# Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

## 225

# Third International Conference on Logic Programming

Imperial College of Science and Technology, London, United Kingdom, July 14–18, 1986 Proceedings

Edited by Ehud Shapiro



Springer-Verlag Berlin Heidelberg New York London Paris Tokyo **Editorial Board** 

D. Barstow W. Brauer P. Brinch Hansen D. Gries D. Luckham C. Moler A. Pnueli G. Seegmüller J. Stoer N. Wirth

### Editor

Ehud Shapiro Department of Computer Science The Weizmann Institute of Science Rehovot 76100, Israel

4517

CR Subject Classifications (1985): D.1, D.3, F.1, F.3, I.2

ISBN 3-540-16492-8 Springer-Verlag Berlin Heidelberg New York ISBN 0-387-16492-8 Springer-Verlag New York Heidelberg Berlin

ISBN 0-387-16492-8 (U.S.)

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically those of translation, reprinting, re-use of illustrations, broadcasting, reproduction by photocopying machine or similar means, and storage in data banks. Under § 54 of the German Copyright Law where copies are made for other than private use, a fee is payable to "Verwertungsgesellschaft Wort", Munich.

© Springer-Verlag Berlin Heidelberg 1986. Frinted in Germany

Printing and binding: Beltz Offsetdruck, Hemsbach/Bergstr. 2145/3140-543210

### Foreword

This is the report on the proceedings of the Third International Conference on Logic Programming, held on July 14-18, 1986, at Imperial College of Science and Technology. The two previous conferences took place in Uppsala, Sweden, in 1984, and in Marseille, France, in 1982.

Around 140 papers were submitted to the conference. They were refereed by the members of the program committee and by external referees, who are listed below. It is a pleasure to thank the authors who responded to the call for papers. Unfortunately, only 56 could be accepted, out of which 54 are included in this volume. In addition, seven speakers have responded to our invitation to lecture at the conference: K. Fuchi (keynote speaker), J. McCarthy (banquet speaker), and T. Chikayama, J.L. Lassez, M. McCord, A. Takeuchi, and J.D. Ullman. Papers by the invited speakers (except for the banquet speaker) are also included.

I would like to thank the program committee members, who deliberated in an attempt to provide a high-quality and balanced program, the referees who reviewed several papers each under a short schedule, and to Sarah Fliegelmann, Michael Codish, Michael Hirsch, and Steve Taylor for helping with the management of the refereeing procedure. Thanks to John Conery for the Prolog programs for maintaining the submissions database.

Rehovot, April 1986

Ehud Shapiro

### General Chairman

Keith Clark, Imperial College, U.K.

### **Program Committee**

Michel van Caneghem, University of Marseille-Aix, France Keith Clark, Imperial College, U.K. Veronica Dahl, Simon Fraser University, Canada Maarten van Einden, University of Waterloo, Canada Kazuhiro Fuchi, ICOT, Japan Koichi Furukawa, ICOT, Japan Ake Hansson, Uppsala University, Sweden Kenneth M. Kahn, Xerox PARC, U.S.A. Peter Koves, Logicware Inc., Canada Giorgio Levi, University of Pisa, Italy John Lloyd, University of Melbourne, Australia Frank G. McCabe, Imperial College, U.K. Jack Minker, Maryland University, U.S.A. David H.D. Warren, Manchester University, U.K. Antonio Porto, University of Lisbon, Portugal Ehud Shapiro, Weizmann Institute, Israel; Chairman

### List of Referees

Abramson, Harvey Angluin, Dana Barbuti, R. Ben-Ari, Mordechai Berkling, Klaus Bowen, Ken Bruynooghe, Maurice Byrd, Lawrence Carlsson, Mats Chikayama, Takashi Clocksin, William F. Codish, Michael Cohen, Jacques Cohen, Shimon Colmerauer, Alain Conery, John Crammond, Jim Darlington, John Davis, Al DeGroot, Doug Dershowitz, Nachum Eggert, P. Francez, Nissim Futo, Ivan Gallaire, Herve Caugin, J.A. Gostelow, Kim P. Goto, Atsuhiro Gregory, Steve Hammond, Peter Harel, David Haridi, Seif Harrison, P.G. Jchiyoshi, Nobuyuki Jaffar, J.

