Texts in Computer Science

Editors
David Gries
Fred B. Schneider

For further volumes:
www.springer.com/series/3191

Krzysztof R. Apt
Frank S. de Boer
Ernst-Rüdiger Olderog

Verification of Sequential and Concurrent Programs

Third, Extended Edition

Springer

Krzysztof R. Apt
Centrum Wiskunde
& Informatica
Science Park 123
1098 XG Amsterdam
Netherlands
k.r.apt@cwi.nl

Frank S. de Boer
Centrum Wiskunde
& Informatica
Science Park 123
1098 XG Amsterdam
Netherlands
F.S.de.Boer@cwi.nl

Ernst-Rüdiger Olderog
Department für Informatik
Universität Oldenburg
26111 Oldenburg
Germany
olderog@informatik.uni-oldenburg.de

Series Editors
David Gries
Department of Computer Science
Upson Hall
Cornell University
Ithaca, NY 14853-7501, USA

Fred B. Schneider
Department of Computer Science
Upson Hall
Cornell University
Ithaca, NY 14853-7501, USA

ISSN 1868-0941 e-ISSN 1868-095X
ISBN 978-1-84882-744-8 e-ISBN 978-1-84882-745-5
DOI 10.1007/978-1-84882-745-5
Springer Dordrecht Heidelberg London New York

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2009932111

© Springer-Verlag London Limited 2009, Reprinted with corrections 2010
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 of licenses issued by the Copyright Licensing Agency. Enquiries concerning reproduction outside those terms should be sent to the publishers.
The use of registered 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 laws and regulations and therefore free for general use.
The publisher makes no representation, express or implied, with regard to the accuracy of the information contained in this book and cannot accept any legal responsibility or liability for any errors or omissions that may be made.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Endorsements

THE THIRD EDITION is an excellent new version of a valuable book. Enhanced with new material on recursion and object-oriented programs, this book now covers methods for verifying sequential, object-oriented, and concurrent programs using well-chosen sample programming languages that highlight fundamental issues and avoid incidental complications. With growing challenges today to produce correct software systems for the future, this book lets students wisely use a few months now to master concepts that will last them a lifetime.

John C. Mitchell, Stanford University

Verification of programs is the Holy Grail of Computer Science. This book makes its pursuit seem both pleasant and worthwhile. Its unique strength lies in the way the authors have deconstructed the apparently complex subject such that each piece carries exactly one idea. The beauty of the presentation extends from the overall structure of the book to the individual explanations, definitions and proofs.

Andreas Podelski, University of Freiburg

Program verification became an interesting research topic of computing science about forty years ago. Research literature on this topic has grown quickly in accordance with rapid development of various programming paradigms. Therefore it has been a challenge to university lecturers on program verification how to carefully select an easy but comprehensive approach, which can fit in with most programming paradigms and can be taught in a systematic way. The publication of this book is an answer to the challenge, and to my knowledge quite many university lecturers have been influenced by the earlier editions of this book if not chosen them as

textbook. Given that the third edition includes verification of object-oriented programs – the most fashionable programming paradigm, and presents it in a way coherent with the approach adopted by the earlier ones, we can expect a further impact of the new edition on university teachings.

Zhou Chaochen, Chinese Academy of Sciences, Beijing

Foreword

THIS BOOK CONTAINS a most comprehensive text that presents syntax-directed and compositional methods for the formal verification of programs. The approach is not language-bounded in the sense that it covers a large variety of programming models and features that appear in most modern programming languages. It covers the classes of sequential and parallel, deterministic and non-deterministic, distributed and object-oriented programs. For each of the classes it presents the various criteria of correctness that are relevant for these classes, such as interference freedom, deadlock freedom, and appropriate notions of liveness for parallel programs. Also, special proof rules appropriate for each class of programs are presented. In spite of this diversity due to the rich program classes considered, there exist a uniform underlying theory of verification which is syntax-oriented and promotes compositional approaches to verification, leading to scalability of the methods.

The text strikes the proper balance between mathematical rigor and didactic introduction of increasingly complex rules in an incremental manner, adequately supported by state-of-the-art examples. As a result it can serve as a textbook for a variety of courses on different levels and varying durations. It can also serve as a reference book for researchers in the theory of verification, in particular since it contains much material that never before appeared in book form. This is specially true for the treatment of object-oriented programs which is entirely novel and is strikingly elegant. I strongly recommend this book to both teachers who wish to train students in the most advanced techniques of verification, and to researchers in this important area.

