

A SCIENCE OF CONCURRENT PROGRAMS

Turing Award-winner Leslie Lamport shares the key lessons he has learned about concurrent and distributed computing over decades of writing and reasoning about their algorithms.

Algorithms are not programs, and they shouldn't be written in a programming language. Instead, this book explores how to write them and reason about them by using mathematics. It explains the principles underlying abstract programs and understanding those principles helps avoid concurrency errors. Designing an abstract program before writing any code can lead to better, more reliable programs.

The book has very few mathematical prerequisites, with an appendix summarizing the necessary knowledge. Many of the examples are available online, written in the formal language TLA^+ , and can be checked with the TLA^+ tools.

This is a fascinating read for any graduate students and researchers in theoretical computer science, concurrency, and distributed systems.

LESLIE LAMPORT was Distinguished Scientist at Microsoft Research until his retirement. Dr. Lamport won the 2013 Turing Award for “fundamental contributions to the theory and practice of distributed and concurrent systems.” He is a member of the National Academies of Science and Engineering and the American Association of Arts and Sciences.

A SCIENCE OF CONCURRENT PROGRAMS

LESLIE LAMPORT

Researcher Emeritus, Microsoft



Cambridge University Press & Assessment
978-1-009-71985-8 — A Science of Concurrent Programs
Leslie Lamport
Frontmatter
[More Information](#)



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

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/9781009719858
DOI: 10.1017/9781009719841

© Leslie Lamport 2026

This publication is in copyright. Subject to statutory exception and to the provisions
of relevant collective licensing agreements, no reproduction of any part may take place
without the written permission of Cambridge University Press & Assessment.

When citing this work, please include a reference to the DOI 10.1017/9781009719841

First published 2026

Cover image: Filo/DigitalVision Vectors/Getty Images

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

A Cataloging-in-Publication data record for this book is available from the Library of Congress

ISBN 978-1-009-71985-8 Hardback

Cambridge University Press & Assessment has no responsibility for the persistence
or accuracy of URLs for external or third-party internet websites referred to in this publication and does not
guarantee that any content on such websites is, or will
remain, accurate or appropriate.

For EU product safety concerns, contact us at Calle de José Abascal, 56, 1^o, 28003 Madrid,
Spain, or email eugpsr@cambridge.org

To Ellen

Contents

About this Book	xiii
Acknowledgments	xiv
1 Introduction	1
1.1 Who Am I?	1
1.2 Who Are You?	2
1.3 The Origin of the Science	2
1.3.1 The Origin of the Theory	2
1.3.2 The Origin of the Practice	3
1.4 Correctness	5
1.5 A Preview	6
1.6 Why Math?	9
2 Ordinary Math	12
2.1 Arithmetic as a Mathematical Theory	13
2.2 The Mathematical Theory of Algebra	14
2.3 Mathglish	17
2.4 Boolean Arithmetic (Propositional Logic)	18
2.5 ZF	21
2.6 Meaningless Expressions	23
2.7 Quantification and Bound Variables	25
2.7.1 Quantification	25
2.7.2 Bound Variables	26
2.8 Defining Mappings and Functions	28
2.8.1 Mappings	28
2.8.2 Functions	30
2.8.3 Sequences and Tuples	31
2.9 Some Useful Notation	32
2.9.1 IF/THEN/ELSE	32
2.9.2 Conjunction and Disjunction Lists	32

3	Describing Abstract Programs with Math	34
3.1	The Behavior of Physical Systems	34
3.2	Behaviors of Digital Systems	37
	Math I	37
3.2.1	From Continuous to Discrete Time	37
3.2.2	An Example: <i>Sqrs</i>	40
3.2.3	A Finer-Grained Example: <i>FGSqrs</i>	44
3.3	Nondeterminism	46
	Math II	46
3.3.1	Sources of Nondeterminism	47
3.3.2	An Example: <i>Increment</i>	48
3.4	Temporal Logic	52
	Math III	52
3.4.1	The Logic of Actions	53
	3.4.1.1 Eliminating State Numbers	53
	3.4.1.2 The Semantics of the Logic of Actions	55
	3.4.1.3 The Prime Operator	56
	3.4.1.4 Action Composition	58
3.4.2	The Temporal Logic RTL _A	59
	3.4.2.1 Simple RTL _A	60
	3.4.2.2 The Complete RTL _A	62
	3.4.2.3 The \square Operator	63
	3.4.2.4 Eventually (\diamond)	64
	3.4.2.5 Eventually Always ($\diamond\square$)	66
	3.4.2.6 Infinitely Often ($\square\diamond$)	66
	3.4.2.7 The End of the Line	67
	3.4.2.8 Leads To (\rightsquigarrow)	68
	3.4.2.9 Warning	69
3.5	TLA	70
	Math IV	70
3.5.1	The Problem	71
3.5.2	The Solution	72
3.5.3	Stuttering Insensitivity	74
3.5.4	The Definition of TLA	76
4	Safety, Liveness, and Fairness	79
4.1	Safety and Liveness	79
	Math V	79
4.1.1	Definitions	81
4.1.2	A Completeness Theorem	82

