Cambridge Tracts in Theoretical Computer Science 50
Process Algebra: Equational Theories of Communicating Processes

Process algebra is a widely accepted and much used technique in the specification and verification of parallel and distributed software systems. This book sets the standard for the field. It assembles the relevant results of most process algebras currently in use, and presents them in a unified framework and notation.

The authors describe the theory underlying the development, realization and maintenance of software that occurs in parallel or distributed systems. A system can be specified in the syntax provided, and the axioms can be used to verify that a composed system has the required external behavior. As examples, two protocols are completely specified and verified in the text: the Alternating-Bit communication Protocol, and Fischer's Protocol of mutual exclusion.

The book serves as a reference text for researchers and graduate students in computer science, offering a complete overview of the field and referring to further literature where appropriate.

J. C. M. BAETEN is Professor of formal methods in the Division of Computer Science at Eindhoven University of Technology, Netherlands.

T. BASTEN is Professor of computational models in the Faculty of Electrical Engineering at Eindhoven University of Technology, Netherlands, and Research Fellow at the Embedded Systems Institute, Eindhoven.

M. A. RENIERS is Assistant Professor in the Division of Computer Science at Eindhoven University of Technology, Netherlands.

Cambridge Tracts in Theoretical Computer Science 50

Editorial Board

Titles in the series

A complete list of books in the series can be found at
http://www.cambridge.org/uk/series/sSeries.asp?code=CTTC.
Recent titles include the following:

21. D. A. Wolfram *The Clausal Theory of Types*
22. V. Stoltenberg-Hansen, Lindström & E. R. Griffor *Mathematical Theory of Domains*
23. E.-R. Olderog *Nets, Terms and Formulas*
26. P. D. Mosses *Action Semantics*
27. W. H. Hesselink *Programs, Recursion and Unbounded Choice*
28. P. Padawitz *Deductive and Declarative Programming*
29. P. Gärdenfors (ed.) *Belief Revision*
30. M. Anthony & N. Biggs *Computational Learning Theory*
31. T. F. Melham *Higher Order Logic and Hardware Verification*
32. R. L. Carpenter *The Logic of Typed Feature Structures*
33. E. G. Manes *Predicate Transformer Semantics*
34. F. Nielson & H. R. Nielson *Two-Level Functional Languages*
35. L. M. G. Feijs & H. B. M. Jonkers *Formal Specification and Design*
36. S. Mauw & G. J. Veltink (eds.) *Algebraic Specification of Communication Protocols*
37. V. Stavridou *Formal Methods in Circuit Design*
38. N. Shankar *Metamathematics, Machines and Gödel's Proof*
39. J. B. Paris *The Uncertain Reasoner's Companion*
40. J. Desel & J. Esparza *Free Choice Petri Nets*
41. J.-J. Ch. Meyer & W. van der Hoek *Epistemic Logic for AI and Computer Science*
42. J. R. Hindley *Basic Simple Type Theory*
43. A. S. Troelstra & H. Schwichtenberg *Basic Proof Theory*
44. J. Barwise & J. Seligman *Information Flow*
45. A. Asperti & S. Guerrini *The Optimal Implementation of Functional Programming Languages*
46. R. M. Amadio & P.-L. Curien *Domains and Lambda-Calculi*
47. W.-P. de Roever & K. Engelhardt *Data Refinement*
48. H. Kleine Büning & T. Lettman *Propositional Logic*
49. L. Novak & A. Gibbons *Hybrid Graph Theory and Network Analysis*
51. H. Simmons *Derivation and Computation*
52. P. Blackburn, M. de Rijke & Y. Venema *Modal Logic*
53. W.-P. de Roever et al *Concurrency Verification*
54. Terese *Term Rewriting Systems*
55. A. Bundy et al *Rippling: Meta-Level Guidance for Mathematical Reasoning*

Process Algebra: Equational Theories of Communicating Processes

J. C. M. BAETEN

T. BASTEN

M. A. RENIERS

Eindhoven University of Technology, Netherlands

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 238467x

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/9780521820493

First published 2010

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

ISBN 978-0-521-82049-3 Hardback

Additional resources for this publication at www.processalgebra.org

Contents

Forewords

Tony Hoare
Cambridge, United Kingdom, February 2009

Algebra is the simplest of all branches of mathematics. After the study of numerical calculation and arithmetic, algebra is the first school subject which gives the student an introduction to the generality and power of mathematical abstraction, and a taste of mathematical proof by symbolic reasoning. Only the simplest reasoning principle is required: the substitution of equals for equals. (Even computers are now quite good at it.) Nevertheless, the search for algebraic proof still presents a fascinating puzzle for the human mathematician, and yields results of surprising brevity and pleasing elegance.