Amir Pnueli
New York University and the Weizmann Institute of Science, Rehovot

Preface

COMPUTER PROGRAMS ARE by now indispensable parts of systems that we use or rely on in our daily lives. Numerous examples include booking terminals in travel agencies, automatic teller machines, ever more sophisticated services based on telecommunication, signaling systems for cars and trains, luggage handling systems at airports or automatic pilots in airplanes.

For the customers of travel agencies and banks and for the passengers of trains and airplanes the proper functioning and safety of these systems is of paramount importance. Money orders should reflect the right bank accounts and airplanes should stay on the desired route. Therefore the underlying computer programs should work correctly; that is they should satisfy their requirements. A challenge for computer science is to develop methods that ensure program correctness.

Common to the applications mentioned above is that the computer programs have to coordinate a number of system components that can work concurrently, for example the terminals in the individual travel agencies accessing a central database or the sensors and signals used in a distributed railway signaling system. So to be able to verify such programs we need to have at our disposal methods that allow us to deal with correctness of concurrent programs, as well.

Structure of This Book

The aim of this book is to provide a systematic exposition of one of the most common approaches to program verification. This approach is usually called assertional, because it relies on the use of assertions that are attached to program control points. Starting from a simple class of sequential pro-

grams, known as **while** programs, we proceed in a systematic manner in two directions:

- to more complex classes of sequential programs including recursive procedures and objects, and
- to concurrent programs, both parallel and distributed.

We consider here sequential programs in the form of deterministic and nondeterministic programs, and concurrent programs in the form of parallel and distributed programs. Deterministic programs cover **while** programs, recursive programs, and a simple class of object-oriented programs. Nondeterministic programs are used to analyze concurrent programs and the concept of fairness by means of program transformations. Parallel programs consist of several sequential components that can access shared memory. By contrast, distributed programs consist of components with local memory that can communicate only by sending and receiving messages.

For each of these classes of programs their input/output behavior in the sense of so-called partial and total correctness is studied. For the verification of these correctness properties an axiomatic approach involving assertions is used. This approach was initiated by Hoare in 1969 for deterministic programs and extended by various researchers to other classes of programs. It is combined here with the use of program transformations.

For each class of programs a uniform presentation is provided. After defining the syntax we introduce a structured operational semantics as originally proposed by Hennessy and Plotkin in 1979 and further developed by Plotkin in 1981. Then proof systems for the verification of partial and total correctness are introduced, which are formally justified in the corresponding soundness theorems.

The use of these proof systems is demonstrated with the help of case studies. In particular, solutions to classical problems such as producer/consumer and mutual exclusion are formally verified. Each chapter concludes with a list of exercises and bibliographic remarks.

The exposition assumes elementary knowledge of programming languages and logic. Therefore this book belongs to the area of programming languages but at the same time it is firmly based on mathematical logic. All prerequisites are provided in the preparatory **Chapter 2** of Part I.

In Part II of the book we study deterministic programs. In **Chapter 3** Hoare's approach to program verification is explained for **while** programs. Next, we move to the more ambitious structuring concepts of recursive and object-oriented programs. First, parameterless recursive procedures are studied in **Chapter 4**, and then call-by-value parameters are added in **Chapter 5**. These two chapters are taken as preparations to study a class of object-oriented programs in **Chapter 6**. This chapter is based on the work of the second author initiated in 1990, but the presentation is entirely new.

In Part III of the book we study parallel programs with shared variables. Since these are much more difficult to deal with than sequential programs,

they are introduced in a stepwise manner in **Chapters 7**, **8**, and **9**. We base our presentation on the approach by Owicki and Gries originally proposed in 1976 and on an extension of it by the authors dealing with total correctness.

In Part IV we turn to nondeterministic and distributed programs. Nondeterministic sequential programs are studied in **Chapter 10**. The presentation is based on the work of Dijkstra from 1976 and Gries from 1981. The study of this class of programs also serves as a preparation for dealing with distributed programs in **Chapter 11**. The verification method presented there is based on a transformation of distributed programs into nondeterministic ones proposed by the first author in 1986. In **Chapter 12** the issue of fairness is studied in the framework of nondeterministic programs. The approach is based on the method of explicit schedulers developed by the first and third authors in 1983.

