Lecture Notes in Computer Science 556
Edited by G. Goos and J. Hartmanis

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

J.-M. Jacquet

Conclog: A Methodological Approach to Concurrent Logic Programming

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
Upson Hall
Ithaca, NY 14853, USA

Author

Jean-Marie Jacquet
Centre for Mathematics and Computer Science
Kruislaan 413, 1098 SJ Amsterdam, The Netherlands
and
Department of Computer Science, University of Namur
Rue Grandgagnage 21, 5000 Namur, Belgium

CR Subject Classification (1991): D.1.3, D.2.10, D.3.1, I.2.3, I.2.5

ISBN 3-540-54938-2 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-54938-2 Springer-Verlag New York Berlin Heidelberg

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

To my parents
who began this whole work

Preface

Content

As suggested by the intensive activity in logic programming both in the research and the application areas, the use of logic as a programming language is now widely accepted. Besides, the recent development of parallel architectures of computers has strengthened the interest for concurrent computations. Concurrent logic programming has emerged from these two fields of activities. It seems particularly promising because of its declarative and symbolic appeals and because of its inherent non-deterministic feature, which makes it very well suited for parallel executions. This book is a step in the study of concurrent logic programming. It discusses the design of a concurrent logic programming language, named Conclog, and a methodology for constructing programs in this language.

(i) The language

The main features of Conclog originate from the approach adopted in its design. Instead of directly dealing with operational tricks to ensure efficiency, the ideal logic programming paradigm has been taken as a reference. A sound and complete parallel execution model of Horn clauses is described first. It uses or-parallelism and full and-parallelism. Subgoals of conjunctions are reduced independently, regardless of shared variables. Subsequent conflicting bindings are reconciled intermittently by equation manipulations. This basic scheme is then extended to incorporate negation. To that effect, substitutions are generalized and inequations are introduced in the reconciliation process. The extended scheme turns out to be sound and as complete as possible when the negation-as failure-rule and the resolution principle are used. Finally, extra-logical features are defined for optimization and practicability purposes.

We claim that the resulting language allows an easy declarative programming. As a consequence of our concern of computing general Horn clauses in a sound and as complete as possible way, problems can be solved in a declarative manner. In particular, annotation handling is not required to obtain soundness and completeness. Furthermore, multi-directional and multi-solution procedures are supported quite naturally. Nevertheless, efficiency has also been taken into account. Assuming suitable hypotheses on their use, procedures can be

transformed into very efficient ones through the introduction of appropriate control information.

Since our main concern is the ease of programming, extra-logical features are kept minimal and have a simple semantics. These properties are not obtained at the expense of generality. Examples ranging from pure logical to classical concurrent applications are developed to support our claim.

(ii) The methodology

The proposed methodology is aimed at guiding the construction of a program in Conclog in a rigorous way. Although Conclog has been taken as target language, we believe that it can be easily adapted to other concurrent logic languages.

The methodology covers the entire programming process, from informal specifications to efficient Conclog programs. It is based on three phases. The first phase is dedicated to writing specifications. A specification consists essentially of a structured description of the computed relation in natural language. Necessary types, environment conditions and operational properties may be specified as well. The second phase consists in constructing a description in (pure) first order logic. Such a description is composed of two parts: a definition of predicates and a set of properties relating them. The last phase consists in deriving Conclog programs from the logic description. It proceeds in two steps. A first correct program is derived from the logic description. It is then transformed into more efficient versions by means of correctness-preserving transformations.

Structure

The book is structured in twelve chapters organised in four parts: an introductory part, a concluding part and two main parts discussing the design of Conclog and programming in it.

(i) The introduction part

The introductory part is composed of two chapters. Chapter 1 presents the goals of the book and places it in the concurrent logic programming context. Chapter 2 provides the background material necessary to understand it.

(ii) The design part

The design of the language is described in six chapters. Chapter 3 presents the auxiliary reconciliation calculus. Chapter 4 discusses the design of the parallel execution model of Horn clauses. Chapter 5 extends it to tackle negation. Chapter 6 describes the Conclog extra-logical features. Chapter 7 examines a possible extension of the resulting model. Chapter 8 compares Conclog with other parallel schemes and concurrent logic languages.

