NETS, TERMS AND FORMULAS

Cambridge Tracts in Theoretical Computer Science

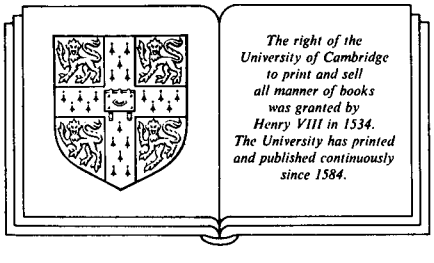
Titles in the series

1. G. Chaitin *Algorithmic Information Theory*
2. L.C. Paulson *Logic and Computation*
3. M. Spivey *Understanding Z*
4. G. Revesz *Lambda Calculus, Combinators and Logic Programming*
5. A. Ramsay *Formal Methods in Artificial Intelligence*
6. S. Vickers *Topology via Logic*
7. J-Y Girard, Y. Lafont & P. Taylor *Proofs and Types*
8. J. Clifford *Formal Semantics & Pragmatics for Natural Language Processing*
9. M. Winslett *Updating Logical Databases*
10. K. McEvoy & J.V. Tucker (eds) *Theoretical Foundations of VLSI Design*
11. T.H. Tse *A Unifying Framework for Stuctured Analysis and Design Models*
12. G. Brewka *Nonmonotonic Reasoning*
13. G. Smolka *Logic Programming over Polymorphically Order–Sorted Types*
15. S. Dasgupta *Design Theory and Computer Science*
17. J.C.M. Baeten (ed) *Applications of Process Algebra*
18. J.C.M. Baeten & W. P. Weijland *Process Algebra*
23. E.-R. Olderog *Nets, Terms and Formulas*

NETS, TERMS AND FORMULAS

Three views of Concurrent Processes and Their Relationship

E.-R. OLDEROG
Professor of Computer Science
Fachbereich Informatik, Universität Oldenburg



CAMBRIDGE UNIVERSITY PRESS
Cambridge
New York Port Chester Melbourne Sydney

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo

Cambridge University Press
The Edinburgh Building, Cambridge CB2 2RU, UK

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

www.cambridge.org
Information on this title: www.cambridge.org/9780521400442

First published 1991
This digitally printed first paperback version 2005

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

ISBN-13 978-0-521-40044-2 hardback
ISBN-10 0-521-40044-9 hardback

ISBN-13 978-0-521-01845-6 paperback
ISBN-10 0-521-01845-5 paperback

PREFACE

The stepwise development of complex systems through various levels of abstraction is good practice in software and hardware design. However, the semantic link between these different levels is often missing. This book is intended as a detailed case study how such links can be established. It presents a theory of concurrent processes where three different semantic description methods are brought together in one uniform framework. Nets, terms and formulas are seen as expressing complementary views of processes, each one describing processes at a different level of abstraction.

- Petri nets are used to describe processes as concurrent and interacting machines which engage in internal actions and communications with their environment or user.
- Process terms are used as an abstract concurrent programming language. Due to their algebraic structure process terms emphasise compositionality, i.e. how complex terms are composed from simpler ones.
- Logical formulas of a first-order predicate logic, called trace logic, are used as a specification language for processes. Logical formulas specify safety and liveness aspects of the communication behaviour of processes as required by their users.

At the heart of this theory are two sets of transformation rules for the top-down design of concurrent processes. The first set can be used to transform logical formulas stepwise into process terms, and the second set can be used to transform process terms into Petri nets. These rules are based on novel techniques for the operational and denotational semantics of concurrent processes.

This book grew out of my research work in the area of concurrent processes which started during my visit to the Programming Research Group in Oxford. The text is based on the my habilitation thesis – a kind of second doctoral thesis – completed at the University of Kiel, and on graduate courses on the subject given at the Universities

of Kiel, Saarbrücken and Oldenburg. Parts of the material have also been presented at international summer schools in France, The Netherlands and Germany.

