
PARALLEL EXECUTION OF LOGIC PROGRAMS

**THE KLUWER INTERNATIONAL SERIES
IN ENGINEERING AND COMPUTER SCIENCE**

**PARALLEL PROCESSING
AND FIFTH GENERATION COMPUTING**

Consulting Editor

Doug DeGroot

PARALLEL EXECUTION OF LOGIC PROGRAMS

by

John S. Conery
University of Oregon

WKAP ARCHIEF



KLUWER ACADEMIC PUBLISHERS
Boston/Dordrecht/Lancaster

Distributors for North America:

Kluwer Academic Publishers
101 Philip Drive
Assinippi Park
Norwell, Massachusetts 02061, USA

Distributors for the UK and Ireland:

Kluwer Academic Publishers
MTP Press Limited
Falcon House, Queen Square
Lancaster LA1 1RN, UNITED KINGDOM

Distributors for all other countries:

Kluwer Academic Publishers Group
Distribution Centre
Post Office Box 322
3300 AH Dordrecht, THE NETHERLANDS

Library of Congress Cataloging-in-Publication Data

Conery, John S.

Parallel execution of logic programming.

(The Kluwer international series in engineering and
computer science ; SECS 25)

Revision of the author's thesis (Ph. D.—University
of Oregon) originally presented under the title: The
AND/OR process model for parallel interpretation of logic
programs.

Bibliography: p.

Includes index.

1. Logic programming. 2. Parallel processing
(Electronic computers) I. Title. II. Series.

QA76.6.C644 1987 004'.35 86-27838

ISBN-13: 978-1-4612-9187-9

e-ISBN-13: 978-1-4613-1987-0

DOI: 10.1007/978-1-4613-1987-0

Copyright © 1987 by Kluwer Academic Publishers

Softcover reprint of the hardcover 1st edition 1987

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher, Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 02061.

This book was formatted by the author at the
University of Oregon, using the $\text{\TeX}$ document
preparation system (by D. E. Knuth) and $\text{\LaTeX}$
macro package (L. Lamport).

For Leslie ♡

Contents

List of Figures	xi
Preface	xiii
1 Introduction	3
2 Logic Programming	7
2.1 Syntax	8
2.2 Semantics	12
2.3 Control	16
2.4 Prolog	19
2.4.1 Evaluable Predicates and Arithmetic	19
2.4.2 Higher Order Functions	21
2.4.3 The Cut Symbol	22
2.5 Alternate Control Strategies	26
2.5.1 Selection by Number of Solutions	27
2.5.2 Selection by Number of Uninstantiated Variables	28
2.5.3 Intelligent Backtracking	29
2.5.4 Coroutines	30
2.6 Chapter Summary	33
3 Parallelism in Logic Programs	35
3.1 Models for OR Parallelism	37
3.1.1 Pure OR Parallelism	39
3.1.2 OR Processes	42
3.1.3 Distributed Search	45
3.1.4 Summary	47
3.2 Models for AND Parallelism	48
3.2.1 Stream Parallel Models	48
3.2.2 AND Processes	52
3.2.3 AND Parallelism in the Goal Tree	56

3.2.4	Summary	56
3.3	Low Level Parallelism	57
3.4	Chapter Summary	59
4	The AND/OR Process Model	63
4.1	Oracle	64
4.2	Messages	65
4.3	OR Processes	66
4.4	AND Processes	67
4.5	Interpreter	68
4.6	Programming Language	70
4.7	Chapter Summary	71
5	Parallel OR Processes	73
5.1	Operating Modes	73
5.2	Execution	74
5.3	Example	75
5.4	Chapter Summary	80
6	Parallel AND Processes	83
6.1	Ordering of Literals	84
6.1.1	Dataflow Graphs	85
6.1.2	The Ordering Algorithm	86
6.1.3	Examples	88
6.2	Forward Execution	93
6.2.1	Forward Execution Algorithm	93
6.2.2	Solution of a Deterministic Function	95
6.3	Backward Execution	98
6.3.1	Generating Tuples of Terms	98
6.3.2	Definitions for Backward Execution	99
6.3.3	The Backward Execution Algorithm	100
6.4	Detailed Example	104
6.4.1	Ordering	104
6.4.2	Forward Execution	106
6.4.3	Backward Execution	107
6.4.4	Additional Solutions	108
6.5	Discussion	111
6.5.1	Relative Order of Incoming Messages	111
6.5.2	Definition of Candidate Set	112
6.5.3	Result Cache	113
6.5.4	Infinite Domains	115
6.5.5	Multisets of Results	116
6.6	Chapter Summary	118