Teaching from This Book

This book is appropriate for either a one- or two-semester introductory course on program verification for upper division undergraduate studies or for graduate studies.

In the first lecture the zero search example in Chapter 1 should be discussed. This example demonstrates which subtle errors can arise during the design of parallel programs. Next we recommend moving on to Chapter 3 on **while** programs and before each of the sections on syntax, semantics and verification, to refer to the corresponding sections of the preparatory Chapter 2.

After Chapter 3 there are three natural alternatives to continue. The first alternative is to proceed with more ambitious classes of sequential programs, i.e., recursive programs in Chapters 4 and 5 and then object-oriented programs in Chapter 6. The second alternative is to proceed immediately to parallel programs in Chapters 7, 8, and 9. The third alternative is to move immediately to nondeterministic programs in Chapter 10 and then to distributed programs in Chapter 11. We remark that one section of Chapter 10 can be studied only after the chapters on parallel programs.

Chapter 12 on fairness covers a more advanced topic and can be used during specialized seminars. Of course, it is also possible to follow the chapters in the sequential order as they are presented in the book.

This text may also be used as an introduction to operational semantics. We present below outlines of possible one-semester courses that can be taught using this book. The dependencies of the chapters are shown in Fig. 0.1.

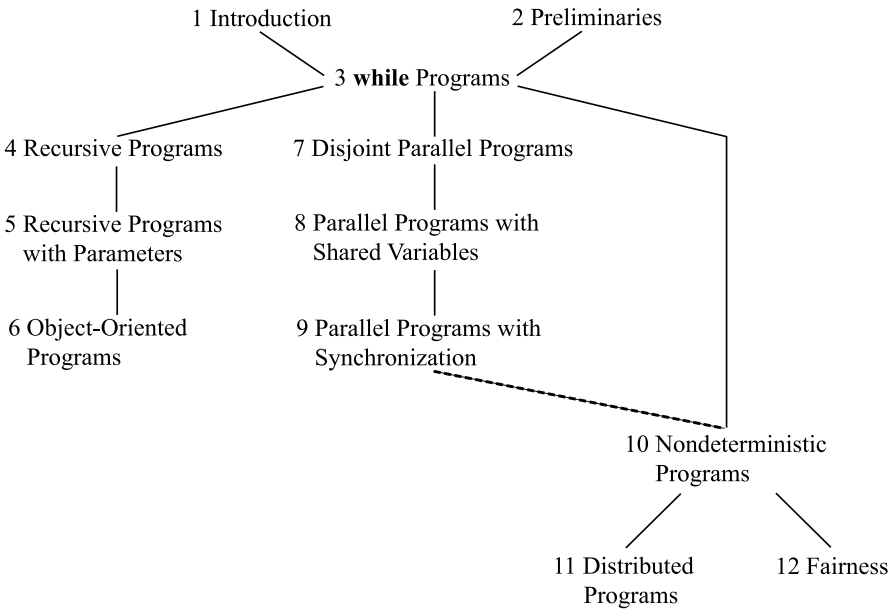

Fig. 0.1 Dependencies of chapters. In Chapter 10 only Section 10.6 depends on Chapter 9.

Changes in the Third Edition

The present, third edition of this book comes with a new co-author, Frank S. de Boer, and with an additional topic that for many years has been at the heart of his research: verification of object-oriented programs. Since this is a notoriously difficult topic, we approach it in a stepwise manner and in a setting where the notational complexity is kept at a minimum. This design decision has led us to add three new chapters to our book.

- In Chapter 4 we introduce a class of recursive programs that extends deterministic programs by parameterless procedures. Verifying such programs makes use of proofs from assumptions (about calls of recursive procedures) that are discharged later on.
- In Chapter 5 this class is extended to the recursive procedures with call-by-value parameters. Semantically, this necessitates the concept of a stack for storing the values of the actual parameters of recursively called procedures. We capture this concept by using a block statement and a corresponding semantic transition rule that models the desired stack behavior implicitly.
- In Chapter 6 object-oriented programs are studied in a minimal setting where we focus on the following main characteristics of objects: they pos-

sess (and encapsulate) their own local variables and interact via method calls, and objects can be dynamically created.

To integrate these new chapters into the original text, we made various changes in the preceding Chapters 2 and 3. For example, in Chapter 3 parallel assignments and failure statements are introduced, and a correctness proof of a program for partitioning an array is given as a preparation for the case study of the *Quicksort* algorithm in Chapter 5. Also, in Chapter 10 the transformation of parallel programs into nondeterministic programs is now defined in a formal way. Also the references have been updated.