What I found most difficult when designing the structure of this book was to choose the definitions and concepts in such a way that everything fits together smoothly: Petri nets, process terms, logical formulas, and the transformations. Various subtletees that do not come to surface when writing short research papers had to solved.

How to use this Book

This book is intended for graduate students and researchers interested in concurrency theory. The emphasis is on the relationship between Petri nets, algebraic process terms and logical formulas in the systematic construction of concurent processes. This book does not contain material on Petri nets or process algebra that is covered by other books.

The prerequisites for this book are fluency in the mathematical concepts of sets, relations and functions; familiarity with the basic concepts around automata and regular languages; some practice in reading formulas in predicate logic; and a background in programming. In some parts also basic knowledge from decidability theory is assumed. Apart from these assumptions I have presented the central concepts of this book in a self-contained manner.

A large part of this book is appropriate for a one-semester course on concurrent processes for graduate studies. For a course with emphasis on process construction I suggest the following structure:

Introduction:	1.1,
Petri Nets:	2.1 – 2.2, A, 2.4,
Process Terms:	3.1 – 3.4, 3.5 and 3.8 without proofs, B,
Logical Formulas:	4.1 – 4.5,
Process Construction:	5.1, 5.3 – 5.6.

The figures refer to the sections in the list of contents, and the letters A and B to possible additions from other sources:

- A: More on Petri nets, e.g. from [Rei85]. In particular, verification of net properties using S-invariants and illustrated by mutual exclusion examples fits well to rest of the book.
- B: Introduction to process algebra, e.g. from [BW90b]. The axiomatic view of process algebra should be explained. Also simple examples of process verification using algebraic laws are recommended.

The remaining sections of this book, in particular 2.3, 3.6, 3.7, 4.6, 4.7, 5.2 and 5.7, are advanced. They are suitable for study groups or seminar work. Extensions of the theory and directions for further research are stated in the sections 6.1 – 6.5.

Acknowledgements

H. Langmaack carefully introduced me to computer science and all there is around it, and directed my interest towards program correctness and verification. During a summer school, F.L. Bauer and H. Wössner gave me the bad conscience that program construction by transformation is much better than a posteriori verification. But how to put their advice into practice, I began to see only at Oxford where C.A.R. Hoare introduced me to the exiting world of communicating processes.

My understanding of this subject broadened during visits to Amsterdam with joint work with J.W. de Bakker, J.A. Bergstra, J.W. Klop, J.-J. Ch. Meyer and J.I. Zucker and to Edinburgh with discussions with M. Hennessy, G. Milne, R. Milner, G.D. Plotkin and C. Stirling. My technical interest in nets originates from the work of P. Degano, R. DeNicola and U. Montanari. Invitations by G. Rozenberg to a course on Petri nets and by G. Hotz to his VLSI group in Saarbrücken have been very helpful.

Moreover, I have greatly benefitted from discussions with K.R. Apt, E. Best, M. Bretschneider, M. Broy, R. van Glabbeek, U. Goltz, B. Jonsson, M. Nielsen, A. Pnueli, W. Reisig, W.P. de Roever, P.S. Thiagarajan, B.A. Trakhtenbrot, B. Steffen, F. Vaandrager and J. Zwiers. I am grateful for detailed comments on draft versions of this book from M. Broy, H. Langmaack, S. Rössig, M. Schenke, J. Zwiers, from the participants of a seminar organised by J. Loeckx and A. Schmitt, in particular A. Heckler and M. Hell, and from D. Tranah of Cambridge University Press. Thanks go also to R. Marzinkewitsch and P. Pichol who have critically assisted my lectures on the subject of this book.

Finally, I would like to thank M. Engels, O. Mehlberg, A. Mengel, C. Schier and A. Wallaschek who have – very often concurrently – transformed various fragments of my manuscripts into this LaTeX typescript.

CONTENTS