

Lecture Notes in Computer Science

586

Edited by G. Goos and J. Hartmanis

Advisory Board: W. Brauer D. Gries J. Stoer



A. Cheese

Parallel Execution of Parlog

Springer-Verlag

Berlin Heidelberg New York

London Paris Tokyo

Hong Kong Barcelona

Budapest

Series Editors

Gerhard Goos
Universität Karlsruhe
Postfach 69 80
Vincenz-Priessnitz-Straße 1
W-7500 Karlsruhe, FRG

Juris Hartmanis
Department of Computer Science
Cornell University
5149 Upson Hall
Ithaca, NY 14853, USA

Author

Andrew Cheese
Siemens-Nixdorf Information Systems AG, Multiprocessor Unix Kernel Group
Otto Hahn Ring 6, W-8000 Munich 83, FRG

CR Subject Classification (1991): D.3.4

ISBN 3-540-55382-7 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-55382-7 Springer-Verlag New York Berlin Heidelberg

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-Verlag. Violations are liable for prosecution under the German Copyright Law.

© Springer-Verlag Berlin Heidelberg 1992
Printed in Germany

Typesetting: Camera ready by author
Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr.
45/3140-543210 - Printed on acid-free paper

Preface

Logic programming has been proposed as a programming methodology that may help in producing more reliable and maintainable computing systems than those presently in service. At the same time there has been a trend for making faster computers by building machines using multiple CPUs. As a result of these two factors a family of concurrent logic languages has been developed to tackle the problems of programming these parallel computer architectures. It is important that the implementations of such computer languages should be efficient, otherwise the benefits of parallelism will be lost.

This monograph concentrates on the programming language Parlog and on computational models for its efficient execution. Two such models are developed, one a fine-grain Packet-Rewriting model and the other more coarse-grained, the Multi-Sequential model. Both models are reviewed in detail and software simulators have been built for them. Results from the simulations show that the Multi-Sequential model is very promising whereas the Packet-Rewriting model does not appear to be suitable for the efficient execution of logic languages. These results have considerable importance for the design of parallel logic programming systems, and the implications are outlined and discussed in the concluding chapter.

I have many people to thank for helping me with this research. First and most of all, I would like to thank my supervisor, David Brailsford for all his help and encouragement during my research.

I am very grateful to the following people for proof-reading sections, and in some cases all, of this monograph and for their comments : Uri Baron, Steve Benford, David Brailsford, Mark O'Brien, Sergio Delgado-Rannau, Dave Elliman, Colin Higgins, Graem Ringwood, Andy Walker, and Marion Windsor.

Marion Windsor, the secretary of the Department of Computer Science, University of Nottingham, was particularly helpful throughout the course of this research.

I would like to thank the following members of the Department of Computing Science at the University of Nottingham for providing a friendly and stimulating working environment, William Armitage, Peter Cowan, David Evans, David Ford, Eric Foxley, Godwin Gwei, Leon Harrison, Mike Heard, Roger Henry, Phillipa Hennessey, Emiko Hiraga, Kevin Hopkins, Anne Lomax, Graeme Lunt, Julian Onions, William Shu, Hugh Smith, and Mary Tolley.

Special thanks are due to Simone Mahlenbrei who helped me at various times whilst I was putting the finishing touches to this book at the European Computer-Industry Research Centre GmbH (ECRC).

This research was funded by the UK Science and Engineering Research Council (SERC).

February 1992

Andrew Cheese

Contents

Chapter 1 Introduction

Background 1

Logic Programming 4

Developments in Prolog Implementation 9

Computer Architecture Developments 12

Parallelism in Logic Programs 12

Concurrent Logic Programming 14

Objectives and Contributions of this Research 24

Preview of Book Contents 25

Chapter 2 Parlog A Concurrent Logic Programming Language

Introduction 27

Concurrency 27

Inter-Process Communication 27

Indeterminacy 29

Synchronization 30

Other Parlog Syntax and Operational Features 31

Example Programs 33

Compilation 35

Chosen Dialect 45

Chapter 3 A Fine-Grain Graph Reduction Model of Computation

Introduction 48

Graph Reduction 49

The Computational Model 51

Nature of Packets 52

Packet Structure	52
Operational Semantics of the Model	54
Sharing of Computation	56
Packet Description Language	57
An Example	59
Selectors and Constructors	63
Remarks on the Model of Computation	64

Chapter 4 Implementing Parlog on a Packet-Rewriting Computational Model

Introduction	65
The Implementation	65
Throttling	89
Evaluation	89
Summary	94

Chapter 5 The Multi-Sequential Coarse-Grain Approach

Multi-Sequential Architectures	95
Code Space	96
Data Space	96
Processing Element Structure	100
Environments	101
Task Data Structures	102
Control Data Structures	103
Management of Queue Data Structures	106
Load Balancing	109

Recovery from Resource Exhaustion	111
Abstract Instruction Set	112
Simulation of Model	120
Summary	132
 Chapter 6 Summary, Further Work and Conclusions	
Introduction	133
The Packet-Rewriting Model	133
The Multi-Sequential Model	136
Comparison of the Packet-Rewriting and Multi-Sequential Models	138
Further Work	140
Conclusions	144
 Appendix 1 Fine-Grain Execution of merge/3	147
 Appendix 2 A Physical Bit-Level Packet Representation	157
 Appendix 3 PPM Instruction Set Listing	159
 Appendix 4 Compiled Form of merge/3 for PPM	161
 Bibliography	165