Monographs in Theoretical Computer Science
An EATCS Series

Springer-Verlag Berlin Heidelberg GmbH

Eike Best
Raymond Devillers
Maciej Koutny

Petri Net Algebra

With 111 Figures

Springer

Authors

Prof. Dr. Eike Best
Fachbereich Informatik
Carl von Ossietzky Universität
Oldenburg
26111 Oldenburg, Germany
Eike.Best@informatik.uni-oldenburg.de

Prof. Dr. Raymond Devillers
Faculté des Sciences
Laboratoire d'Informatique Théorique
Université Libre de Bruxelles
1050 Bruxelles, Belgium
rdevil@ulb.ac.be

Prof. Dr. Maciej Koutny
Department of Computing Science
University of Newcastle
Newcastle upon Tyne NE1 7RU, U.K.
Maciej.Koutny@newcastle.ac.uk

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
Turku Centre for Computer Science
Lemminkäisenkatu 14 A, 20520 Turku
Finland
asalomaa@utu.fi

Library of Congress Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Best, Eike:
Petri net algebra / Eike Best ; Raymond Devillers ; Maciej Koutny. -
Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ;
Milan ; Paris ; Singapore ; Tokyo : Springer, 2001
(Monographs on theoretical computer science)

ACM Computing Classification (1998): F.3.2, F.1.1–2, D.2.2, G.2

ISBN 978-3-642-08677-9 ISBN 978-3-662-04457-5 (eBook)
DOI 10.1007/978-3-662-04457-5

Originally published by Springer-Verlag Berlin Heidelberg New York in 2001.
Softcover reprint of the hardcover 1st edition 2001

Production: PRO EDIT GmbH, Heidelberg
Cover Design: *design & production* GmbH, Heidelberg
Typesetting: Camera ready by authors
Printed on acid-free paper SPIN: 10765212 45/3142/hs – 5 4 3 2 1 0

Preface

In modern society services and support provided by computer-based systems have become ubiquitous and indeed have started to fundamentally alter the way people conduct their business. Moreover, it has become apparent that among the great variety of computer technologies available to potential users a crucial role will be played by concurrent systems. The reason is that many commonly occurring phenomena and computer applications are highly concurrent: typical examples include control systems, computer networks, digital hardware, business computing, and multimedia systems. Such systems are characterised by ever increasing complexity, which results when large numbers of concurrently active components interact. This has been recognised and addressed within the computing science community. In particular, several formal models of concurrent systems have been proposed, studied, and applied in practice.

This book brings together two of the most widely used formalisms for describing and analysing concurrent systems: Petri nets and process algebras. On the one hand, process algebras allow one to specify and reason about the design of complex concurrent computing systems by means of algebraic operators corresponding to common programming constructs. Petri nets, on the other hand, provide a graphical representation of such systems and an additional means of verifying their correctness efficiently, as well as a way of expressing properties related to causality and concurrency in system behaviour. The treatment of the structure and semantics of concurrent systems provided by these two types of models is different, and it is thus virtually impossible to take full advantage of their overall strengths when they are used separately.

The idea of combining Petri nets and process algebras can be traced back to the early 1970s. More directly, this book builds on work carried out by its authors within two EU-funded research projects. It presents a step-by-step development of a rigorous framework for the specification and verification of concurrent systems, in which Petri nets are treated as composable objects, and as such are embedded in a general process algebra. Such an algebra is given an automatic Petri net semantics so that net-based verification techniques, based on structural invariants and causal partial orders, can be applied. The book contains full proofs and carefully chosen examples, and

describes several possible directions for further research. A unique aspect is that the development of the Petri net algebra is handled so as to allow for further application-oriented extensions and modifications.

The book is primarily aimed at researchers, lecturers, and graduate students interested in studying and applying formal methods for concurrent computing systems. It is self-contained in the sense that no previous knowledge of Petri nets and process algebras is required, although we assume that the reader is familiar with basic concepts and notions of set theory and graph theory.

We would like to express our deepest gratitude to our friends and colleagues with whom we conducted the work presented in this monograph. In particular, we would like to thank Javier Esparza, Hans Fleischhack, Bernd Grahlmann, Jon G. Hall, Richard P. Hopkins, Hanna Klaudel, and Elisabeth Pelz. We would also like to thank Wilfried Brauer for his support and encouragement during the writing of this book, Hans Wössner for his excellent editorial advice, and an anonymous reviewer for several very useful comments and suggestions. Last but not least, we would like to thank our respective families for their unfailing support.

