

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

University of Dortmund, Germany

Madhu Sudan

Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos

New York University, NY, USA

Doug Tygar

University of California, Berkeley, CA, USA

Moshe Y. Vardi

Rice University, Houston, TX, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Ali E. Abdallah Cliff B. Jones
Jeff W. Sanders (Eds.)

Communicating Sequential Processes

The First 25 Years

Symposium on the Occasion of 25 Years of CSP
London, UK, July 7-8, 2004
Revised Invited Papers



Springer

Volume Editors

Ali E. Abdallah
London South Bank University
Faculty of BCIM
Institute for Computing Research
103 Borough Road, London, SE1 0AA, UK
E-mail: A.Abdallah@lsbu.ac.uk

Cliff B. Jones
University of Newcastle upon Tyne
School of Computing Science
Newcastle upon Tyne, NE1 7RU, UK
E-mail: cliff.jones@ncl.ac.uk

Jeff W. Sanders
Oxford University Computing Laboratory
Parks Road, Oxford OX1 3QD, UK
E-mail: Jeff.Sanders@comlab.ox.ac.uk

The cover illustration is the work of Bill Roscoe.

Library of Congress Control Number: 2005925390

CR Subject Classification (1998): D.2.4, F.3, D.1.3, D.3.1

ISSN	0302-9743
ISBN-10	3-540-25813-2 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-25813-1 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springeronline.com

© Springer-Verlag Berlin Heidelberg 2005
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11423348 06/3142 5 4 3 2 1 0

Preface

This volume, like the symposium CSP25 which gave rise to it, commemorates the semi-jubilee of Communicating Sequential Processes.

Tony Hoare's paper "Communicating Sequential Processes"¹ is today widely regarded as one of the most influential papers in computer science. To commemorate it, an event was organized under the auspices of BCS-FACS (the British Computer Society's Formal Aspects of Computing Science specialist group). CSP25 was one of a series of such events organized to highlight the use of formal methods, emphasize their relevance to modern computing and promote their wider application. BCS-FACS is proud that Tony Hoare presented his original ideas on CSP at one of its first meetings, in 1978.

The two-day event, 7–8 July 2004, was hosted by London South Bank University's Institute for Computing Research, Faculty of Business, Computing and Information Management. The intention was to celebrate, reflect upon and look beyond the first quarter-century of CSP's contributions to computer science. The meeting examined the impact of CSP on many areas stretching from semantics (mathematical models for understanding concurrency and communications) and logic (for reasoning about behavior), through the design of parallel programming languages (i/o, parallelism, synchronization and threads) to applications varying from distributed software and parallel computing to information security, Web services and concurrent hardware circuits. It included a panel discussion with panelists Brookes, Hoare, de Roever and Roscoe (chaired by Jeff Sanders), poster presentations by PhD students and others, featured a fire alarm (requiring evacuation in the rain!) and concluded with the presentation of a fountain pen to Prof. Sir C. A. R. Hoare.

We owe thanks to the BCS-FACS steering committee and its chairman, Jonathan P. Bowen, for their overwhelming support. Special thanks are due to Dedian Hopkin (LSBU Vice Chancellor) for opening the event in the newly built Keyworth Centre; Chris Clare (Dean), Geoff Elliot (Deputy-Dean), and Terry Fogarty (Head of the Institute for Computing Research) for providing a stimulating environment for hosting the event. Our gratitude goes to our sponsors for their generous support: Microsoft Research, Cambridge, UK; Formal Systems Europe, Limited; Handshake Solutions, Philips, Netherlands; Verified Systems International, GmbH, Germany; Formal Methods Europe (FME); and London South Bank University, Institute for Computing Research. We would like to thank the local organization team: Ali N. Haidar, Michelle Hammond, Kalpesh Kapour and Paul Boca for their hard work to ensure the smooth running of the local arrangements.

¹ *Communications of the ACM*, **21**(8):666–667, 1978.