A more systematic study of algebra provides a family tree that unifies the study of many of the other branches of mathematics. It identifies the basic mathematical axioms that are common to a whole sub-family of branches. The basic theorems that are proved from these axioms will be true in every branch of mathematics which shares them. At each branching point in the tree, the differences between the branches are succinctly highlighted by their choice between a pair of mutually contradictory axioms. In this way, algebra is both cumulative in its progress along the branches, and modular at its branching points.

It is a surprise to many computer programmers that computer programs, with all their astronomical complexity of structure and behavior, are as amenable to the axioms of algebra as simple numbers were at school. Indeed, algebra scales well from the small to the large. It applies to the large-scale behavior of systems evolving concurrently in parallel; and it underlies the manipulation of the minute detail of the individual instructions of code. At the highest level, algebraic reasoning provides the essential basis for program transformations that match the structure of a complete system to that of the available hardware configuration; and at the lowest level, it provides the justification for optimization

of the code of a program so as to achieve the full potential of the processor instruction set.

This book exploits the power of algebra to explore the properties of concurrent programs, particularly those that control the behavior of distributed systems, communicating with neighbors on identified channels. It starts with the simplest theories at the base of the tree, and gradually extends them in a modular way by additional axioms to deal with sequential programming as well as concurrent, and with communication as well as assignment. In the later chapters, it exploits the modularity of algebra to describe priorities, probabilities, and mobility.

The technical approach of the book is grounded in Computer Science. It emphasizes term models based on the syntax of the operators used in the algebra, and it justifies the axioms by appeal to the execution of the terms as programs. Its crowning achievement is to exploit the unifying power of algebra to cover a range of historic theories, developed for various purposes up to 20 years ago. Earlier these theories were thought to be irreconcilable rivals; but as a result of research by the authors of this book and others, the rivals are now seen to be close family members, each superbly adapted to the particular range of problems that it was designed to tackle.

Robin Milner
Cambridge, United Kingdom, February 2009

Nowadays, much of what is still called 'computing' involves the behavior of composite systems whose members interact continually with their environment. A better word is 'informatics', because we are concerned not just with calculation, but rather with autonomous agents that interact with – or inform – one another. This interactivity bursts the bounds of the sequential calculation that still dominates many programming languages. Does it enjoy a theory as firm and complete as the theory of sequential computation? Not yet, but we are getting there.

What is an informatic process? The answer must involve phenomena foreign to sequential calculation. For example can an informatic system, with many interacting components, achieve deterministic behavior? If it can, that is a special case; non-determinism is the norm, not the exception. Does a probability distribution, perhaps based upon the uncertainty of timing, replace determinism? Again, how exactly do these components interact; do they send each other messages, like email, to be picked up when convenient? – or is each interaction a kind of synchronized handshake?

Over the last few decades many models for interactive behavior have been

proposed. This book is the fruit of 25 years of experience with an algebraic approach, in which the constructors by which an informatic system is assembled are characterized by their algebraic properties. The characteristics are temporal, in the same way that sequential processes are temporal; they are also spatial, describing how agents are interconnected. And their marriage is complex.

The authors have teased out primitive elements of this structure. In doing so they have applied strict mathematical criteria. For example, to what extent can the dynamic characteristics of a set of process constructors be reflected, soundly or completely, by a collection of algebraic axioms? And to what extent can the authors' calculus ACP (Algebra of Communicating Processes) be harmonized with other leading calculi, especially Hoare's CSP (Communicating Sequential Processes) and Milner's CCS (Calculus of Communicating Systems)? Consideration of these questions gives the book a firm and appreciable structure. In particular, it shows up a shortcoming of what (for some 50 years) has been called automata theory: that theory never seriously attempted to model the ways in which two automata might interact.

So it may seem that the book is mainly an account of the frontiers of research into process theory. It is much more, and I hope that syllabus designers will take note: the presentation is lucid and careful, enriched with exercises, to the extent that many parts of it can be used as a basis for university courses, both undergraduate and postgraduate. If in such courses we expose students to theories that are still advancing, then they share the excitement of that progress.

