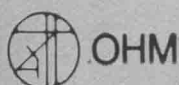


FIFTH GENERATION COMPUTER SYSTEMS 1988

**Proceedings of the International Conference on
Fifth Generation Computer Systems 1988**

**Edited by
Institute for New Generation
Computer Technology (ICOT)**

Volume 3



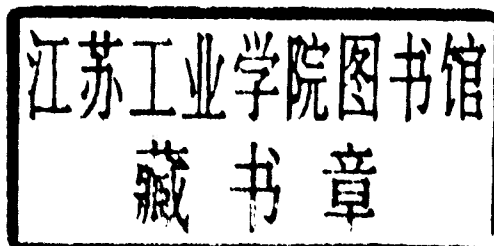
Springer-Verlag



FIFTH GENERATION COMPUTER SYSTEMS 1988

**Proceedings of the International Conference on
Fifth Generation Computer Systems 1988**
Tokyo, Japan
November 28-December 2, 1988

**Edited by
Institute for New Generation
Computer Technology (ICOT)
Tokyo, Japan**



Volume 3



OHM



Springer-Verlag

FIFTH GENERATION COMPUTER SYSTEMS 1988

Proceedings of the International Conference on Fifth Generation Computer Systems 1988

Copyright © 1988 by Institute for New Generation Computer Technology

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, electronic, mechanical, recording or otherwise, without the prior permission of the copyright owner.

Distribution

Sole distribution rights outside Japan granted to Springer-Verlag

All orders should be sent to the following addresses:

Japan:

OHMSHA LTD., 3-1 Kanda Nishiki-cho, Chiyoda-ku, Tokyo 101, Japan

North America:

Springer-Verlag NY Inc., 175 Fifth Avenue, New York, NY 10010

Rest of the world:

SPRINGER-VERLAG, Heidelberger Platz 3, 1000 Berlin 33, FRG

British Library Cataloguing in Publication Data

International Conference on Fifth Generation Computers: 1988

International Conference on Fifth Generation Computer Systems (FGCS '88).

1. Computer systems

I. Title

004

ISBN 3-540-19558-0

Library of Congress Cataloging-in-Publication Data

International Conference on Fifth Generation Computer systems (1988: Tokyo, Japan)

Proceedings of the International Conference on Fifth Generation Computer Systems 1988:

Nov. 28– Dec. 2, 1988, Tokyo Prince Hotel, Tokyo Japan/

Institute for New Generation Computer Technology.

p. cm.

Bibliography: p.

Includes indexes,

ISBN 0-387-19558-0 (U.S.)

1. Fifth generation computers — Congresses. I. Shin Sedai Konpyūta Gijutsu Kaihatsu Kikō (Japan) II. Title.

QA76.85.I58, 1988

004 — dc20

89-11366

CIP

ISBN-3-540-19558-0 Springer-Verlag Berlin Heidelberg New York

ISBN 0-387-19558-0 Springer-Verlag New York Berlin Heidelberg

ISBN 4-274-19558-0 OHMSHA, LTD.

Printed in Japan

TABLE OF CONTENTS

Volume 1

KEYNOTE SPEECH

Hop, Step and Jump <i>K. Fuchi</i>	1
---	---

ICOT RESEARCH AND DEVELOPMENT

Present Status and Plans for Research and Development <i>T. Kurozumi</i>	3
Research and Development of the Parallel Inference System in the Intermediate Stage of the FGCS Project <i>S. Uchida, K. Taki, K. Nakajima, A. Goto and T. Chikayama</i>	16
Knowledge Base System in Logic Programming Paradigm <i>H. Itoh, H. Monoi, S. Shibayama, N. Miyazaki, H. Yokota and A. Konagaya</i>	37
Problem-Solving and Inference Software <i>R. Hasegawa and researchers of the First Research Laboratory</i>	54
The Research and Development of Natural Language Processing Systems in the Intermediate Stage of the FGCS Project <i>S. Uchida, T. Yoshioka, R. Sugimura, Y. Tanaka, K. Hasida and K. Mukai</i>	70
Experimental Knowledge Processing System <i>Y. Fujii, H. Taki and other researchers of the Fifth Research Laboratory</i>	85

INVITED LECTURES

Prospects for Cognitive Science <i>H.A. Simon</i>	111
Logic Programming Schemes <i>K.L. Clark</i>	120

