of requirement, an automatic verification is possible using the tool UPPAAL and the timed automata semantics of the PLC-Automaton.

How to read this book

The titles and dependencies of the chapters are shown in Figure 0.2. First, the introduction in Chapter 1 should be read. Here two case studies (railroad

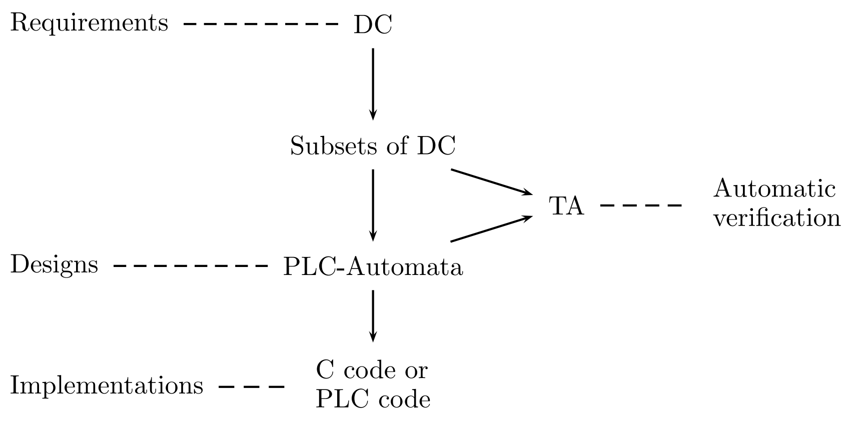

Fig. 0.1. Overview of design method

crossing and gas burner) provide a feeling for the delicacies of real-time systems. Then one can continue with Chapter 2 (Duration Calculus) or Chapter 4 (Timed automata).

Chapter 2 presents the basic knowledge of the Duration Calculus. First, the syntax and semantics of the logic are defined. Then the proof rules of the calculus are introduced, including a simple induction rule. These rules are applied to the case study of the gas burner.

Chapter 3 presents advanced topics on the Duration Calculus. First, decidability results are discussed for the cases of discrete and continuous time domains. Then a subset of the Duration Calculus that is closer to the implementation level is presented, the so-called DC implementables. Finally, Constraint Diagrams are introduced as a graphic representation for requirements with a semantics in the Duration Calculus.

Chapter 4 presents the basic facts of timed automata. In particular, the most prominent result of timed automata is shown: the decidability of the reachability problem. It is then explained which variant of timed automata and properties the model checker UPPAAL can decide.

Chapter 5 introduces PLC-Automata as a class of implementable real-time automata. First, these automata are motivated using an example of a real-time filter. Then it is described how PLC-Automata can be compiled into code that is executable on Programmable Logic Controllers (PLCs). To link the PLC-Automata with the Duration Calculus, their semantics are defined in terms of this logic. As a consequence, a general result estimating the reaction times of PLC-Automata to input stimuli can be proved. Also, an algorithm is discussed that synthesises a PLC-Automaton from a given set of DC implementables provided this set is consistent. Finally, hierarchical PLC-Automata are defined.

Chapter 6 ties together the results of Chapters 4 and 5 for the purposes of automatic verification. It turns out that certain real-time properties of PLC-Automata can be proven automatically using the model checker UPPAAL for timed automata. To this end, an alternative and equivalent semantics of PLC-Automata in terms of timed automata is defined. Then it is shown that real-time requirements expressed in a subset of Constraint Diagrams can be verified against PLC-Automata by checking the reachability of certain states with UPPAAL. This is all supported by the tool Moby/RT, which is described briefly as well. Also, Moby/RT enables the user to compile PLC-Automata into PLC code or C code.