Jan Bergstra
Amsterdam, the Netherlands, February 2009

This book about process algebra improves on its predecessor, written by Jos Baeten and Peter Weijland almost 20 years ago, by being more comprehensive and by providing far more mathematical detail. In addition the syntax of ACP has been extended by a constant 1 for termination. This modification not only makes the syntax more expressive, it also facilitates a uniform reconstruction of key aspects of CCS, CSP as well as ACP, within a single framework.

After renaming the empty process (ϵ) into 1 and the inactive process (δ) into 0, the axiom system ACP is redesigned as BCP. This change is both pragmatically justified and conceptually convincing. By using a different acronym instead of ACP, the latter can still be used as a reference to its original meaning, which is both useful and consistent.

Curiously these notational changes may be considered marginal and significant at the same time. In terms of theorems and proofs, or in terms of case

studies, protocol formalizations and the design of verification tools, the specific details of notation make no real difference at all. But by providing a fairly definitive and uncompromising typescript a major impact is obtained on what might be called 'nonfunctional qualities' of the notational framework. I have no doubt that these nonfunctional qualities are positive and merit being exploited in full detail as has been done by Baeten and his co-authors. Unavoidably, the notational evolution produces a change of perspective. While, for instance, the empty process is merely an add on feature for ACP, it constitutes a conceptual cornerstone for BCP.

Group theory provides different notational conventions (additive and multiplicative) for the same underlying structure, and in a similar fashion, the format of this book might be viewed as a comprehensive and consistent notational convention for a theory of process algebra. But the same theory might instead be captured, when used in another context, with a different preferred notation.

Process algebra as presented here is equational logic of processes viewed as a family of first order theories. Each theory is provided with its Tarski semantics, making use of many-sorted algebras with total operations and nonempty sorts. Although this may sound already quite technical and may be even prohibitive for a dedicated computer scientist, these are the most stable and clear-cut semantic principles that mathematical logic has developed thus far. When developing axioms for process algebras the authors do not deviate from that path for ad hoc reasons of any kind. Thus there is a very clear separation between the logical metatheory, consisting of first order equational logic and its model-theory on the one hand and the many design decisions concerning the subject matter that is process theory on the other hand. A prominent methodological principle underlying the work is that the significant ramification of design decisions concerning various process formalisms can be made systematic in a way comparable to the development of say ring theory in mathematics.

In addition to making use of first order logic, the equational form of most axioms allows a systematic exploitation of technical results from term rewriting. This is not a matter of process theory per se but it is proving helpful throughout the book.

One may ask why process algebra should be considered a topic in computer science and not just in applied mathematics. While some parts of process theory are now moving in the direction of systems biology, the process algebras covered in this book may sooner or later show up in physics. Quantum process algebras have not been covered here but various forms already exist and that kind of development is likely to continue for many more years.

Just like its predecessor has been for many years, this book will definitely be useful as a reference work for research in process theory.

Preface

What is this book about?

This book sets the standard for process algebra. It assembles the relevant results of most process algebras currently in use, and presents them in a unified framework and notation. It addresses important extensions of the basic theories, like timing, data parameters, probabilities, priorities, and mobility. It systematically presents a hierarchy of algebras that are increasingly expressive, proving the major properties each time.

For researchers and graduate students in computer science, the book will serve as a reference, offering a complete overview of what is known to date, and referring to further literature where appropriate.

Someone familiar with CCS, the Calculus of Communicating Systems, will recognize the minimal process theory MPT as basic CCS, to which a constant expressing successful termination is added, enabling sequential composition as a basic operator, and will then find a more general parallel-composition operator. Someone familiar with ACP, the Algebra of Communicating Processes, will see that termination is made explicit, leading to a replacement of action constants by action prefixing, but will recognize many other things. The approaches to recursion of CCS and ACP are both explained and integrated. Someone familiar with CSP, Communicating Sequential Processes, will have to cope with more changes, but will see the familiar operators of internal and external choice and parallel composition explained in the present setting.

The book is a complete revision of another (Baeten & Weijland, 1990). Moreover, as the unification theme has become more important, it can also be seen as a successor to (Milner, 1989) and (Hoare, 1985).