We would also like to thank Bill Roscoe for his “Golden Valley”² painting used in the cover of this book. This was the favorite among CSP25 authors who considered several other alternatives. What’s its relevance to CSP25? In the words of one of the contributing authors:

It’s a lovely scene with a prominent feature, the much-branching tree representing CSP, and the road winding off representing the 25 years so far, with the rest hidden behind the tree. (Who knows where it may still lead?)

After presentation at the symposium, the contributions were reworked by their authors and fully refereed. We are grateful to all for their timely and efficient work, particularly in the refereeing process where helpful and incisive comments were made. The resulting papers are gathered here, as they were in the workshop, into session-sized chunks, described below.

The conference website can be found at www.lsbu.ac.uk/menass/csp25 and www.bcs-facs.org

Semantic Foundations

The first paper to confront the denotational semantics of CSP with due regard to the interplay between communication and abstraction was “A Theory of Communicating Sequential Processes”³ by Steve Brookes, Tony Hoare and Bill Roscoe. Before it, the simplistic but intuitively compelling traces model had been the basis for a semantics capable of capturing safety properties but not of capturing liveness (being too weak to capture deadlock or divergence). That paper concentrated on the communicating fragment of CSP, TCSP, based on recursively defined communicating processes evolving in parallel. The study concentrated, inevitably, on the distinction between the choice of events due to the environment choosing from a menu (external choice) and as the result of abstraction (internal, or nondeterministic, choice); in a subsequent paper⁴ Brookes and Roscoe extended the denotational model to account also for divergence.

This collection begins with two papers on the semantic foundations of CSP. Brookes replaces the naive traces semantics with one based on actions and Roscoe extends the semantics of divergence and provides an appropriate definition of fixed point. Each paper responds to developments in theoretical computer science during the couple of decades since the 1984 and 1985 papers: the former by acknowledging work on action-based transition systems and the latter by acknowledging progress in our understanding of divergence and fixed points. Each paper provides stronger techniques whilst retaining the flavor of the original CSP.

² Autumn scene in the Golden Valley, Herefordshire, 2000.

³ JACM, 31(3):560–599, 1984.

⁴ “An Improved Failures Model for CSP,” Proc. Seminar on Concurrency, Springer, LNCS 197, 1985.

In the cleverly titled *Retracing the Semantics of CSP* Brookes argues for a traces semantics that is at once more general than that of CSP and yet retains much of the simplicity and design elegance of the original. The only cost is re-evaluation of the notion of trace, to make it action based, and imposition of a fairness condition on processes. The result is a general formalism allowing a bisimulation-type equivalence between processes that differ only in atomicity of their actions.

In *Seeing Beyond Divergence* Roscoe shows how to refine the standard denotational model of a mild extension of TCSP to reveal traces of a process, more extended than just the minimal traces, after which it may diverge. Concentrating on possible divergence, and so ignoring ‘refusals or failures’ information, he constructs a model (named *SBD* as in the title of the paper) to distinguish a process’s various opportunities to diverge — something TCSP has never done. To provide meaning to recursion in *SBD* Roscoe shows that neither greatest nor least fixed points would be correct and so he is forced to use a two-stage process whose result he calls a reflected fixed point.

A further contribution to fixed-point theory in CSP is provided by Mike Reed in his paper *Order, Topology and Recursion Induction in CSP* later in this volume.

Refinement and Simulation

The major difference between CSP and, for example, the process algebra CCS⁵ lies in the distinction each makes between processes. Whilst processes in CSP are related by refinement (one can be replaced by the other for the purpose of implementation), those in CCS are related by the finer notion of (bi)simulation.

In July 2002 a workshop was held at Microsoft Research Ltd. Cambridge to contemplate the differences and similarities between the various process algebras, with the aim of reconciling the fundamental ideas of refinement and simulation, particularly for CSP and CCS. One outcome has been the two papers in this section. Not surprisingly, established co-authors Hoare and He have produced related treatments. Each paper uses the notion of barbed traces in a treatment of process algebra in which refinement and simulation coincide. In fact a combination of the papers, which the reader will find of quite contrasting styles, might be regarded as an extra chapter for their book⁶. Use of barbed traces might be regarded as an alternative solution to the high-level plan of Brookes.

