NORTH-HOLLAND

GUEST EDITORS' INTRODUCTION: HIGH-PERFORMANCE IMPLEMENTATIONS OF LOGIC PROGRAMMING SYSTEMS

GOPAL GUPTA AND MATS CARLSSON

This special issue of JLP is devoted to high-performance implementations of logic programming systems. The idea of having a special issue was born during a conversation with the editor-in-chief of the Journal of Logic Programming, Maurice Bruynooghe, at the International Logic Programming Symposium in Ithaca, NY, in 1994. Maurice discussed the idea with Manuel Hermenegildo, the area editor for implementation and architecture, and gave us a go-ahead in December 1994. We were motivated to have a special issue on high-performance implementations, because if one looks at the conference proceedings of ILPSs and ICLPs and at journals such as JLP, one finds very few articles on implementation aspects of LP. Surely, this is not because no interesting work is being done in the area of LP implementation. Thus, our objective in compiling this special issue is to bring the research work being done on logic programming implementation to the foreground.

A call for papers was issued in early January 1995. Abstracts were to be submitted first by February 15th, followed by full papers by February 28th. To our surprise we received a large number of abstracts, about 35. However, only 28 of them materialized into papers. The papers came from 15 different countries spread across four different continents. For one abstract we did not receive a full paper because one of its co-authors was from Kobe University. Between the deadlines for submission of the abstract and the full paper, Kobe, Japan, was hit by a devastating earthquake. Therefore, we dedicate this special issue to the survivors and victims of the Kobe earthquake.

Each paper was sent to three referees for reviewing. After the first round of reviewing, ten papers were tentatively selected. Most papers were of very high quality, and we could not include many good papers due to lack of space. Of the ten papers selected, nine had been submitted as extended papers and one as a short paper. The authors of the selected papers were asked to revise their papers. The revised papers went through another round of reviewing, and those that

THE JOURNAL OF LOGIC PROGRAMMING

655 Avenue of the Americas, New York, NY 10010

0743-1066/96/$15.00
PII S0743-1066(96)00066-0

needed further revision were asked to be revised again. The revised papers were finally checked by us to make sure that all of the concerns raised by the reviewers had been met. Nine of the ten papers appear in this special issue. One paper (Automatic Compile-time Parallelization of Logic Programs for Restricted, Goal Level Independent And-Parallelism by K. Muthukumar, F. Bueno, María José García de la Banda, and M. Hermenegildo) could not make this issue due to various delays, and will appear later.

The ten papers selected represent a wide array of topics in sequential and parallel implementation technology. Most papers report results from long-running research projects. Topics covered range from hardware support for efficient execution, compilation to novel abstract instruction sets, static analysis for execution-speed enhancement, to models and implementations for exploiting parallelism from Prolog. These papers provide a glimpse into today's state-of-the-art of logic programming implementation technology. Having compiled this special issue, we can say with confidence that the area of LP implementation is alive and vibrant.

The first paper in this issue, by Andrew Taylor, reports experience with the PARMA project. Using abstract interpretation and global analysis techniques, and by compiling Prolog to MIPS instruction set, the author was able to match (or surpass) the speed of equivalent C code. This paper is the only short paper in the special issue.

The second paper, titles "The Execution Algorithm of Mercury, An Efficient Purely Declarative Logic Programming Language," by Zoltan Somogyi, Fergus Henderson, and Thomas Conway, reports on the design and implementation of the Mercury language. Mercury is designed to enable writing of large programs by groups of programmers. It has a sophisticated compiler that catches many more errors than standard Prolog compilers. The paper outlines the techniques used in the design of the compiler and the authors' experience in programming this compiler in Mercury.

The third paper, by Paul Tarau, Koen De Bosschere, and Bart Demoen, titled "Partial Translation: Towards a Portable and Efficient Prolog Implementation Technology," presents a new language translation framework in which selected abstract machine instruction sequences are compiled into C code. Translation to C has become a popular technique for implementing logic programming systems. This paper provides insights into this new implementation technique.

The fourth paper, by Andreas Krall, presents the design of the Vienna Abstract Machine. The Vienna Abstract Machine is a modification of the WAM to make it more efficient. Three versions of the Vienna Abstract Machine are described—for native code compilation, for interpretation, and for abstract execution.

The fifth paper, by Bruce Holmer et al., presents minimal extensions needed to a standard microprocessor architecture in order to execute Prolog programs efficiently. An extended chip with support for Prolog-specific operations has been fabricated and tested, and the results reported in this paper.

The sixth paper, by Evan Tick, Bart Massey, and James Larson, describes the design of an optimizing compiler for a parallel implementation of a flat committed choice language. The authors describe the salient features of this optimizing compiler along with their experience in building it.

The seventh paper, by Saumya Debray, David Gudeman, and Peter Bigot, presents a dataflow analysis for recognizing suspension-free logic programs. Several

optimizations that rely on this analysis are developed and incorporated into the authors' `jc` system and the resulting Performance improvement reported in the paper.

The eighth paper, by Donald Smith, presents the MultiLog system that can be used for executing Prolog programs in data-parallel. Data-parallel execution of Prolog programs, when possible, leads to considerable efficiency, as Smith's results show.

The ninth paper, by Kish Shen, presents issues in designing and implementing a parallel Prolog system that exploits dependent and-parallelism. The paper presents an abstract execution model called DDAS for exploiting dependent and-parallelism as well as its concrete implementation in the DASWAM system.

We hope you will enjoy this special issue.

We were helped by approximately 70 reviewers in the reviewing process, who graciously agreed to help us. The help of these reviewers is gratefully acknowledged. Special thanks go to Maurice Bruynooghe, the editor-in-chief of the JLP, and Manuel Hermenegildo, area editor for implementation and architecture, for not only accepting the idea of a special issue, but also providing a tremendous amount of help. Maurice also helped us in making many of the difficult decisions during the whole process and answered many of our questions related to logistics of publication. Thanks also to Federico Bassetti and Enrico Pontelli for help in distributing the papers to reviewers. Finally, we would like to acknowledge the support provided by our respective institutions.

Gopal Gupta
Las Cruces, New Mexico
Mats Carlsson
Uppsala, Sweden