PANEL DISCUSSIONS

Social Impact of Information Technology and International Collaboration

Social Impact of Information Technology and International Collaboration <i>H. Karatsu</i>	143
Artificial Intelligence: Perspectives and Predictions <i>J.H. Siekmann</i>	145
International Collaboration in IT <i>T.H. Walker</i>	147
Social Impacts of Advanced Computers <i>F.W. Weingarten</i>	151

Theory and Practice of Concurrent Systems

The Panel on Theory and Practice of Concurrent Systems <i>E. Shapiro</i>	152
Mechanisms for Concurrent Computing <i>W.J. Dally</i>	154
Theory and Practice of Concurrent Systems <i>G. Fox</i>	157

Knowledge Processing	
<i>C. Hewitt</i>	161
Some Directions in Concurrency Theory	
<i>R. Milner</i>	163
Theory and Practice of Concurrent Systems—The Role of Kernel Language in the FGCS Project—	
<i>K. Ueda</i>	165
Theory and Practice of Concurrent Systems — A Position Paper	
<i>D.H.D. Warren</i>	167

SPECIAL SESSION

Progress and Future Plans of Knowledge Information Processing

Advanced Information Processing in ESPRIT — Status and Plans	
<i>J-M Cadiou</i>	171
A Review of MCC's Accomplishments and Strategic Outlook for Knowledge-Based Systems	
<i>E. Lowenthal</i>	180
UK IKBS Programmes	
<i>T.E. Walker</i>	189

ICOT RESEARCH TOPICS

Overview of Knowledge Base Mechanism	
<i>S. Shibayama, H. Sakai, T. Takewaki, H. Monoi, Y. Morita and H. Itoh</i>	197
Overview of the Parallel Inference Machine Architecture (PIM)	
<i>A. Goto, M. Sato, K. Nakajima, K. Taki and A. Matsumoto</i>	208
Overview of the Parallel Inference Machine Operating System (PIMOS)	
<i>T. Chikayama, H. Sato and T. Miyazaki</i>	230
Overview of the Knowledge Base Management System (KAPPA)	
<i>K. Yokota, M. Kawamura and A. Kanaegami</i>	252
Constraint Logic Programming Language CAL	
<i>A. Aiba, K. Sakai, Y. Sato, D.J. Hawley and R. Hasegawa</i>	263
Overview of the Dictionary and Lexical Knowledge Base Research	
<i>Y. Tanaka and T. Yoshioka</i>	277
A Software Environment for Research into Discourse Understanding Systems	
<i>R. Sugimura, K. Hasida, K. Akasaka, K. Hatano, Y. Kubo, T. Okunishi and T. Takizuka</i>	285
Expert System Architecture for Design Tasks	
<i>Y. Nagai, S. Terasaki, T. Yokoyama and H. Taki</i>	296

AUTHORS INDEX	i
---------------------	---

Volume 2

FOUNDATION

INVITED PAPER

Interpreting One Concurrent Calculus in Another <i>R. Milner</i>	321
---	-----

SUBMITTED PAPERS

Functional Logic Programming

The Semantics of a Functional Logic Language with Input Mode <i>D.W. Shin, J.H. Nang, S.R. Maeng and J.W. Cho</i>	327
Conditional Equational Programming and the Theory of Conditional Term Rewriting <i>N. Dershowitz and M. Okada</i>	337

Theory of Parallel Computation

Uniform Abstraction, Atomicity and Contractions in the Comparative Semantics of Concurrent Prolog <i>J.W. de Bakker and J.N. Kok</i>	347
Parallel Computational Complexity of Logic Programs and Alternating Turing Machines <i>Y. Okabe and S. Yajima</i>	356
Finite Failures and Partial Computations in Concurrent Logic Languages <i>M. Falaschi and G. Levi</i>	364
A Declarative Semantics of Parallel Logic Programs with Perpetual Processes <i>M. Murakami</i>	374

Formal Semantics

Semantics of Logic Programs over Sequence Domains <i>S. Yamasaki</i>	382
Local Definitions with Static Scope Rules in Logic Programming <i>L. Giordano, A. Martelli and G.F. Rossi</i>	389
WEIGHTED GRAPHS, A Tool for Expressing the Behaviour of Recursive Rules in Logic Programming <i>P. Devienne</i>	397