Process algebra has become a widely accepted and used technique in the specification and verification of parallel and distributed software systems. A system can be specified in the syntax provided, and the axioms can be used

to verify that a composed system has the required external behavior. As examples, a couple of protocols are completely specified and verified in the text: the Alternating-Bit communication Protocol, and Fischer's protocol of mutual exclusion. The book explains how such a task can be undertaken for any given parallel or distributed system. As the system gets bigger though, tool support in this endeavor, using for example the mCRL2 tool set, will become necessary.

Despite the breadth of the book, some aspects of process algebra are not covered. Foremost, relationships with logic, important when discussing satisfaction of requirements, are not addressed (except briefly in Chapter 10). This was left out as it does not pertain directly to the algebraic treatment and would make the book too voluminous. Readers interested in this subject may refer to (Bradfield & Stirling, 2001). Some of the extensions (probabilities, priorities, and mobility) are only briefly touched upon; references are given to literature that provides a more in-depth treatment.

How to use this book?

The book can be used in teaching a course to students at advanced undergraduate or graduate level in computer science or a related field. It contains numerous exercises varying in difficulty. Such a course should cover at least Chapters 1 through 8, together comprising a complete theory usable in applications. From the remaining chapters, different choices can be made, since the chapters can be read independently. The text is also suitable for self-study.

Chapter 1 presents an introduction that delineates more precisely the field of process algebra. It also gives a historic overview of the development of the field. In Chapter 2, some notions are explained that are used in the remainder of the book. The material concerns equational theories, algebras, and term rewriting systems. Chapter 3 covers the semantic domain of transition systems. The notion of bisimilarity is explained, and the concept of a structural operational semantics, a standard way to assign transition systems to terms in a language, is introduced. Some relevant theorems about structural operational semantics are presented and are used in the remainder of the book when providing semantics to equational theories.

Chapter 4 starts with process algebra. A minimal process theory is presented, that illustrates the basic steps involved in establishing a set of laws and a model for an equational theory. Then, two basic extensions are considered: first, the extension with the successful-termination constant 1, and second, the extension with projection operators. Differences between the two types of extensions are explained. As a prequel to the succeeding chapter, the extension

with the iteration operator is considered. By means of this operator, some infinite processes can be defined.

Chapter 5 is devoted to recursion, which is the main means of specifying non-terminating processes. It is shown how to define such processes, and how to reason with them. The unbounded stack is considered as a first example.

Chapter 6 adds sequential composition. Furthermore, it looks at renaming operators, and operators that can block or skip part of a process. Chapter 7 adds parallel composition and communication. Buffers and bags are considered as examples. As a larger example, the specification of the Alternating-Bit Protocol is presented.

Chapter 8 considers abstraction, which enables one to hide some of the behavior of a system in order to focus on the rest. It is the central ingredient for verification. As an example, a verification is presented of the Alternating-Bit Protocol. The notions of divergence and fairness are explained and treated.

Chapter 9 gives a short introduction to the extension with explicit timing. Fischer's protocol is presented as an example. Chapter 10 considers the interplay between data and processes, and at the same time takes a closer look at the notion of a state, and what can be observed from it. Chapter 11 briefly covers some extensions to and variants of the basic theories, namely priorities, probabilities, mobility, and different forms of parallel composition. The book concludes with Chapter 12, which considers other semantics, besides bisimulation semantics, and their interrelations.

The book is supported through the website **www.processalgebra.org**. This website contains supplementary material, such as solutions to selected exercises, slides presenting the book content, additional exercises and exam problems, a tool to support algebraic reasoning, an up-to-date process algebra bibliography, and much more. Lecturers, students, and researchers are invited to have a look at the website, to get the most out of using the book.

Acknowledgements

A book like this can only be written thanks to the contributions and support of many individuals. The authors wish to acknowledge Luca Aceto, Suzana Andova, Jan Bergstra, Maarten Boote, Victor Bos, Beverley Clarke, Clare Dennison, Henk Goeman, Jan Friso Groote, Evelien van der Heiden, Leszek Holenderski, Abigail Jones, Hugo Jonker, Gerben de Keijzer, Uzma Khadim, Fabian Kratz, Kim Larsen, Bas Luttik, Jasen Markovski, Sjouke Mauw, Kees Middelburg, Mohammad Mousavi, Dennis van Opzeeland, Ralph Otten, Alban Ponse, Isabelle Reymen, Maria van Rooijen, Eugen Schindler, Natalia

Sidorova, David Tranah, Pieter Verduin, Jeroen Voeten, Marc Voorhoeve, Peter Weijland, and Tim Willemse.