Kaplan, Stephane Kasif, Simon Keller, Robert M. Kibler, Dennis Kitakami, H. Kodratoff, Yves Komorowski, Jan Kowalski, Robert Kusalik, Tony Lassez, J.L. Levy, Jacob Lieberman, Henry Lindstrom, Gary Lowry, Andy Lusk, Ewing Maher, Michael Malachi, Yonni Maler, Oded Martelli, Maurizio Matsumoto, Yuji McCord, Michael C. McDermott, Drew V. Mellish, Christopher S. Miranker, Dan Miyazaki, T. Mizugochi, F. Naish, Lee Nakushima, H. O'Keefe, Richard A. Onai, Rikio Overheek, Ross Percira, Fernando Pereira, Luis M. Pinter, Ron Plaisted, David

Pnueli, Amir Reddy, Uday Reeve, Mike Robinson, Alan Roussel, Phillipe Rudolph, Larry Safra, Shmuel Sammut, Claude Saraswat, Vijay Sato, Masahiko Sato, Taisuke Sergut, M. Shamir, Adi Shertz, Zahava Shmueli, Oded Snir, Mark Spacek, Libor Sridharan, N.S. Sterling, Leon Stickel, Mark E. Takeuchi, Akikazu Tamaki, Hisao Tarnlund, Sten Ake Taylor, Steve Tick, Evan Ueda, Kazunori Veinbaum, David Waldinger, R. Walker, Adrian Warren, David S. Weiner, Jim Wilson, Walter Wise, Michael Yokota, M. Yoshida, Hiroyuki

### Contents

Keynote address: The role of logic programming in the Fifth Generation Computer Project Kazuhiro Fuchi and Koichi Furukawa, ICOT
Session 1a: Parallel implementations
An abstract machine for restricted AND-parallel execution of logic programs Manuel V. Hermenegildo, University of Texas at Austin
Efficient management of backtracking in AND-Parallelism Manuel V. Hermenegildo, University of Texas at Austin, and Roger I. Nasr, MCC 40
An intelligent backtracking algorithm for parallel execution of logic programs Yow-Jian Lin, Vipin Kumar and Clement Leung, University of Texas at Austin
Delta Prolog: a distributed backtracking extension with events Luis Moniz Pereira, Luis Monteiro, Jose Cunha and Joaquim N. Aparicio, University Nova de Lisboa
Session 1b: Theory and complexity
OLD resolution with tabulation Hisao Tamaki, Ibaraki University and Taisuke Sato, Sakuramura
Logic programs and alternation Petr Štěpánek and Olga Štěpánková, MFF Prague
Intractable unifiability problems and backtracking D.A. Wolfram, Syracuse University
On the complexity of unification sequences Heikki Mannila and Esko Ukkonen, University of Helsinki
Session 2a: Implementations and architectures
How to invent a Prolog machine Peter Kursawe, GMD and University of Karlsruhe
A sequential implementation of Parlog Ian Foster, Steve Gregory, Graem Ringwood, Imperial College, and Ken Satoh, Fujitsu Ltd. 149
A GHC abstract machine and instruction set Jacob Levy, Weizmann Inst
A Prolog processor based on a pattern matching memory device Ian Robinson, Schlumberger Palo Alto Res

### Session 2b: Inductive inference and debugging

An improved version of Shapiro's model inference system Matthew Huntbach, University of Sussex
A framework for ICAI systems based on inductive inference and logic programming Kazuhisa Kawai, Toyohashi University, Riichiro Mizoguchi, Osamu Kakusho and Jun'ichi Toyoda, Osaka University
Rational debugging in logic programming Luis Moniz Pereira, University Nova de Lisboa
Using definite clauses and integrity constraints as the basis for a theory formation approach to diagnostic reasoning Randy Goebel, University of Waterloo, Koichi Furukawa, ICOT, and David Poole, University of Waterloo
Invited talk: Some issues and trends in the semantics of logic programming
J. Jaffar, Jean-Louis Lassez and M.J. Maher, IBM
Wednesday, July 16
Invited talk: Parallel logic programming languages Akikazu Takeuchi and Koichi Furukawa, ICOT
Session 3a: Concurrent logic languages
P-Prolog: a parallel logic language based on exclusive relation Rong Yang and Hideo Aiso, Keio University
Making exhaustive search programs deterministicKazunoti Ueda, ICOTKazunoti Ueda, ICOT
Compiling OR-parallelism into AND-parallelism Michael Codish and Ehud Shapiro, Weizmann Inst
Shared memory execution of committed-choice languages Jacob Levy, Weizmann Inst
Session 3b: Theory and semantics
Logic program semantics for programming with equations Joxan Jaffar and Peter J. Stuckey, Monash University
On the semantics of logic programming languages Alberto Martelli and Gianfranco Rossi, University di Torino

Towards a formal semantics for concurrent logic programming languages Lennart Beckman, Uppsala University