Program Analysis and Transformation (2)

Horn Equality Theories and Complete Sets of Transformations <i>S. Hölldobler</i>	405
Preservation of Stronger Equivalence in Unfold/Fold Logic Program Transformation <i>T. Kawamura and T. Kanamori</i>	413
An Abstract Interpretation Scheme for Logic Programs Based on Type Expression <i>A.K. Bansal and L. Sterling</i>	422
Transformation of Strictness-Related Analyses Based on Abstract Interpretation <i>M. Ogawa and S. Ono</i>	430

Reasoning and Learning

Rules and Justifications: A Uniform Approach to Reason Maintenance and Non-Monotonic Inference <i>M. Reinfrank and H. Freitag</i>	439
---	-----

An Efficient Learning of Context-Free Grammars for Bottom-Up Parsers <i>Y. Sakakibara</i>	447
Nonmonotonic Reasoning by Minimal Belief Revision <i>K. Satoh</i>	455
Generating Rules with Exceptions <i>J. Arima</i>	463

Situation Semantics

Situation Semantics and Semantic Interpretation in Constraint-Based Grammars <i>P-K. Halvorsen</i>	471
Partially Specified Term in Logic Programming for Linguistic Analysis <i>K. Mukai</i>	479
Towards a Computational Interpretation of Situation Theory <i>H. Nakashima, H. Suzuki, P.K. Halvorsen and S. Peters</i>	489

Logic and Theorem Proving

Knowledge Representation and Inference Based on First-Order Modal Logic <i>K. Iwanuma and M. Harao</i>	499
Declarative Semantics for Modal Logic Programs <i>Ph. Balbiani, L. Farin�s Del Cerro and A. Herzig</i>	507
Epistemic Logic Programming <i>Y.J. Jiang</i>	515
Theorem-Proving with Resolution and Superposition: An Extension of the Knuth and Bendix Procedure to a Complete Set of Inference Rules <i>M. Rusinowitch</i>	524

SPECIAL SESSION

Messages from Parallel Complexity Theory: Does Parallelism Help?

Parallel Complexity and P-Complete Problems <i>S. Miyano</i>	532
Parallel Approximation Algorithms <i>E.W. Mayr</i>	542

SOFTWARE

SUBMITTED PAPERS

Program Analysis and Transformation (1)

Algebraic Meta-Level Programming in Prolog <i>G. Louis and M. Vauclair</i>	555
Program Transformation Applied to the Derivation of Systolic Arrays <i>N. Yoshida</i>	565
The Use of Assertions in Algorithmic Debugging <i>W. Drabent, S. Nadjm-Tehrani and J. Maluszynski</i>	573
Transformation Rules for GHC Programs <i>K. Ueda and K. Furukawa</i>	582

SPECIAL SESSION

Meta-Computation and Reflection

A Tutorial Introduction to Metaclass Architecture as Provided by Class Oriented Languages <i>P. Cointe</i>	592
---	-----

Directions for Meta-Programming	
<i>J.W. Lloyd</i>	609
Reasoning about Knowledge and Ignorance	
<i>L. C. Aiello, D. Nardi and M. Schaerf</i>	618

SUBMITTED PAPERS

Computation Models

Software for the Rewrite Rule Machine	
<i>J.A. Goguen and J. Meseguer</i>	628
A'UM — A Stream-Based Concurrent Object-Oriented Language —	
<i>K. Yoshida and T. Chikayama</i>	638
Guarded Horn Clause Languages: Are They Deductive and Logical?	
<i>C. Hewitt and G. Agha</i>	650

Functional Programming

Lazy Evaluation of FP Programs: A Data-Flow Approach	
<i>Y-H. Wei and J-L. Gaudiot</i>	658
Committed Choice Functional Programming	
<i>G. Båge and G. Lindstrom</i>	666
A Progress Report on the LML Project	
<i>B. Bertolino, P. Mancarells, L. Meo, L. Nini, D. Pedreschi and F. Turini</i>	675

INVITED PAPER

Program Evaluation and Generalized Partial Computation	
<i>Y. Futamura</i>	685

SUBMITTED PAPERS

Constraint Logic Programming

The Constraint Logic Programming Language CHIP	
<i>M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf and F. Berthier</i>	693
Applications of a Canonical Form for Generalized Linear Constraints	
<i>J.L. Lassez and K. McAloon</i>	703