(iii) The programming part

Three chapters are devoted to programming in Conclog. Chapter 9 presents the methodology. Chapter 10 and Chapter 11 apply it to concrete examples ranging from applications free from behavioral requirements (Chapter 10) to the simulation of dynamic systems (Chapter 11).

(iv) The conclusion part

Finally, the conclusion part, composed only of Chapter 12, sums up our work and suggests subjects for future research.

Guide to the reader

Conclog has been designed in full details. Theoretical properties have been proved too. It follows that some parts are necessarily specific and/or theoretical. They may discourage the reader interested in using Conclog or interested in having just an overview of it. To this end, the chapters of the book have been conceived in order to allow some parts to be skipped without any trouble. In a first reading, we suggest that the reader consult the introductory Sections 3.1.1, 3.2.1, 4.1, 5.1, 6.1 and Section 6.2. He should get enough information to understand the language and to use it.

Similarly, Chapters 10 and 11 have been written in such a way that a minimal knowledge of the methodology is necessary to understand them. The reader just interested in the direct coding of programs can thus just read the introductory Section 9.1.

Our methodology does not intend to be a universal panacea. Nevertheless, it aims at easing the construction of the programs. We thus warmly recommend Chapter 9 explaining it.

Finally, this book has been conceived to be as self-contained as possible. The reader unfamiliar with logic programming can read it without first consulting other books. Nevertheless, we have been quite concise in recalling the basic foundations of logic programming. References are given where the reader can find complementary information, if necessary.

Acknowledgments

This book is a revised version of my Ph.D. thesis ([Jacquet, 1989]), accepted by the University of Namur (Facultés Universitaires Notre-Dame de la Paix de Namur), Belgium, in November 1989. Most of the work presented here was carried out when I was there as a Research Assistant supported by the Belgian National Fund for Scientific Research. The final version was written when I was participating in the ESPRIT Project Integration at the Centre for Mathematics and Computer Science (CWI) in Amsterdam, The Netherlands. I would like to thank all these institutions for having supported my work and for having provided me with

optimal facilities to achieve it. It also gives me great pleasure to thank the people who so ably helped me in writing this book.

I am indebted to Axel van Lamsweerde, my supervisor. He introduced me to logic programming several years ago and initiated my research in concurrent logic programming. His careful reading of earlier versions has substantially improved the quality of this work.

I wish to acknowledge Yves Deville, my office mate at the University of Namur, as well. Much of the material presented subsequently has been influenced by our numerous discussions. I am also pleased to acknowledge Baudouin Le Charlier for his pertinent remarks.

I am grateful to Ugo Montanari for his interest in my work. His comments and advice on my Ph.D. thesis have been very useful for its publication. I am also grateful to Maurice Bruynooghe and Jean Fichefet for having served on my Ph.D. thesis committee. Their comments on the draft version of the thesis have also been of great aid.

I wish to thank the members of the ESPRIT Project Integration and especially Krzysztof Apt, Jaco de Bakker, Keith Clark, Frank Mac Cabe, Luis Monteiro, Catuscia Palamidessi, Antonio Porto, Jan Rutten, for their interest in my work and helpful discussions.

Several members of the Institut d'Informatique of the University of Namur, of the Ecole de Langues Vivantes of the University of Namur and of the Centre for Mathematics and Computer Science in Amsterdam were so kind to read parts or previous versions of this book: Jorge Barreto, Mete Celiktin, Pierre De Boek, Guy Deville, Pierre Flener, Naji Habra, Rosane Pagano, Daniele Turi, Fer-Jan de Vries, Jeroen Warmerdam. I take this opportunity to thank them. I am also particularly indebted to Dominique Adams and Anne Collard who carefully checked each word of this book. All remaining faults are, of course, mine.

I am also grateful to Springer-Verlag, in particular to Alfred Hofmann and Hans Wössner, for their help in getting the manuscript published.

Last but not least, special thanks are due to my family. Their permanent support throughout these years substantially contributed to the completion of this book. The interest of friends and colleagues has also been much appreciated. May they all find my gratitude here.

September 1991 *J.-M. Jacquet*

Table of contents