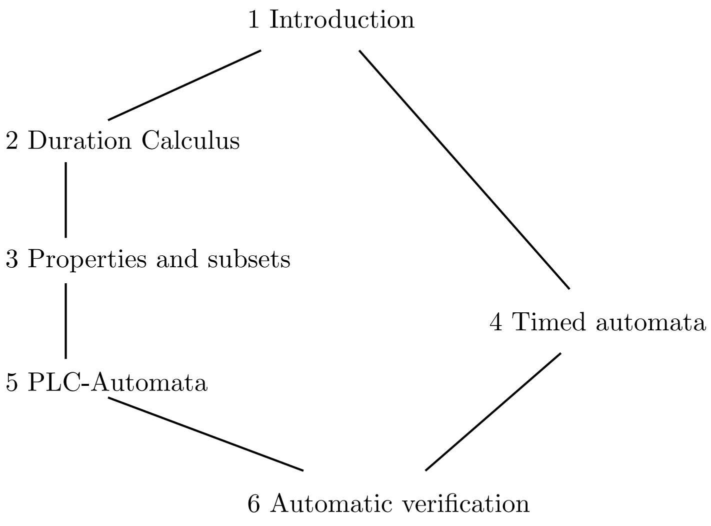

Fig. 0.2. Dependency of chapters

Actually, only Section 5.5 (Synthesis) of Chapter 5 depends on Section 3.2 (DC implementables) of Chapter 3. The remainder of Chapter 5 can thus also be read immediately after Chapter 2.

Intended audience

This textbook is appropriate for either a course on formal methods for real-time systems in the upper division of undergraduate studies or for graduate studies in computer science and engineering. It can also be used for self study, and will be of interest for engineers of embedded real-time systems. Readers are expected to have a basic understanding of mathematical and logical notations.

Courses based on this book

Our own course on real-time systems at the University of Oldenburg is for M.Sc. and advanced B.Sc. students in computer science with an interest in embedded systems; it proceeds as follows:

Course at Oldenburg	
Introduction	1
Duration Calculus	2
Properties and subsets	3.1–3.2
Timed automata	4
PLC-Automata	5.1–5.5
Automatic verification	6 (only short indication)

The course takes one semester with three hours of lectures and one hour of exercises per week.

At Oldenburg an in-depth study of Chapter 6 (Automatic verification) with the use of the tools UPPAAL and Moby/RT is delegated to practical work of the students in separate labs on real-time systems. There LEGO Mindstorm robots are used for implementing the systems. Once desirable real-time properties have been verified, the compiler from PLC-Automata to C is applied to generate code for the LEGO Mindstorms.

An alternative usage of the material of this book could be in (part of) a course on timed automata as follows:

Course based on timed automata	
Introduction	1
Timed automata	4
PLC-Automata	5.1–5.3 and 5.6
Automatic verification	6

Further information and additional material can be found on the webpage `http://csd.informatik.uni-oldenburg.de/rt-book`.

Acknowledgements

Our first inspiring contacts with real-time systems were in the context of the basic research project ProCoS (Provably Correct Systems) funded by the European Commission from 1989 to 1995. This project was planned by Dines Bjørner (Technical University of Denmark), Tony Hoare (Oxford University), and Hans Langmaack (University of Kiel). Its goal was to develop a mathematical basis for the development of embedded, real-time, computer systems.

Returning from a sabbatical at the University of Austin at Texas, Tony Hoare was impressed by the work of Robert S. Boyer and J Strother Moore on mechanical verification exemplified in a case study known as the “CLInc Stack”. Talking to Dines Bjørner and Hans Langmaack, a project on the foundation of verification of many-layered systems was conceived: ProCoS. The different levels of abstraction studied in this project became known as the “ProCoS Tower”. They comprise (informal) expectations, (formal) requirements, (formal) system specifications, programs (occam), machine code (for transputers), and circuit diagrams (netlists). During the project the case study of a gas burner was defined in collaboration with a Danish gas burner manufacturer.

