# Texts in Computer Science

*Editors*
David Gries
Fred B. Schneider

For other titles published in this series, go to
www.springer.com/series/3191

A.W. Roscoe

# Understanding Concurrent Systems

Springer

Prof. A.W. Roscoe
Oxford University Computing Laboratory
Wolfson Bldg., Parks Road
Oxford OX1 3QD
UK
Bill.Roscoe@comlab.ox.ac.uk



ISSN 1868-0941
ISBN 978-1-84882-257-3
DOI 10.1007/978-1-84882-258-0
Springer London Dordrecht Heidelberg New York

e-ISSN 1868-095X
e-ISBN 978-1-84882-258-0

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

Library of Congress Control Number: 2010936468



*Cover design*: VTEX, Vilnius

Printed on acid-free paper

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

# Preface

Since C.A.R. Hoare's text *Communicating Sequential Processes* [60] was published in 1985, his CSP notation has been extensively used for teaching and applying concurrency theory. I published a book [123] on this topic in 1997 entitled "The Theory and Practice of Concurrency". We will be referring to the latter many times in the present book, and will abbreviate it *TPC*. Hoare's book and TPC are now freely available via the web. The present book draws heavily on material from TPC: parts of the text are updated versions of sections of TPC, mainly in Part I, and we will frequently refer to material in it. However, we omit most advanced material from TPC that does not require updating. Chapter 4 and each of Chaps. 8–20 are either mainly or entirely new.

When I started writing this book, I thought of it as a much revised second edition of TPC. That remains true for the first (introductory) part, but for the rest the completed book is perhaps better described as a "sequel".

## Overview and Goals

The aim of this book is to be a comprehensive introduction and reference volume for CSP, providing the material for a number of different courses. It should also be the first point of reference for anyone wanting to use CSP or find out about its theory. It introduces other views of concurrency, using CSP to model and explain these. This book is integrated with, and uses, CSP-based tools and especially FDR to a much greater extent than TPC, and in addition describes how to create new tools based on FDR.

FDR is freely downloadable for all but commercial use. Almost all chapters either explain or illustrate the use of FDR, and there are two chapters (8 and 16) specifically about it.

This book is divided into parts on similar lines to TPC: the first three are, respectively, an introductory course on CSP, a review of the theory of CSP, and some topics on the application of CSP and its tools. Whereas TPC has extensive appendices on

the mathematical background, the CSP$_M$ language and FDR, the present book has a fourth proper part.

I have become convinced that CSP and FDR are ideal vehicles for understanding and reasoning about a wide variety of concurrent systems, not just those initially described in this language. I and others have written a number of compilers that take programs written in other notations and translate them to CSP for analysis on FDR—and most also take the output of FDR and translate it back to give feedback appropriate to the non-CSP input notation. The fourth part is devoted to this topic, while simultaneously providing an introduction to the other sorts of concurrent systems that we explain in CSP.

## Organisation and Features

### *Part I: An Introduction to CSP*

This part is designed to be used for an introductory course on CSP.

It owes much to TPC, though some of the more difficult topics in Part I of TPC have been delayed to later parts, or removed. In their place we have included some hopefully more interesting case studies.

### *Part II: Theory*

The theory of CSP has made considerable advances since 1997 in a number of directions, notably the following:

- Models have been developed [126] that do not need the sometimes controversial *divergence strictness* previously needed in all models that handle infinite behaviours.
- Our understanding of the overall hierarchy of CSP's semantic models [129, 130] has advanced: new models have been discovered, and we also know that there is no alternative to the weakest few models.
- We now have compositional models [99] for *Discrete Timed CSP*, lying between the informal *tock*-CSP language from Chap. 14 of TPC and the (continuous) *Timed CSP* of [111, 146]. Importantly, advances [98, 99] have been made in linking all these together.
- Links have been made between operational semantics in a general setting and CSP models [132]: it has thus been *proved* that one can give a semantics *in CSP* to a wide variety of languages.
- We now understand [89, 128] what properties can be specified using the (FDR-like) refinement check $F(P) \sqsubseteq G(P)$ for CSP contexts $F$ and $G$.
- Impressive theories, for example [116], have been built on mechanical theorem provers to check the properties of CSP models and semantics.

- It is necessary to change much of the structure of the algebraic semantics for CSP presented in TPC if they are to encompass some of the new models now available.
- The theory of model checking CSP (which is what FDR does) has developed substantially in a number of ways.

In this book we will discuss most of these topics, some in Part II and some in Part III.