Hardware Synthesis

From its early days CSP has been closely associated with hardware design. David May provides an entertaining account of those days in *CSP, occam and transputers*, the paper of his after-dinner speech. He makes a convincing case for

⁵ Robin Milner, *Communication and Concurrency*, Prentice-Hall, 1989.

⁶ *Unifying Theories of Programming*, Prentice-Hall, 1998.

remembering our own (collective) principles as we progress and for valuing more highly the things at which we are good; and he draws potent conclusions for industry, research and education.

At the same time as *inmos*, *occam* and CSP exploited highly-synchronized communication, asynchronous hardware design was enjoying a resurgence of popularity⁷. The appropriate modification to CSP and the revised laws (thought of as being obtained by inserting unbounded buffers along channels) was undertaken by Mark Josephs whose paper *Models for Data-Flow Sequential Processes* extends that work to a wider family of processes and more sophisticated semantic models. It provides some laws and concentrates on denotational semantics.

Philips Electronics, Eindhoven, has shown a long-standing commitment to the use of formal methods and in particular to the work on asynchronous CSP. In his monumental paper *Implementation of Handshake Components*, Ad Peeters shows how CSP underlies the techniques of the established Handshake Technology developed at Philips for the design and implementation of unlocked circuits. The interest focuses on handshaking protocols that are efficient and correct in the various paradigms for unlocked design — summarized in this self-contained article. Peeters demonstrates the remarkable extent to which process algebra successfully pervades the various levels of abstraction.

Transactions

The laws satisfied by asynchronous processes communicating lazily via streams, as treated in the previous section, for example in the article by Mark Josephs, resemble those satisfied in transaction processing: a topic at the heart of applied formal methods. In fact, in his book *Communicating Sequential Processes* Hoare introduced operators to model the interrupt, checkpoint, rollback and recovery of transaction-processing systems. In this section that topic is further explored; the main concern is to maintain atomicity in a distributed system. Some treatments have attempted to do so using event refinement, the process algebra version of the data refinement of sequential programming.

But in *A Trace Semantics for Long-Running Transactions* Michael Butler, Tony Hoare and Carla Ferreira give an elegant calculus of compensations for a restriction of CSP to achieve a similar result. They adopt a traces semantics in which an action is compensable if it can subsequently be undone atomically, and presents a compositional ‘cancellation’ semantics for processes with nested interruption and compensation.

In *Practical Application of CSP and FDR to Software Design* Jonathan Lawrence acknowledges the difficulty confronting transfer of research — in this case concerning CSP — to industry and presents a case study encapsulating valuable lessons. The study centers on a recent IBM project using CSP and FDR to produce a multi-threaded connection pooling mechanism connecting a transaction-processing system to a Web server. The project spanned three days

⁷ Ivan Sutherland, “Micropipelines”, CACM, 32:720–738, 1989.

and included formal specification in CSP of the required system, validation with some degree of confidence that it captured the informal requirements, expression of the design in CSP and verification of its correctness in FDR. The result was so successful that subsequent enhancements to the delivered Java code could confidently be done by hand. Lawrence highlights the value of applied MSc's which include projects providing students with an opportunity to transfer what they have learnt on the MSc to the workplace.

Concurrent Programming

The extremely active *occam* user group continues the application of CSP begun in the work described by May in this volume to programming-language design. In *Communicating Mobile Processes* Peter Welch and Frederick Barnes introduce *occam- π* as a hybrid of *occam* and the π -calculus introduced by Milner and studied extensively in CCS. The approach is largely pragmatic, including benchmarks and the outline of applications. It is envisaged that a semantics would be denotational, following those of CSP and influenced by the π -calculus.

In *Model-Based Design of Concurrent Programs* Jeff Magee and Jeff Kramer use Label Transition Systems (LTS), a notation based on CSP, to model concurrent systems and to study their behaviours. Their approach combines clear modelling with tools that support graphical animations and systematic generation of parallel implementations in concurrent Java. Both safety checks (essentially traces properties) and liveness checks (under assumptions concerning scheduling and choice) are achieved. They conclude that such animations are useful both to students and practitioners in overcoming resistance to formal methods.