335

### Thursday, July 17

Invited talk: Design of a Prolog-based machine translation system Michael McCord, IBM
Session 4a: Parallel applications and implementations
Parallel logic programming for numeric applications Ralph Butler, Ewing Lusk, William McCune and Ross Overbeek, Argonne Natl. Lab 375
Sequential and concurrent deterministic logic grammars Harvey Abramson, University of British Columbia
A parallel parsing system for natural language analysis Yuji Matsumoto, ICOT
Session 4b: Theory and higher-order functions
Equivalences of logic programs Michael J. Maher, University of Melbourne
Qualified answers and their application to transformation Phil Vasey, Imperial College
Procedures in Horn-clause programming M.A. Nait Abdallah, University of W. Ontario
Higher-order logic programming Dale A. Miller and Gopalan Nadathur, University of Pennsylvania
Session 5a: Program analysis
Abstract interpretation of Prolog programs C.S. Mellish, University of Sussex
Verification of Prolog programs using an extension of execution Tadashi Kanamori, Mitsubishi Electric Corp., and Hirohisa Seki, ICOT
Detection and optimization of functional computations in Prolog Saumya K. Debray and David S. Warren, SUNY at Stony Brook
Control of logic program execution based on the functional relations Katsuhiko Nakamura, Tokyo Denki University
Session 5b: Applications and teaching
Declarative graphics Richard Helm and Kim Marriott, University of Melbourne
Test-pattern generation for VLSI circuits in a Prolog environment Rajiv Gupta, SUNY at Stony Brook
Using Prolog to represent and reason about protein structure C.J. Rawlings, W.R. Taylor, J. Nyakairu, J. Fox and M.J.E. Sternberg, Imperial Cancer Res.

Fund

**BIBLIOTHEQUE DU CERIST** 

and Birkbeck College	536
A New approach for introducing Prolog to naive users Oded Maler, Zahava Scherz and Ehud Shapiro, Weizmann Inst	544
Invited talk: Prolog programming environments: Architecture and implementation Takashi Chikayama, ICOT	552
Friday, July 18	
Invited talk: Design overview of the NAIL! system Katherine Morris, Jeffrey D. Ullman and Allen Van Gelder, Stanford University $\dots$ .	554
Session 6a: Implementations and databases	
A superimposed codeword indexing scheme for very large Prolog databases Kotagiri Ramamohanarao and John Shepherd, University of Melbourne	569
Interfacing Prolog to a persistent data store D.S. Moffat and P.M.D. Gray, University of Aberdeen	577
A general model to implement DIF and FREEZE P. Boizumault, CNRS	585
Cyclic tree traversal Martin Nilsson and Hidehiko Tanaka, University of Tokyo	593
Session 6b: Theory and negation	
Completeness of the SLDNF-resolution for a class of logic programs R. Barbuti, University di Pisa, and M. Martelli, C.N.R., and Syracuse University	600
Choices in, and limitations of, logic programming Paul J. Voda, University of British Columbia	615
, -	
Negation and quantifiers in NU-Prolog Lee Naish, University of Melbourne	624
Negation and quantifiers in NU-Prolog Lee Naish, University of Melbourne	624 635
Negation and quantifiers in NU-Prolog Lee Naish, University of Melbourne	624 635
Negation and quantifiers in NU-Prolog Lee Naish, University of Melbourne	624 635 642
Negation and quantifiers in NU-Prolog   Lee Naish, University of Melbourne   Cracefully adding negation and disjunction to Prolog   David L. Poole and Randy Goebel, University of Waterloo   Session 7a: Compilation   Memory performance of Lisp and Prolog programs   Evan Tick, Stanford University   The design and implementation of a high-speed incremental portable Prolog compiler   Kenneth A. Bowen, Kevin A. Buettner, Ilyas Cicekli and Andrew K. Turk, Syracuse   University	624 635 642 650
Negation and quantifiers in NU-Prolog   Lee Naish, University of Melbourne   Gracefully adding negation and disjunction to Prolog   David L. Poole and Randy Goebel, University of Waterloo   Session 7a: Compilation   Memory performance of Lisp and Prolog programs   Evan Tick, Stanford University   The design and implementation of a high-speed incremental portable Prolog compiler   Kenneth A. Bowen, Kevin A. Buettner, Ilyas Cicekli and Andrew K. Turk, Syracuse   University   Compiler optimizations for the WAM   Andrew K. Turk, Syracuse University	624 635 642 650 657

Fast decompilation of compiled Prolog clauses Kevin A. Buettner, Syracuse University	663
Session 7b: Models of computation and implementation	
Logic continuations Christopher T. Haynes, Indiana University	671
Cut and Paste - defining the impure primitives of Prolog Chris Moss, Imperial College	686
Tokio: logic programming language based on temporal logic and its compilation to Prolog M. Fujita, Fujitsu Labs. Ltd., S. Kono, H. Tanaka and T. Moto-oka, University of Tokyo .	695
The OR-forest description for the execution of logic programs Sun Chengzheng and Tzu Yungui, Changsha Inst.	710