Deductive Data Bases

A Query Independent Method for Magic Set Computation on Stratified Databases	
<i>I. Balbin, K. Meenakshi and K. Ramamohanarao</i>	711
Efficient Query Answering on Stratified Databases	
<i>J-M. Kerisit and J-M. Pugin</i>	719
Answering Linear Recursive Queries in Cyclic Databases	
<i>C-S. Wu and L.J. Henschen</i>	727
CAP — A Three-Phase Query Processing Technique for Indefinite Databases	
<i>S. Chi and L.J. Henschen</i>	735

Parallel Programming Languages

ANDORRA Prolog — An Integration of Prolog and Committed Choice Languages	
<i>S. Haridi and P. Brand</i>	745
Design of a Concurrent Language for Distributed Artificial Intelligence	
<i>J. Ferber and J-P. Briot</i>	755
The Language FCP (:,?)	
<i>S. Kliger, E. Yardeni, K. Kahn and E. Shapiro</i>	763
Meta-Interpreters and Reflective Operations in GHC	
<i>J. Tanaka</i>	774

Logic Programming Languages

Tables as a User Interface for Logic Programs

M.H.M. Cheng, M.H. van Emden and J.H.M. Lee 784

Modular and Communicating Objects in SICStus Prolog

N.A. Elshiewy 792

Benchmarking of Prolog Procedures for Indexing Purposes

M. Meier 800

Foundations of DISLOG, Programming in Logic with Discontinuities

P. Saint-Dizier 808**AUTHORS INDEX** i

Volume 3

ARCHITECTURE

SUBMITTED PAPERS

Parallel Prolog Systems

The Aurora Or-Parallel Prolog System

E. Lusk, R. Butler, T. Disz, R. Olson, R. Overbeek, R. Stevens, D.H.D. Warren, A. Calderwood, P. Szeredi, S. Haridi, P. Brand, M. Carlsson, A. Ciepielewski and B. Hausman 819

Cut and Side-Effects in Or-Parallel Prolog

B. Hausman, A. Ciepielewski and A. Calderwood 831

The Parallel ECRC Prolog System PEPsSys: An Overview and Evaluation Results

U. Baron, J. Chassin de Kergommeaux, M. Hailperin, M. Ratcliffe, P. Robert, J-C. Syre and H. Westphal 841

Performance of AND-Parallel Execution of Logic Programs on a Shared-Memory Multiprocessor

Y-J. Lin and V. Kumar 851

Parallel Architectures (1)

Design of an Efficient Dataflow Architecture Without Data Flow

G.R. Gao, R. Tio and H.H.J. Hum 861

Cell and Ensemble Architecture for the Rewrite Rule Machine

S. Leinwand, J.A. Goguen and T. Winkler 869

A VLSI Building Block for Massively Parallel Computation

A. Ashana, B. Mathews, C.J. Briggs and M.R. Cravatts 879

Multiport Memory Architectures

Y. Tanaka 887

Parallel Architectures (2)

Unification-Based Query Language for Relational Knowledge Bases and Its Parallel Execution

H. Monoi, Y. Morita, H. Itoh, T. Takewaki, H. Sakai and S. Shibayama 896

A New External Reference Management and Distributed Unification for KL1

N. Ichiyoshi, K. Rokusawa, K. Nakajima and Y. Inamura 904

Parallelism in the PESA I Multiprocessor

F. Schreiner and G. Zimmermann 914

Implementation Techniques for Inference Machines

A Light-Weight Prolog Garbage Collector

H. Touati and T. Hama 922

A Wide Instruction Word Architecture for Parallel Execution of Logic Programs Coded in BSL

K. Ebcioglu and M. Kumar 931

INVITED PAPER

Data Diffusion Machine — A Scalable Shared Virtual Memory Multiprocessor

D.H.D. Warren and S. Haridi 943

SUBMITTED PAPERS

Parallel Inference Machines

Macro-Call Instruction for the Efficient KL1 Implementation on PIM

T. Shinogi, K. Kumon, A. Hattori, A. Goto, Y. Kimura and T. Chikayama 953

CARMEL-2: A Second Generation VLSI Architecture for Flat Concurrent Prolog	
<i>A. Harsat and R. Ginosar</i>	962
Multi-Context Processing and Data Balancing Mechanism of the Parallel Inference Machine PIE64	
<i>H. Koike and H. Tanaka</i>	970
A Load Balancing Mechanism for Large Scale Multiprocessor Systems and Its Implementation	
<i>Y. Takeda, H. Nakashima, K. Masuda, T. Chikayama and K. Taki</i>	978