At the project start in 1989 the first author of this book moved from Kiel to Oldenburg to take up a professorship in computing science at the University of Oldenburg and became one of the site leaders of ProCoS. He is very grateful for six rewarding years of research contacts with the members of the ProCoS project group, in particular Hans Langmaack, Tony Hoare, Dines Bjørner, Zhou Chaochen, He Jifeng, Jonathan Bowen, Michael R. Hansen, Anders P. Ravn, Hans Rischel, Kirsten M. Hansen, Martin Fränzle, Markus Müller-Olm, Stephan Rössig, and Michael Schenke. Two highlights evolved during the ProCoS project: the case study of the gas burner and the Duration Calculus, both featuring prominently in this book.

In the first years of ProCoS the second author of this book was a student of computing science and mathematics at Oldenburg. His first contact with the real-time systems of ProCoS was during his master thesis on “The production cell as a verified real-time system” – formalised using the Duration Calculus.

The next decisive step was the collaborative project UniForM (Universal Workbench for Formal Methods) together with Bernd Krieg-Brückner and Jan Peleska (University of Bremen) as well as Alexander Baer and Wolfgang Nowak (company Elpro AG in Berlin). One of the challenges of this project was to develop a formal method to support the real-time programming of tram control systems targeted at Programmable Logic Controllers. Motivated by this challenge the second author developed the concept of a PLC-Automaton, which serves for design specifications in this book.

Inspired by ProCoS and UniForM the research on specification and verification of real-time systems gained momentum at our group on “Correct System Design” at Oldenburg. In particular, we wish to thank Cheryl Kleuker, who contributed Constraint Diagrams, Jochen Hoenicke, who can spot even subtle errors in a minute, and Andreas Schäfer, who saw how to extend the Duration Calculus to cope with space and time. Under the guidance of Josef Tapken the tool MOBY/RT was developed to provide support for the theory presented in this book. We are particularly grateful to the following people who helped create this tool: Hans Fleischhack, Marc Lettrari, Michael Möller, Marco Oetken, Josef Tapken, and Tobe Toben.

The second author spent an extended research visit at the Aalborg University to work with the UPPAAL group on automatic verification and planning of timed automata. He would like to thank Kim Larsen, Gerd Behrmann, Alexandre David, Anders P. Ravn, Wang Yi, and Paul Petterson for inspiring cooperation.

Both authors are pleased to acknowledge the research momentum gained by the Collaborative Research Center AVACS (Automatic Verification and Analysis of Complex Systems) which has been funded by the German Research Council (DFG) since 2004. AVACS groups at the universities of Oldenburg, Freiburg and Saarbrücken, as well as the Max-Planck Institute for Informatics in Saarbrücken, address automatic verification and analysis of real-time systems, hybrid systems, and systems of systems. In the research area of real-time systems we would like to thank our close colleagues Werner Damm, Bernd Becker, Reinhard Wilhelm, Johannes Faber, Roland Meyer, Ingo Brückner, Heike Wehrheim, Bernd Finkbeiner, Andreas Podelski, Andrey Rybalchenko, Viorica Sofroni-Stokkermans, Bernhard Nebel, Jörg Hoffmann, and Sebastian Kupferschmid. We also thank Willem-Paul de Roever for his support of this large-scale project and for many refreshing remarks and suggestions over the years.

Everyone who has written a book knows how difficult it is to find the time to work intensively on the manuscript. Very helpful in this respect was a sabbatical of the first author in the winter semester 2004/05 at ETH Zürich. Many thanks to my perfect hosts David Basin and Barbara Geiser. The first author would also like to thank Krzysztof R. Apt, with whom he wrote his first book, for setting a lucid example of how a book should look and for many pieces of invaluable advice during the past years.

We are very grateful to Michael Möller for creating a draft on which the cover design of this book is based. Last but not least we wish to thank David Tranah and his team from Cambridge University Press who have been very supportive throughout this book project.