+

**BIBLIOTHEQUE DU CERIST** 

# **BIBLIOTHEQUE DU CERIST**

### The Role of Logic Programming in the Fifth Generation Computer Project

Kasuhiro Fuchi and Koichi Furukawa

ICOT Research Center Institute for New Generation Computer Technology 1-4-28, Mita, Minato-ku, Tokyo 108 Japan

**Abstract.** This paper describes the role of logic programming in the Fifth Generation Computer Project. We started the project with the conjecture that logic programming is the "bridge" connecting knowledge information processing and parallel computer architecture. Four years have passed since we started the project, and now we can say that this conjecture has been substantially confirmed. The paper gives an overview of the developments which have reinforced this foundational conjecture and how our "bridge" is being realized.

### 1. INTRODUCTION

The FGCS project started four years ago, but its roots go back three years before that. More than a hundred representative researchers in Japan participated in the discussions during those three years. A clear consensus emerged that logic programming should be placed at the center of the project.

We announced this idea at FGCS '81 in the fall of 1981. It was the most controversial issue at the conference, criticized as a reckless proposal without scientific justification.

Why did we persist in our commitment to logic programming? Because we were inspired by the insight that logic programming could become the newly unified principle in computer science. At the conference, one of the authors pointed out that logic programming covers computer architecture, new programming style, semantics of program language and database semantics. It was also pointed out that logic programming is playing an important role in linguistics and artificial intelligence.

Looking at the situation now, we can say that our conjecture has taken more concrete shape. It is not too much to say that the successes of the project so far are in large measure due to our adoption of logic programming. We would like to emphasize that the key feature of our project is not knowledge information science or non von Neumann architecture, but logic programming.

The results we have achieved seem to be quite natural. Therefore, it may be more appropriate to say that what we have here is a case of "discovery" rather than "invention."

Our approach may be compared to the process of solving a jigsaw puzzle. The process of putting each piece of a jigsaw puzzle in the right place may correspond to the process of discovering truth in our research. Also, the completed form of the jigsaw puzzle corresponds to the highly parallel computer for knowledge information processing realized in VLSI. Logic programming is the centerpiece of the jigsaw puzzle. To use Kowalski's expression [Kowalski 82], logic programming is the "missing link" connecting knowledge information processing and highly parallel architecture. Once we realized this, the puzzle started falling into place rapidly.

We often hear people say that our project ought to be more software-oriented rather than hardware-oriented. But our perspective is the whole jigsaw puzzle, of which software and hardware are certainly inseparable aspects, but neither takes priority as such. We proceed step by step, with our vision of the new computer guiding our approach to solution of each problem as it arises.

How much of this giant jigsaw puzzle have we solved in these four years? To answer this question, we would like to describe the achievements of the FGCS Project in Section 2. Section 3 is an attempt to summarize trends in research around the world and noteworthy results. We will give the conclusion in Section 4. As a part the conclusion, we discuss what research topics remain for the future, emphasizing the importance of international cooperation.

### 2. MAIN RESULTS OF FGCS PROJECT SO FAR

The work of the FGCS project extends over four research areas: software research, hardware research, development of software tools for R & D, and applications researches as feasibility studies.

In this section, we survey the main results in each of the first three areas. With regard to the fourth areas, we omit the description because of space limitation.

### 2.1. Software Research

### 2.1.1. Kernel Language

The most important issue in a programming language is its expressive power. The reason we put logic programming languages at the heart of the project is because we realized that logic programming is potentially very rich in expressive power.

When we started the project, we selected Prolog as a tentative target language and proceeded with research. The fact was that for practical purposes, Prolog was the only logic programming language available at that time. It was well known that Prolog possessed superior database and solution-searching capabilities. For us, its suitability for algorithm description, list processing and meta programming was actually more important. The importance of these functions became even clearer over the four years research of the project.

By the way, when the project started, we had already recognized that what was lacking in Prolog was the object-oriented programming facility. Object-oriented programming mechanisms cannot be realized by simple call-return control structures. Interrupt/resume computation facilities are necessary for handling exchange of messages. What this means is that we need the functions of parallel programming. Various studies have been carried out to realize parallel programming in logic programming. The following three approaches predominate in these studies: (1) Addition of parallel control primitives such as "wait" and "resume" to Prolog. (2) Delay of evaluation of goals until the specific data arrives. (3)