CONTENTS

ix

4.1.3	The Operator \mathcal{C}	84
4.1.4	What Good is Liveness?	85
4.2	Fairness	86
4.2.1	Traditional Programs and Enabled	86
4.2.2	Concurrent Programs	87
4.2.2.1	Mutual Exclusion	88
4.2.2.2	Machine Closure	92
4.2.3	Weak Fairness	93
4.2.4	Temporal Logic Reasoning	96
4.2.5	Reasoning With Weak Fairness	97
4.2.5.1	Liveness for Mutual Exclusion	97
4.2.5.2	The One-Bit Algorithm	98
4.2.5.3	Proving Liveness	100
4.2.6	Strong Fairness	104
4.2.6.1	Starvation Free Mutual Exclusion	104
4.2.6.2	The Definition of Strong Fairness	105
4.2.6.3	Using a Strongly Fair Semaphore	106
4.2.7	Properties of WF and SF	107
4.2.8	What is Fairness?	109
5	Interlude	111
5.1	Possibility and Accuracy	111
5.1.1	Possibility Conditions	111
5.1.2	Expressing Possibility in TLA	112
5.1.3	Checking Accuracy	114
5.2	Real-Time Programs	115
	Math VI	116
5.2.1	Fischer's Algorithm	117
5.2.2	Correctness of Fischer's Algorithm	120
5.2.3	Fairness and Zeno Behaviors	121
5.2.4	Discrete Time	123
5.2.5	Hybrid Systems	125
6	Refinement	126
6.1	A Sequential Algorithm	128
	Math VII	128
6.1.1	A One-Step Program	129
6.1.2	Two Views of Refinement Mappings	131
6.1.3	A Step and Data Refinement	132
6.2	Invariance Under Refinement	135

6.3	An Example: The Paxos Algorithm	135
6.3.1	The Consensus Problem	136
6.3.2	The Paxos Consensus Algorithm	138
6.3.2.1	The Specification of Consensus	139
6.3.2.2	The Voting Algorithm	139
6.3.2.3	The Paxos Abstract Program	142
6.3.3	Implementing Paxos	144
6.4	Proving Refinement	145
	Math VIII	146
6.4.1	The Refinement Mapping	148
6.4.2	Refinement of Safety	149
6.4.3	Refinement of Fairness	152
6.4.4	A Closer Look at \mathbb{E}	155
6.4.4.1	A Syntactic View	155
6.4.4.2	Computing \mathbb{E}	156
6.4.4.3	The Trouble With \mathbb{E}	158
6.5	A Warning	161
7	Auxiliary Variables	163
7.1	Variable Hiding	163
	Math IX	163
7.1.1	Introduction	164
7.1.2	Reasoning About \exists	166
7.1.3	The Definition of \exists	167
7.2	History Variables	169
7.2.1	How to Add a History Variable	169
7.2.2	History Variables and Fairness	173
7.2.3	A Completeness Result for History Variables	174
7.3	Stuttering Variables	175
	Math X	175
7.3.1	The Example	176
7.3.2	Adding Stuttering Steps After an Action	178
7.3.3	Adding Stuttering Steps Before an Action	182
7.3.4	Fairness and Stuttering Variables	183
7.3.5	Infinite-Stuttering Variables	186
7.4	Prophecy Variables	187
	Math XI	187
7.4.1	Simple Prophecy Variables	188
7.4.2	Predicting the Impossible and Liveness	192
7.4.3	General Prophecy Variables	193

CONTENTS

xi

7.4.3.1	A Sequence of Prophecies	194
7.4.3.2	A Set of Prophecies	197
7.4.3.3	Further Generalizations	200
7.5	The Existence of Refinement Mappings	202
7.6	The FIFO Queue	203
7.6.1	<i>Fifo</i> – A Linearizable Specification	203
7.6.2	<i>POFifo</i> – A More General Specification	205
7.6.2.1	The Background	205
7.6.2.2	Program <i>POFifo</i>	207
7.6.3	Showing <i>IPOFifo</i> Implements <i>Fifo</i>	211
7.6.3.1	The Prophecy Variable	212
7.6.3.2	The History Variable $qBar$	213
7.6.3.3	Stuttering and the Refinement Mapping	216
7.7	Prophecy Constants	217
8	Loose Ends	220
8.1	Reduction	220
8.1.1	Introduction	220
8.1.1.1	What Reduction Is	220
8.1.1.2	The TLA Approach	222
8.1.2	An Example	224
8.1.2.1	The Reduced Behaviors	224
8.1.2.2	The Program $S \otimes S^R$	226
8.1.2.3	The Invariant	230
8.1.3	Reduction In General	232
8.1.4	The Hypothesis $\Box \Diamond \neg \mathcal{L}$	236
8.1.5	Adding Liveness	236
8.1.5.1	Fairness of Subactions of E^R	237
8.1.5.2	Fairness of Subactions of M^R	240
8.1.5.3	The Reduction Theorem with Fairness	243
8.1.6	An Example: Making Critical Sections Atomic	244
8.1.7	Another Example: Pipelining	247
8.2	Decomposing and Composing Programs	251
8.2.1	Decomposing Programs	253
8.2.1.1	Writing a Program as a Conjunction	253
8.2.1.2	Decomposing Proofs	255
8.2.2	Composing Components	261