Scheduling for Parallel Machines

Load-Dispatching Strategy on Parallel Inference Machines	
<i>M. Sugie, M. Yoneyama, N. Ido and T. Tarui</i>	987
Compile-Time Granularity Analysis for Parallel Logic Programming Languages	
<i>E. Tick</i>	994
A Highly Parallel Chess Program	
<i>E.W. Felten and S.W. Otto</i>	1001

SPECIAL SESSION

Parallelism in AI

Artificial Intelligence Related Research on the Connection Machine	
<i>D.L. Waltz and C. Stanfill</i>	1010
Artificial Intelligence and Neural Computing	
<i>Y. Anzai</i>	1025

SUBMITTED PAPERS

Implementation Model for Parallel Logic Languages

Massively Parallel Implementation of Flat GHC on the Connection Machine	
<i>M. Nilsson and H. Tanaka</i>	1031
KL1 in Condition Graphs on a Connection Machine	
<i>J. Barklund, N. Hagner and M. Wafin</i>	1041
A Parallel Implementation of GHC	
<i>J.R.W. Glauert and G.A. Papadopoulos</i>	1051
LogDf: A Data-Driven Abstract Machine Model for Parallel Execution of Logic Programs	
<i>P. Biswas and C-C. Tseng</i>	1059

APPLICATIONS

SUBMITTED PAPERS

Graphics and Music

Toward Intelligent Interfaces for Graphic Design Applications	
<i>H. Liberman</i>	1073
How to Realize Jazz Feelings — A Logic Programming Approach —	
<i>K. Hirata, T. Aoyagi and H. Konaka</i>	1081

INVITED PAPER

Multiple Reasoning Styles in Logic Programming	
<i>H. Gallaire</i>	1089

SUBMITTED PAPERS

Natural Language (1)

Direct Memory Access Translation for Speech Input — A Massively Parallel Network of Episodic/Thematic and Phonological Memory	
<i>H. Tomabechi, T. Mitamura and M. Tomita</i>	1100

Overview of the Core Language Engine	
H. Alshawi, D.M. Carter, J. van Eijck, R.C. Moore, D.B. Moran and S.G. Pulman	1108
Projections and Semantic Description in Lexical-Functional Grammar	
P-K. Halvorsen and R.M. Kaplan	1116
ADAM: An Extension of Situation Semantics for Practical Use	
C. Numaoka and M. Tokoro	1123

Natural Language (2)

Preference Judgement in Comprehending Conversational Sentences Using Multi-Paradigm World Knowledge	
T. Ukita, K. Sumita, S. Kinoshita, H. Sano and S. Amano	1133
A Multi-Target Machine Translation System	
M.C. McCord	1141
The Design of Post-Analysis in the JETS Japanese/English Machine Translation System	
D.E. Johnson	1150

Knowledge Representation

The Knowledge Dictionary: A Relational Tool for the Maintenance of Expert Systems	
B. Jansen and P. Compton	1159
Knowledge Representation with Abstractive Layers for Information Retrieval	
T. Koguchi, H. Kondo, M. Oba and H. Itoh	1168
CHEMILOG — A Logic Programming Language/System for Chemical Information Processing —	
T. Akutsu and S. Ohsuga	1176

Qualitative Reasoning

A Symbolic Framework for Qualitative Kinematics	
B. Faltings	1184
An Examination for Applicability of FGHC: The Experience of Designing Qualitative Reasoning System	
H. Ohwada and F. Mizoguchi	1193
Methods for Partition of Target Systems in Qualitative Reasoning	
K. Sakane, M. Ohki, J. Sawamoto and Y. Fujii	1201
Sphinx — A Hybrid Knowledge Representation System	
S. Han and J.W. Cho	1211