December 2000,

Oldenburg — Eike Best
Bruxelles — Raymond Devillers
Newcastle upon Tyne — Maciej Koutny

This book was typeset with Leslie Lamport's LaTeX document preparation system, which is itself based on Donald Knuth's TeX. To produce diagrams, we used the *graphs* macro package by Frank Drewes.

Contents

1. Introduction 1

2. The Petri Box Calculus 7
2.1 An Informal Introduction to CCS 8
2.2 An Informal Introduction to Petri Nets 9
2.3 The Structure and Behaviour of PBC Expressions 11
2.4 Sequential Composition 14
2.5 Synchronisation 15
2.6 Synchronisation and Parallel Composition 21
2.7 Other PBC Operators 24
2.8 Modelling a Concurrent Programming Language 25
2.9 Literature and Background 28

3. Syntax and Operational Semantics 29
3.1 Standard PBC Syntax 29
3.2 Structured Operational Semantics 33
3.2.1 The Basic Setup 35
3.2.2 Equivalence Notions 37
3.2.3 Elementary Actions 41
3.2.4 Parallel Composition 42
3.2.5 Choice Composition 45
3.2.6 Sequential Composition 46
3.2.7 Synchronisation 47
3.2.8 Standard PBC Synchronisation 47
3.2.9 Auto-synchronisation and Multilink-synchronisation 50
3.2.10 Step-synchronisation 52
3.2.11 Basic Relabelling 53
3.2.12 Restriction 54
3.2.13 Scoping 55
3.2.14 Iteration 58
3.2.15 Recursion 59
3.3 Extensions 62
3.3.1 Generalised Iterations 62
3.3.2 Data Variables 64

3.3.3 Generalised Control Flow Operators 66
3.3.4 Generalised Communication Interface Operators 67
3.4 Extended PBC Syntax 69
3.5 Examples of Transition Systems 69
3.6 Literature and Background 71

4. Petri Net Semantics 73
4.1 Compositionality and Nets 73
4.2 Labelled Nets and Boxes 75
4.2.1 An Example 75
4.2.2 Actions and Relabellings 76
4.2.3 Labelled Nets 77
4.2.4 Equivalence Notions 82
4.2.5 Boxes 86
4.3 Net Refinement 90
4.3.1 Operator Boxes 90
4.3.2 Intuition Behind Net Refinement 92
4.3.3 Place and Transition Names 95
4.3.4 Formal Definition of Net Refinement 97
4.3.5 Remarks on Net Refinement 99
4.3.6 Properties 101
4.3.7 Discussion 103
4.4 Petri Net Semantics of PBC 104
4.4.1 Elementary Actions 106
4.4.2 Parallel Composition 106
4.4.3 Choice Composition 107
4.4.4 Sequential Composition 109
4.4.5 Basic Relabelling 110
4.4.6 Synchronisation 110
4.4.7 Restriction 119
4.4.8 Scoping 120
4.4.9 Iteration 120
4.4.10 Data Variables 123
4.4.11 Generalised Control Flow Operators 124
4.4.12 Generalised Communication Interface Operators 126
4.4.13 Generalised Iterations 127
4.5 Refined Operators 129
4.6 Literature and Background 132

5. Adding Recursion 133
5.1 Inclusion Order on Labelled Nets 133
5.2 Solving Recursive Equations 135
5.2.1 Using Fixpoints to Solve Recursive Equations 137
5.2.2 Places and Transitions in Net Solutions 140
5.2.3 An Example of the Limit Construction 144

5.2.4 Deriving Seed Boxes ... 145
5.2.5 A Closed Form of the Maximal Solution ... 151
5.2.6 Minimal Solutions ... 153
5.3 Finitary Equations and Finite Operator Boxes ... 157
5.3.1 Finitary Equation ... 157
5.3.2 Finite Operator Box ... 159
5.4 Further Examples ... 161
5.4.1 Unbounded Parallel Composition ... 161
5.4.2 Rear-unguardedness ... 162
5.4.3 Concurrency Within Unbounded Choice ... 164
5.4.4 Extreme Unguardedness ... 166
5.4.5 (Non)use of Empty Nets in the Limit Construction ... 167
5.5 Solving Systems of Recursive Equations ... 167
5.5.1 Approximations, Existence, and Uniqueness ... 168
5.5.2 A Closed Form of the Maximal Solution ... 169
5.5.3 Guarded Systems ... 171
5.6 Literature and Background ... 172