Appendix	266
A Miscellany	266
A.1 Ordinary Math Summary	266
A.1.1 Arithmetic	266
A.1.2 Propositional Logic	266
A.1.3 Predicate Logic	267
A.1.4 Sets	267
A.1.5 The CHOOSE Operator	268
A.1.6 Functions	269
A.1.7 Sequences	269
A.1.8 Notation	270
A.1.9 Recursive Definitions	270
A.2 Structured Proofs	271
A.3 Why Not All Mappings Are Sets	274
A.4 How Not to Write x'''	275
A.5 Hoare Logic	276
A.6 Another Way to Look at Safety and Liveness	279
A.6.1 Metric Spaces	279
A.6.2 The Metric Space of Behaviors	282
B Proofs	284
B.1 Invariance Proof of <i>Increment</i>	284
B.2 Proof of Theorem 4.3	287
B.3 Proof of Theorem 4.4	289
B.4 Proof of Theorem 4.5	290
B.5 Proof of Theorem 4.6	291
B.6 Proof of Theorem 4.7	291
B.7 Proof of Theorem 4.8	293
B.8 Proof Sketch of Theorem 4.9	295
B.9 Proof of Theorem 7.2	296
B.10 Proof Sketch of Theorem 7.3	298
B.11 Proof Sketch of Theorem 7.6	299
B.12 Proof of Theorem 8.2	302
B.13 Proof of Theorem 8.3	302
Bibliography	304
Index	310

CONTENTS

xiii

About this Book

The first chapter, which is a little over ten pages, explains what this book is about. I believe it will tell you whether you should read the book. If you decide to read it, here are some things you should know.

The book's science is embodied in a language called TLA⁺. Many of its examples have been written in TLA⁺ and checked with the TLA⁺ tools. However, TLA⁺ is not used here because the requirements of handling industrial applications require it to be a little more complicated than is necessary for the book. The TLA⁺ versions of the examples and an explanation of how to translate from the book's notation to TLA⁺ will be available at <https://www.cambridge.org/9781009719858>.

The book has no formal exercises, but the text proposes a number of problems for you to solve that can help you learn the material. To learn how the science can be put into practice, you should try writing your own examples in TLA⁺, checking what you do with the TLA⁺ tools.

Acknowledgments

This book presents things that I have learned over decades. During that time, I have discussed them with many wonderful colleagues. I learned many things from them, and I cannot possibly remember who taught me what. Here is a list containing only my coauthors of published papers who I can remember contributing to the content of this book. Omitted are a number of colleagues who taught me a lot that does not appear in the book, and a few who taught me things contained here but with whom I never wrote a published paper.

Martín Abadi, Krzysztof Apt, Brannon Batson, Manfred Broy, K. Mani Chandy, Ernie Cohen, Edsger Dijkstra, Damien Doligez, Urban Engberg, Georges Gonthier, Jim Gray, Peter Grønning, Markus Kuppe, Robert Kurshan, Peter Ladkin, David Langworthy, Nancy Lynch, Michael Melliar-Smith, Stephan Merz, Susan Owicki, Richard Palais, Fred Schneider, Carel Scholten, Robert Shostak, Mark Tuttle, Friedrich H. Vogt, and Yuan Yu.

The following people not in the list above read preliminary versions of the book and reported errors or sent me comments that led to significant changes.

Petros Angelatos, Paul Banks, Sergey Bronnikov, Siegfried Bublit, Hugo Sanz González, Bruno Grenet, Matthias Grundmann, Rob Hagemans, Pedro de las Heras Quirós, Lorin Hochstein, Dave Hughes, Johannes Laire, Tom Moertel, Ron Pressler, Chris Newcombe, Divyanshu Ranjan, Michael Roeleveld, Mark Rogers, Scarlet Schwiderski-Grosche, Peter Sovietov, Alexander Vasilev, and Ugur Y. Yavuz.