SPECIAL SESSION

New Paradigms of Knowledge Acquisition

Knowledge Acquisition Techniques and Tools: Current Research Strategies and Approaches	
J.H. Boose	1221
Comments on Knowledge Acquisition and Learning	
S. Kunifuji (responder)	1236
When Will Machines Learn?	
D.B. Lenat	1239
A Next-Generation Knowledge-Base from the Viewpoint of Extending Logic Framework	
M. Ishizuka (responder)	1246

SUBMITTED PAPERS

Knowledge Acquisition

Knowledge Acquisition by Observation	
H. Taki	1250
Validation in a Knowledge Acquisition System with Multiple Experts	
M.L.G. Shaw	1259

Applying Explanation-Based Generalization to Natural-Language Processing <i>M. Rayner</i>	1267
Knowledge Maintenance	
Problem Solving with Hypothetical Reasoning <i>K. Inoue</i>	1275
Representing Knowledge for Logic-Based Diagnosis <i>D. Poole</i>	1282
A Human Strategy-Based Troubleshooting Expert System for Switching Systems <i>S. Wada, Y. Koseki and T. Nishida</i>	1291
co-LODEX: A Cooperative Expert System for Logic Design <i>F. Maruyama, T. Kakuda, Y. Matsunaga, Y. Minoda, S. Sawada and N. Kawato</i>	1299
AUTHORS INDEX	i

ARCHITECTURE

THE AURORA OR-PARALLEL PROLOG SYSTEM

Ewing Lusk
Ralph Butler
Terrence Disz
Robert Olson
Ross Overbeek
Rick Stevens
*Argonne**

David H. D. Warren
Alan Calderwood
Peter Szeredi[†]
Manchester‡

Seif Haridi
Per Brand
Mats Carlsson
Andrzej Ciepielewski
Bogumil Hausman
SICS§

ABSTRACT

Aurora is a prototype or-parallel implementation of the full Prolog language for shared-memory multiprocessors, developed as part of an informal research collaboration known as the "Gigalips Project". It currently runs on Sequent and Encore machines. It has been constructed by adapting Sicstus Prolog, an existing, portable, state-of-the-art, sequential Prolog system. The techniques for constructing a portable multiprocessor version follow those pioneered in a predecessor system, ANL-WAM. The SRI model was adopted as the means to extend the Sicstus Prolog engine for or-parallel operation. We describe the design and main implementation features of the current Aurora system, and present some preliminary experimental results. We conclude with our plans for the continued development of the system and an outline of future research directions.

1 INTRODUCTION

In the last few years, parallel computers have started to emerge commercially, and it seems likely that such machines will rapidly become the most cost-effective source of computing power. However, developing parallel algorithms is currently very difficult. This is a major obstacle to the widespread acceptance of parallel computers.

Logic programming, because of the parallelism *implicit* in the evaluation of logical expressions, in principle relieves the programmer of the burden of managing parallelism explicitly. Logic programming therefore offers the potential to make parallel computers no harder to program than sequential ones, and to allow software to be migrated transparently between sequential and parallel machines.

It only remains to determine whether a logic program-

ming system coupled with suitable parallel hardware can realise this potential. The Aurora system is a first step towards this goal. Aurora is a prototype or-parallel implementation of the full Prolog language for shared-memory multiprocessors. It currently runs on Sequent and Encore machines. It has been developed as part of an informal research collaboration known as the "Gigalips Project".

The Aurora system has two purposes. Firstly, it is intended to be a research tool for gaining understanding of what is needed in a parallel logic programming system. In particular, it is a vehicle for making concrete, evaluating, and refining one (or more) parallel execution models. The intention is to evaluate the models not just on the present hardware, but with a view to possible future hardware (not necessarily based on shared physical memory).

Secondly, Aurora is intended to be a demonstration system, that will enable experience to be gained of running large applications in parallel. For this purpose, it is vital that the system should perform well on the present hardware, and that it should be a complete and practical system to use.

In order to support *real* applications efficiently and elegantly, it is necessary to implement a logic programming language that is at least as powerful and practical as Prolog. The simplest way to ensure this, and at the same time to make it easy to port existing Prolog applications and systems software, is to include full Prolog with its standard semantics as a true subset of the language. This we have taken some pains to achieve.