Security

One of formal methods' huge successes in the past decade has been to reasoning about security. In terms of CSP, the success has been largely due to work by Bill Roscoe et al. and Gavin Lowe (with the Caspar tool).

In *Verifying Security Protocols: an Application of CSP* Steve Schneider and Rob Delicata provide an elegant case study showing how CSP, with the notion of a rank function, can be used to reason about an authentication protocol. After proposing a putative protocol their analysis locates a flaw and verifies the correctness of a modification. In verification, the rank function is used to show that illegitimate messages do not occur. The paper is self-contained and might be used by those familiar with CSP as an introduction to this topical area.

Over the years various models of computation have been used to formalize non-interference. Typically these floundered on non-determinism, "input/output" distinctions, input totality and so forth. In *Shedding Light on Haunted Corners of Information Security* Peter Ryan outlines how process algebras, in particular CSP, can be applied to give a formal characterization of the absence of information flows through a system. Unfortunately, Peter Ryan was unable to attend due to compelling personal reasons at the last minute. Hence, only the abstract of his talk is included in this volume.

Linking Theories

Whilst security has provided one important playing field for CSP, probability has provided another. The challenge is to express and reason about distributed probabilistic algorithms using a variant of process algebra that includes a combinator for choice, with given probability, between two processes. Unacceptable attempts abound. In *Of Probabilistic wp and CSP — and Compositionality*, Carroll Morgan starts ‘afresh’ from the successful work on probabilistic sequential programming and targets process algebra via the intermediary of action systems. His translation throws up healthiness conditions for probabilistic CSP and suggests a program of work that might — finally — result in a compositional probabilistic process algebra. Incidentally his discussion of (general) compositionality using the example of eye color and the Mendelian concept of allele is a gem.

In this section is included the abstract for the talk by Mike Reed *Order, Topology and Recursion Induction in CSP* that might be thought of as a contribution to semantic foundations. He presents a recursion-induction principle that produces least fixed points for functions whose least fixed points are maximal (i.e., deterministic in the failures model of CSP). The setting is a Scott domain and the results are general enough to cover existing instances of recursion induction in CSP; in topology they are strong enough to provide answers to open questions from domain theory and point-set topology.

Automated Development, Reasoning and Model Checking

As a formal method, CSP was slow to respond to the pressure for automation. Perhaps as a result, Formal Methods’ tool FDR achieved immediate success; for instance it has played a crucial role in many of the papers in this volume. But it, and its scope, still progress as the papers by Michael Goldsmith and Ranko Lazić indicate.

In *Operational Semantics for Fun and Profit* Michael Goldsmith observes that a source of computation inefficiency in FDR is evaluation of the structured operational semantics of the operationally-presented target system (an evaluation that is necessary whenever a denotational property is to be determined). He proposes a *supercompilation* procedure to overcome it, if not in every case then at least in many. An unexpected benefit of supercompilation is transformation of a process to a form accessible to previously studied *watchdog* transformations that enable a refinement check to be recast in more efficient form.

The method of data independence allows a model-checking argument, concerning a process whose data type takes on a single value, to be extended to that process with arbitrary data value. In *On model Checking Data-Independent Systems with Arrays with Whole-Array Operations* Ranko Lazić, the originator of the technique of data independence in CSP, Tom Newcomb, and Bill Roscoe show how to extend it to programs using arrays indexed by one data-independent variable that have values from another. They obtain simple and natural conditions for decidability or undecidability of realistic questions concerning the use of such types.

For all its use, and all its appearance in this volume, FDR is far from being the only formalism for animating CSP. In the article by Magee and Kramer *Model-Based Design of Concurrent Programs* an alternative has already been demonstrated.

Industrial-Strength CSP

We have seen how CSP has been used to study theoretical aspects of concurrency and that it seems to offer yet further potential for doing so. We have seen how it has been used in hardware design, at both the implementation and design levels. And we have seen how its tools offer industrial-strength model checking. But what about the broader scope of software engineering?