List of symbols

$[\nu]$ (region)	159
$[\varphi]$ (region)	159
$\longrightarrow$ (followed-by)	98
$\longrightarrow_0$ (followed-by-initially)	99
$\xrightarrow{\theta}$ (leads-to)	99
$\xrightarrow{\alpha}$ (discrete transition)	140, 142
$\xrightarrow{t}$ (delay transition)	139, 142
$\xrightarrow{\leq\theta}$ (up-to)	100
$\xrightarrow{\leq\theta}_0$ (up-to-initially)	101
$\vdash$ (provability)	54
; (chop)	39
$\circ$ (relational composition)	140
$\cong$ (region equivalence)	157
$\overset{\text{def}}{=}$ (equality by definition)	37
$\overset{\text{def}}{\iff}$ (equivalence by definition)	9
$\Box$ (everywhere)	44
$\Diamond F$ (somewhere)	43
$\lceil P \rceil$ (almost everywhere)	43
$\lceil P \rceil^{t}$ (variant with duration)	43
$\lceil P \rceil^{\leq t}$ (variant with bound)	43
$\lceil\,\rceil$	43
$\mathcal{I}[\![P]\!]$	34
$\models$ (models)	45
$\models_0$ (models from 0)	46
$\models_1$	241
$\simeq$ (isomorphism)	147
A (approaching)	9
Act (set of actions)	136
$B_{?!}$ (action set)	136
Chan (set of channels)	136
Cl (closed)	9
Cr (cross)	9
$DNF(P)$	86
$\mathsf{Des\text{-}1}$ (requirement)	15
$\mathsf{Des\text{-}2}$ (requirement)	15
E (empty)	9
FA (finite alternation)	65
GVar (set of global variables)	31
Intv (set of closed intervals)	36
Lab (set of labels)	136
$\mathbb{N}$ (natural numbers)	294
O (open)	9
Obs (set of observables)	31
$Pred(\pi, t)$	218
Pref (prefix operator)	114
$\mathbb{Q}$ (rational numbers)	294
$\mathbb{R}$ (real numbers)	294
$R(\mathbb{X}, V)$ (reset operations)	167
Req (requirement)	14
Time (time domain)	4
Track	9
Val (set of valuations)	32
$\mathbb{X}$ (set of clocks)	136
$\mathbb{Z}$ (integer)	294
$a!$ (output action)	136
$a?$ (input action)	136
appr	9
$bounds(\pi)$	215
chan (underlying channel)	136
$\mathsf{chan}\, b \bullet \mathcal{A}$ (restriction)	146
cross	9
$delay[\nu]$ (delay operation)	163
empty	9
$first(\pi)$	220
$free$ (set of free global variables)	40
g (gate)	9
$kern(L)$ (kernel)	88
$\langle \ell, \nu \rangle$	140

$\vec{\ell}$ (control vector)	148, 173
$\langle \ell, \nu \rangle, t$ (time-stamped config.)	142
max (maximum)	298
min (minimum)	298
obs (observable)	4
$\langle r_1, \ldots, r_n \rangle$ (reset operations)	167
$\vec{r}$ (reset operations)	167
$\Phi(V)$ (integer constraint)	166
$\Phi(\mathbb{X})$ (clock constraint)	136
$\Phi(\mathbb{X}, V)$ (guards)	167
$\Psi(V)$ (integer expression)	166
$\overline{\alpha}$ (complementary action)	136
λ (label)	136
ν (clock valuation)	138
$\nu + t$ (time shift)	139
$\nu[Y := t]$ (modification)	139
τ (internal action)	136
θ (DC term)	36
ξ (computation path)	142
$\mathcal{C}(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ (network)	173
$\mathcal{D}$ (data type)	4
$\mathcal{I}$ (interpretation)	32
$\mathcal{N}$ (network to timed automata)	148
$\mathcal{P}$ (power set)	295
$\mathcal{R}$ (structure of real numbers)	61
$\mathcal{R}(\mathcal{A})$ (region automaton)	160
$\mathcal{X}$ (formula variable)	72