7	Implementation	119
7.1	Overview of the Interpreter	120
7.2	Parallel AND Processes	121
7.3	Process Allocation	126
7.4	Growth Control	128
7.4.1	Conditional Expressions	128
7.4.2	Process Priorities	129
7.4.3	Message Protocols	129
7.4.4	Secondary Memory	130
7.5	Summary	131
	Bibliography	133
	Index	143

List of Figures

1.1	Layers of Abstraction	4
2.1	Examples of Clauses	9
2.2	An Example of a Logic Program	10
2.3	Examples of Resolution	15
2.4	Goal Tree	18
2.5	Examples of <code>is</code> in DEC-10 Prolog	20
2.6	The Effect of “Cut”	23
2.7	Coroutine vs. Depth-First Control	31
3.1	AND Parallelism vs. OR Parallelism	36
3.2	Environment Stack for Sequential Prolog	38
3.3	Environment Stack in OR-Parallel System	39
3.4	OR Processes	42
3.5	Search Parallelism	45
3.6	A Goal Statement as a Network of Processes	49
3.7	AND Processes	53
3.8	Duplicate Computations in a Goal Tree	60
3.9	Combined AND and OR Parallelism	61
4.1	Sample Interpreter Output	69
5.1	Modes of an OR Process	74
5.2	Starting an OR Process	76
5.3	State Transitions of an OR Process	77
5.4	States of a Parallel OR Process	78
6.1	The Literal Ordering Algorithm	87
6.2	Graph for Disjoint Subgoals	89
6.3	Graph for Shared Variables	90
6.4	Graph for Deterministic Function	91
6.5	Graph for Map Coloring	92

6.6	Forward Execution Algorithm	94
6.7	Sample Graph Reductions	96
6.8	Program for Matrix Multiplication	97
6.9	Backward Execution Algorithm	101
6.10	Maintaining a Cache of Results	103
6.11	Graph for Detailed Example	105
6.12	States of a Parallel AND Process	106
6.13	Candidate Sets	112
6.14	The Effect of Goal Caching on Recomputation	114
7.1	Set Operations in Backward Execution	122
7.2	Map Coloring Program	125

Preface

This book is an updated version of my Ph.D. dissertation, *The AND/OR Process Model for Parallel Interpretation of Logic Programs*. The three years since that paper was finished (or so I thought then) have seen quite a bit of work in the area of parallel execution models and programming languages for logic programs. A quick glance at the bibliography here shows roughly 50 papers on these topics, 40 of which were published after 1983. The main difference between the book and the dissertation is the updated survey of related work.

One of the appendices in the dissertation was an overview of a Prolog implementation of an interpreter based on the AND/OR Process Model, a simulator I used to get some preliminary measurements of parallelism in logic programs. In the last three years I have been involved with three other implementations. One was written in C and is now being installed on a small multiprocessor at the University of Oregon. Most of the programming of this interpreter was done by Nitin More under my direction for his M.S. project. The other two, one written in Multilisp and the other in Modula-2, are more limited, intended to test ideas about implementing specific aspects of the model. Instead of an appendix describing one interpreter, this book has more detail about implementation included in Chapters 5 through 7, based on a combination of ideas from the four interpreters.

One of the implementation methods is an algorithm for generating multiple results in nondeterministic programs during parallel execution. The algorithm in the dissertation had a flaw, in that it fails to generate all possible results in certain situations. The flaw was first pointed out to me by Jung-Herng Chang. Also, N. S. Woo and K-M Choe were kind enough to send me a preprint of their improved algorithm. The algorithm presented here in Chapter 6 does not have the same flaw. It was developed in the context of the Modula-2 implementation of parallel AND processes, and differs from the other two in several respects.

Other than the updated survey and the incorporation of recent implementation methods into the main text, and some minor changes in presentation that will hopefully make some topics more clear, this work is basically the

same as the dissertation.

I would again like to thank my Ph.D. advisor, Dennis Kibler, for his outstanding support and encouragement during my career as a graduate student. Also, I still remember the thoughtful comments of Bruce Porter, Paul Morris, Jim Neighbors, and Steve Fickas. More recently, Paul Bloch, Dave Meyer, Nitin More, Don Pate, and Tsyuoshi Shinogi have all made contributions to the continuing project and the book. I have also benefited from my association with Al Davis and his group at Schlumberger Palo Alto Research, Gary Lindstrom, and Doug DeGroot. In fact, Doug is mostly responsible for the existence of this book. The Department of Computer and Information Science at the University of Oregon provided the resources to format and print the laserscript and to carry out the research described in the last chapter.

Finally, I would like to express my appreciation to my wife Leslie; her love, patience, support, and interest have made this task much easier than it otherwise would have been.

JC