In *Industrial-Strength CSP: Opportunities and Challenges in Model-Checking*, Sadie Creese demonstrates the use of FDR in reasoning about various aspects of high-integrity systems from industry, as seen from her perspective in the Systems Assurance Group within QinetiQ.

In the paper *Applied Formal Methods — from CSP to Executable Hybrid Specifications* Jan Peleska discusses his work at Verified Systems International and the University of Bremen. His case studies are drawn from an impressively realistic range, including an implementation of Byzantine agreement to provide a fault-tolerant component of the International Space Station, and the avionics controller of the Airbus A340. He discusses the difficulties involved in the production of large and complex systems. Hybrid methods become important and executability, in the form of tools available for prototyping, necessary to convince coworkers. But in the end formal methods, and in particular CSP, remainS just one of a spectrum of techniques that contribute to product quality.

Reflections!

It is not often that burgeoning areas are afforded the luxury of reflecting on both their past and futures. With the contributions contained in this volume the reader has evidence enough to decide the relevance of Gilbert Ryle's warning (*Dilemmas*, The Tarner Lectures, 1953, Cambridge University Press, digital printing 2002, page 14.)

Karl Marx was sapient enough to deny the impeachment that he was a Marxist. So too Plato was, in my view, a very unreliable Platonist. He was too much of a philosopher to think that anything that he had said was the last word. It was left to his disciples to identify his foot marks with his destination.

Ali E. Abdallah, Cliff B. Jones and Jeff W. Sanders
London, Newcastle and Oxford, January 2005

Sponsors



BCS- Formal Aspect of
Computing Science
specialist group



London South Bank
University, UK



Microsoft Research,
Cambridge, UK



Formal Methods
Europe



Handshake
Solutions, Philips,
Netherlands



Verified Systems
International, GmbH,
Germany



Formal Systems (Europe)
Limited

Table of Contents

Semantic Foundations

Retracing the Semantics of CSP <i>Stephen Brookes</i>	1
Seeing Beyond Divergence <i>A.W. Roscoe</i>	15

Refinement and Simulation

Process Algebra: A Unifying Approach <i>Tony Hoare</i>	36
Linking Theories of Concurrency <i>He Jifeng</i>	61

Hardware Synthesis

CSP, occam and Transputers <i>David May</i>	75
Models for Data-Flow Sequential Processes <i>Mark B. Josephs</i>	85
Implementation of Handshake Components <i>Ad Peeters</i>	98

Transactions

A Trace Semantics for Long-Running Transactions <i>Michael Butler, Tony Hoare, Carla Ferreira</i>	133
Practical Application of CSP and FDR to Software Design <i>Jonathan Lawrence</i>	151

Concurrent Programming

Communicating Mobile Processes <i>Peter H. Welch, Frederick R.M. Barnes</i>	175
--	-----

Model-Based Design of Concurrent Programs <i>Jeff Magee, Jeff Kramer</i>	211
---	-----

Linking Theories

Of Probabilistic <i>wp</i> and <i>CSP</i> — and Compositionality <i>Carroll Morgan</i>	220
Order, Topology, and Recursion Induction in <i>CSP</i> <i>Mike Reed</i>	242

Security

Verifying Security Protocols: An Application of CSP <i>Steve Schneider, Rob Delicata</i>	243
Shedding Light on Haunted Corners of Information Security <i>Peter Ryan</i>	264

Automated Development and Model Checking

Operational Semantics for Fun and Profit <i>Michael Goldsmith</i>	265
On Model Checking Data-Independent Systems with Arrays with Whole-Array Operations <i>Ranko Lazić, Tom Newcomb, A.W. Roscoe</i>	275

Industrial-Strength CSP

Industrial-Strength CSP: Opportunities and Challenges in Model-Checking <i>Sadie Creese</i>	292
Applied Formal Methods – From CSP to Executable Hybrid Specifications <i>Jan Peleska</i>	293

Author Index	321
---------------------------	-----