Acknowledgments

The authors of this book have collaborated, often together with other colleagues, on the topic of program verification since 1979. During this time we have benefited very much from discussions with Pierre America, Jaco de Bakker, Luc Bougé, Ed Clarke, Werner Damm, Hening Dierks, Edsger W. Dijkstra, Nissim Francez, David Gries, Tony Hoare, Shmuel Katz, Leslie Lamport, Hans Langmaack, Jay Misra, Cees Pierik, Andreas Podelski, Amir Pnueli, Gordon Plotkin, Anders P. Ravn, Willem Paul de Roever, Fred Schneider, Jonathan Stavi and Jeffery Zucker. Many thanks to all of them.

For the third edition Maarten Versteegh helped us with the migration of files to adapt to the new style file. Alma Apt produced all the drawings in this edition.

The bibliography style used in this book has been designed by Sam Buss; Anne Troelstra deserves credit for drawing our attention to it.

Finally, we would like to thank the staff of Springer-Verlag, in particular Simon Rees and Wayne Wheeler, for the efficient and professional handling of all the stages of the production of this book. The TEX support group of Springer, in particular Monsurate Rajiv, was most helpful.

Amsterdam, The Netherlands, *Krzysztof R. Apt* and *Frank S. de Boer*
Oldenburg, Germany, *Ernst-Rüdiger Olderog*

Outlines of One-Semester Courses

PREREQUISITES: Chapter 2.

Course on Program Semantics

Class of programs	Syntax	Semantics
while programs	3.1	3.2
Recursive programs	4.1	4.2
Recursive programs with parameters	5.1	5.2
Object-oriented programs	6.1	6.2
Disjoint parallel programs	7.1	7.2
Parallel programs with shared variables	8.1, 8.2	8.3
Parallel programs with synchronization	9.1	9.2
Nondeterministic programs	10.1	10.2
Distributed programs	11.1	11.2
Fairness	12.1	12.2

Course on Program Verification

Class of programs	Syntax	Semantics	Proof theory
while programs	3.1	3.2	3.3, 3.4, 3.10
Recursive programs	4.1	4.2	4.3, 4.4
Recursive programs with parameters	5.1	5.2	5.3
Object-oriented programs	6.1	6.2	6.3, 6.4, 6.5, 6.6
Disjoint parallel programs	7.1	7.2	7.3
Parallel programs with shared variables	8.1, 8.2	8.3	8.4, 8.5
Parallel programs with synchronization	9.1	9.2	9.3
Nondeterministic programs	10.1	10.2	10.4
Distributed programs	11.1	11.2	11.4

Course Towards Object-Oriented Program Verification

Class of programs	Syntax	Semantics	Proof theory	Case studies
while programs	3.1	3.2	3.3, 3.4	3.9
Recursive programs	4.1	4.2	4.3, 4.4	4.5
Recursive programs with parameters	5.1	5.2	5.3	5.4
Object-oriented programs	6.1	6.2	6.3, 6.4	6.8

Course on Concurrent Program Verification

Class of programs	Syntax	Semantics	Proof theory	Case studies
while programs	3.1	3.2	3.3, 3.4	3.9
Disjoint parallel programs	7.1	7.2	7.3	7.4
Parallel programs with shared variables	8.1, 8.2	8.3	8.4, 8.5	8.6
Parallel programs with synchronization	9.1	9.2	9.3	9.4, 9.5
Nondeterministic programs	10.1	10.2	10.4	10.5
Distributed programs	11.1	11.2	11.4	11.5

Course on Program Verification with Emphasis on Case Studies

Class of programs	Syntax	Proof theory	Case studies
while programs	3.1	3.3, 3.4	3.9
Recursive programs	4.1	4.3, 4.4	4.5
Recursive programs with parameters	5.1	5.4	5.4
Object-oriented programs	6.1	6.3–6.5	6.8
Disjoint parallel programs	7.1	7.3	7.4
Parallel programs with shared variables	8.1, 8.2	8.4, 8.5	8.6
Parallel programs with synchronization	9.1	9.3	9.4, 9.5
Nondeterministic programs	10.1, 10.3	10.4	10.5
Distributed programs	11.1	11.4	11.5

Contents