6. S-invariants ... 173
6.1 S-invariants, S-components, and S-aggregates ... 174
6.1.1 S-invariants ... 176
6.1.2 S-components ... 181
6.1.3 S-aggregates ... 182
6.2 The Synthesis Problem for Net Refinement ... 183
6.2.1 Composing S-invariants ... 185
6.2.2 Multiplicative Distribution Functions ... 190
6.2.3 Ex-binary S-invariants ... 193
6.2.4 Rational Groupings ... 195
6.3 The Synthesis Problem for Recursive Systems ... 200
6.3.1 Name Trees of Nets in the Maximal Solution ... 201
6.3.2 Composing S-invariants for Recursive Boxes ... 202
6.3.3 Coverability Results ... 209
6.4 Finite Precedence Properties ... 216
6.4.1 Process Semantics ... 218
6.4.2 Finite Precedence of Events ... 222
6.4.3 Finiteness of Complete Processes ... 225
6.5 Literature and Background ... 226

7. The Box Algebra ... 227
7.1 SOS-operator Boxes ... 227
7.1.1 A Running Example ... 232
7.1.2 Properties of Factorisations ... 232
7.1.3 The Domain of Application of an SOS-operator Box ... 234
7.1.4 Static Properties of Refinements ... 235
7.1.5 Markings of Nets ... 239

7.2 Structured Operational Semantics of Composite Boxes ... 241
7.2.1 Soundness ... 243
7.2.2 Similarity Relation on Tuples of Boxes ... 245
7.2.3 Completeness ... 248
7.2.4 Solutions of Recursive Systems ... 253
7.2.5 Behavioural Restrictions ... 255
7.3 A Process Algebra and its Semantics ... 259
7.3.1 A Running Example: the DIY Algebra ... 262
7.3.2 Infinite Operators ... 264
7.3.3 Denotational Semantics ... 267
7.3.4 Structural Similarity Relation on Expressions ... 270
7.3.5 Transition-based Operational Semantics ... 279
7.3.6 Consistency of the Two Semantics ... 286
7.3.7 Label-based Operational Semantics ... 287
7.3.8 Partial Order Semantics of Box Expressions ... 290
7.4 Literature and Background ... 294

8. PBC and Other Process Algebras ... 295
8.1 (Generalised) PBC is a Box Algebra ... 295
8.1.1 PBC Without Loops ... 295
8.1.2 Safe Translation of the Ternary PBC Iteration ... 299
8.1.3 PBC with Generalised Loops ... 306
8.2 Other Process Algebras ... 308
8.2.1 CCS ... 310
8.2.2 TCSP ... 311
8.2.3 COSY ... 311
8.3 Literature and Background ... 312

9. A Concurrent Programming Language ... 313
9.1 Syntax of *Razor* ... 313
9.1.1 Programs and Blocks ... 315
9.1.2 Declarations ... 315
9.1.3 Commands and Actions ... 316
9.1.4 Guarded Commands ... 316
9.1.5 Expressions and Operators ... 317
9.1.6 Syntactic Variations ... 317
9.2 Semantics of *Razor* ... 318
9.2.1 Programs, Blocks, and Declarations ... 319
9.2.2 Basic Channel Processes ... 321
9.2.3 Command Connectives ... 324
9.2.4 Actions and Guarded Commands ... 325
9.3 Three *Razor* Programs ... 329
9.4 Adding Recursive Procedures ... 332
9.5 Some Consequences of the Theory ... 336

9.6 Proofs of Distributed Algorithms 340
9.6.1 A Final Set of Petri-Net-Related Definitions 340
9.6.2 Peterson's Mutual Exclusion Algorithm 342
9.6.3 Dekker's and Morris's Mutual Exclusion Algorithms .. 346
9.7 Literature and Background 347

10. Conclusion 349

Appendix: Solutions of Selected Exercises 351

References 362

Index 369