It would be impossible to give an in-depth presentation of the theory of CSP as it now is, as TPC largely did (with the exception of Timed CSP) in 1997. In the present book we explain some basic concepts in detail and introduce the main ideas behind the advanced theory, referring the reader to the resources provided by TPC and many academic papers. Thus Part II provides the core of a textbook, building on itself by referencing these on-line resources. There are chapters on:

- Operational semantics
- The usual denotational models
- The hierarchy of models where everything is finitely observable
- The hierarchy of models that include infinite observations
- Algebraic semantics

## *Part III: Using CSP*

This part covers a number of topics in the application of CSP, and has chapters on:

- Timed modelling and analysis using the "*tock*-time" model introduced in TCP.
- The discrete modelling and verification of Timed CSP.
- More about FDR: advanced topics in the use of FDR and advanced specification techniques.
- Parameterised verifications and the state explosion problem: we introduce some techniques for coping with the exponential growth in the number of states to explore as we look at instances of networks with more processes, and techniques for proving properties of large classes of network. These include data independence, induction, and buffer tolerance.

## *Part IV: Exploring Concurrency*

Here we emphasise the ability of CSP to describe and enable reasoning about parallel systems modelled in other paradigms. There are chapters on:

- Compiling shared variable programs into CSP
- Shared variable concurrency
- Priority and mobile processes

The two chapters on shared variable concurrency introduce a new tool called SVA, created by myself and David Hopkins, as a front end for FDR. We will see that it is highly effective in revealing how shared variable programs behave.

As in TPC I have attempted, in writing Parts III and IV, to rely as little as possible on the theory presented in Part II. Therefore either of these, or various combinations of chapters and sections chosen from them, can be used as the basis of a course on concurrency to follow up Part I.

## Target Audience

This book is aimed at everyone who wants to get an in-depth understanding of concurrent systems, and will be essential reading for anyone interested in Hoare's CSP.

Part I is designed for an audience of undergraduate and Masters'-level graduate computer science students. At Oxford it is used for a second-year undergraduate course, and for both full-time and part-time M.Sc. students.

Part II is designed for people who are familiar with Part I and have fairly theoretical interests. These could be students taking an advanced course based on this material, or researchers interested in the state of the art.

Part III is intended for people who already have some experience in using CSP and FDR in practice, and want to be able to use them better or who are specifically interested in timed systems.

Part IV is designed for people who already understand CSP. They might want to understand other models of concurrent systems in terms of CSP. They might want model shared-variable, mobile or prioritised systems in CSP. Or they might want to write a translator from another language into CSP.

Most of the present book relies on no theoretical background other than a basic knowledge of sets and sequences. Some of Part II relies on a knowledge of basic partial order and metric space theory such as can be obtained by studying Appendix A of TPC. Except in Part II, I have tried to avoid making the reader follow sophisticated mathematical arguments, though this proved unavoidable in parts of Chap. 17.

Whilst I was writing this book, many people asked me to provide many examples of how to program in CSP: *design patterns*. While this is not the book of case studies that some wanted, I have tried to include enough to keep them happy. The main case studies can be found in Chaps. 4, 8, 9, 14, 15, 17, 18, 19 and 20.

## Notes to the Instructor

Chapters 1–6 (with Chap. 7 being optional) provide a comprehensive introductory course on CSP, dipping into Chap. 8 on FDR as required during the course. When deciding whether or not to include Chap. 7, the instructor should bear in mind that Chaps. 18 and 19 (on shared variables) depend heavily on sequential composition.

For an audience already familiar with CSP one could give a theory course based on Part II. Many different courses on the practical uses of CSP and FDR could be based on Chaps. 4, 8 and 14–19 of the present book and Chaps. 12–15 of TPC, and indeed there is probably enough material in Chaps. 18 and 19 on which to base a course on shared variable concurrency.

## Teaching Resources

This book has a web-site www.comlab.ox.ac.uk/ucs where you can find links to complete texts of Hoare's book [60] and TPC and links from which FDR, the ProBE CSP animator, SVA and other CSP-based tools can be down-loaded.

You can also find machine-readable $\text{CSP}_M$ versions of almost all the CSP programs in this book and in TPC, as well as overheads covering most of the material in the two books. Additionally there are practical exercises in the use of FDR that those learning this material can use, whether personally or in a course.

Further practicals and solutions to all the exercises in this book can be obtained from the author by academics using this book for teaching.

## And Finally…

When I started to write this book I assumed that it would include a chapter or two on security. In fact there is very little here on this subject, and that is mainly a summary of material in TCP. This is not because there is no new material on the mixture of CSP and security, but rather the reverse. There is now a book on security protocols via CSP [142] and a lot of additional material that goes well beyond that. It has become too large a subject to be included in a general book about CSP, at least if we want to discuss the state of the art.

While this book is focused on CSP, it covers a very wide variety of concurrent systems including combinatorial, timed, priority-based, mobile, shared variable, statecharts, buffered and asynchronous systems. Furthermore, we see how to translate several other notations into CSP. I hope, therefore, that it justifies its title *Understanding Concurrent Systems*.

Oxford, UK Bill Roscoe
June 2010

# Acknowledgements

I had the good fortune to become Tony Hoare's research student in 1978, which gave me the opportunity to work with him on the development of the 'process algebra' version of CSP and its semantics from the first.[1] I have constantly been impressed that the decisions he took in structuring the language have stood so well the twin tests of time and practical use in circumstances he could not have foreseen. The work in the present book all results, either directly or indirectly, from his vision. Those familiar with his book will recognise that much of my presentation, and many of my examples, have been influenced by it.

The core theory of CSP was developed in the late 1970s and 1980s. The two people most responsible, together with Tony and myself, for the development of the basic theoretical framework for CSP were Steve Brookes and Ernst-Rüdiger Olderog, and I am delighted to acknowledge their contributions. We were, naturally, much influenced by the work of those such as Robin Milner, Matthew Hennessy and Rocco de Nicola who were working at the same time on other process algebras.

The last few months have seen the deaths of Robin Milner and Amir Pnueli, both giants of computer science and concurrency theory in particular. A small aspect of Amir's influence can be seen in Sect. 16.4 of this book. Both were wonderful and generous men who influenced generations of researchers. They will be sadly missed.

Over the years, both CSP and my understanding of it have benefited from the work of too many people for me to list their individual contributions. I would like to thank the following present and former students, colleagues, collaborators and correspondents for their help and inspiration: Samson Abramsky, *Phil Armstrong*, Geoff Barrett, Stephen Blamey, Tilo Buschmann, *Sadie Creese*, Naiem Dathi, Jim Davies, *John Fitzgerald*, Richard Forster, Paul Gardiner, *Michael Goldsmith*, Anthony Hall, He Jifeng, *Philippa Hopcroft* (*née Broadfoot*), *David Hopkins*, *Huang Jian*, Jason Hulance, David Jackson, Lalita Jategaonkar Jagadeesan, Alan Jeffrey, Mark Josephs, Maneesh Khattri, Jonathan Lawrence, *Ranko Lazić*, *Eldar Kleiner*,

[1] An account of the development of CSP, and in particular the transition to the process algebra version of [60] from the "imperative" version of [58], can be found in [71].

*Gavin Lowe*, Helen McCarthy, Christie Marr (née Bolton), Jeremy Martin, Albert Meyer, Michael Mislove, *Nick Moffat*, Lee Momtahan, *Tom Newcomb*, *Long Nguyen*, David Nowak, *Joël Ouaknine*, *Hristina Palikareva*, Ata Parashkevov, David Park, *Sriram Rajamani*, *Joy Reed*, Mike Reed, *Jakob Rehof*, *Markus Roggenbach*, Bill Rounds, Peter Ryan, Jeff Sanders, Bryan Scattergood, Steve Schneider, Brian Scott, Karen Seidel, *Jane Sinclair*, *Antti Valmari*, *Rob van Glabbeek*, *David Walker*, *Wang Xu*, *Peter Welch*, Paul Whittaker, Jim Woodcock, James (Ben) Worrell, Zhenzhong Wu, Lars Wulf, Jay Yantchev, Irfan Zakiuddin and Zhou Chao Chen.

Many of them will recognise specific influences their work has had on my two books. Those *italicised* have had a direct influence on the new work reported in the present book.

Special thanks are due to the present and former staff of Formal Systems (some of whom are listed above) for their work in developing FDR and ProBE. The remarkable capabilities of FDR transformed my view of CSP. One of my main motivations for writing this book is to show how to use FDR effectively. Bryan Scattergood was chiefly responsible for both the design and the implementation of the ASCII version of CSP used on these and other tools. The passage of time since FDR 2 was released has only emphasised the amazing job he did in designing $\mathrm{CSP}_M$, and the huge expressive power of the embedded functional language.

In the last few years the development and maintenance of FDR have become the responsibility of my team at Oxford University Computing Laboratory, led by Phil Armstrong. I am pleased to say that this has led to many exciting developments with the tool and should soon lead to a third major release: FDR 3.

The presentation of this book has been greatly assisted by all those who have commented on and pointed out errors in drafts. In particular I would like to thank Irfan Zakiuddin, who read through the entire final draft and gave many useful comments. Wayne Wheeler and Simon Rees from Springer have given me just the right mixture of help, encouragement and cajoling.

My work on CSP, including the writing of both this book and TPC, has benefited from funding from several bodies over the years, including EPSRC, DRA, QinetiQ, ESPRIT, industry and the US Office of Naval Research. I am particularly grateful to Ralph Wachter from the last of these, without whom most of the research on CSP tools would not have happened.

This book could never have been written without the support of my wife Coby. For the third time (the previous ones being my doctoral thesis and TPC) she has read through hundreds of pages of text on a topic entirely foreign to her, expertly pointing out errors in spelling and style. And, yet again, she put up with me writing it.

# Contents