The bottom line for evaluating a parallel system is whether it is truly competitive with the best sequential systems. To achieve competitiveness, it is necessary to make a parallel logic programming system with a single processor execution speed as close as possible to state-of-the-art sequential Prolog systems, while allowing multiple processors to exploit parallelism with the minimum of overhead. This has been our goal in Aurora.

To summarise the objectives towards which Aurora is addressed, they are to obtain truly competitive performance on real applications by transparently exploiting parallelism in a logic programming language that in-

*Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, U.S.A.

[†]On leave from SZKI, Donati u. 35-45, Budapest, Hungary

[‡]Department of Computer Science, University of Manchester, Manchester M13 9PL, U.K. Now at: Department of Computer Science, University of Bristol, Bristol BS8 1TR, U.K.

[§]Swedish Institute of Computer Science, Box 1263, S-164 28 Kista, Sweden

cludes Prolog as a true subset.

The main purpose of this paper is to describe the issues that must be confronted in or-parallel Prolog implementation and detail the decisions and compromises made in Aurora. We include benchmark tests that point out the strengths and weaknesses of some of these decisions. We conclude by describing some directions for further research.

2 BACKGROUND

In this section we describe the setting in which Aurora was developed and give a short history of the Gialips Project.

2.1 Sequential Prolog Implementations

Prolog implementation entered a new era when the first compiler was introduced, for the DEC-10 [21]. The speed of this implementation, and the portability and availability of its descendant, C-Prolog, set a language standard, now usually referred to as the “Edinburgh Prolog”. The DEC-10 compilation techniques led as well to a standard implementation strategy, usually called the WAM (Warren Abstract Machine) [22]. In a WAM-based implementation, Prolog source code is compiled into the machine language of a stack-based abstract machine. A portable emulator of this abstract machine (typically written in C) yields a fast, portable Prolog system, and a non-portable implementation of crucial parts of the emulator can increase speed still further. A parallel implementation of Prolog is achieved by parallelizing this emulator.

There are now many high-quality commercial and non-commercial Prolog systems based on the WAM. A parallel implementation can obtain considerable leverage by utilizing an existing high-quality implementation as its foundation. We use the Sicstus implementation [7], one of the fastest portable implementations.

Using a fast implementation is important for two reasons. Firstly, the single most important factor determining the speed of a parallel version is the speed of the underlying sequential implementation. Secondly, many research issues related purely to multiprocessing only become apparent in the presence of a fast sequential implementation. (Speedups are too easy to get when speed is too low.)

2.2 Multiprocessors

It is only in the last two years that multiprocessors have emerged from the computer science laboratories to become viable commercial products marketed worldwide. Startup companies like Sequent, Encore, and Alliant have made shared-memory multiprocessors commonplace in industry and universities alike. They are relatively inexpensive and provide a standard system en-

vironment (UNIXTM) thus making them extremely popular as general-purpose computation servers. A similar revolution is happening with local-memory multiprocessors, sometimes called “multicomputers”, but these are currently more specialized machines, despite their scalability advantages.

What the new breed of machines does *not* provide is a unified way of expressing and controlling parallelism. A variety of compiler directives and libraries are offered by the vendors, and while they do allow the programmer to write parallel programs for each machine, they provide neither syntactic nor conceptual portability. A number of researchers are developing tools to address these issues, but at a relatively low level (roughly the same level as the language they are embedded in, such as C or Fortran). A goal of the Gialips Project is to determine whether it is feasible to propose logic programming as the vehicle for exploiting parallelism on these machines.

2.3 Or-Parallelism

As is well known, there are two main kinds of parallelism in logic programs, and-parallelism and or-parallelism. The issues raised in attempting to exploit the two kinds of parallelism are sufficiently different that most research efforts are focussing primarily on one or the other. Much early and current work has been directed towards and-parallelism, particularly within the context of “committed choice” languages (Parlog, Concurrent Prolog, Guarded Horn Clauses) [14, 20]. These languages exploit **dependent** and-parallelism, in which there may be dependencies between and-parallel goals. Other work [11, 18] has been directed towards the important special case of **independent** and-parallelism, where and-parallel goals can be executed completely independently.

The committed choice languages have been viewed primarily as a means of expressing parallelism *explicitly*, by modelling communicating processes. In contrast, one of our main goals is to exploit parallelism *implicitly*, in a way that need have little impact on the programmer. This viewpoint has led us to take a rather different approach, and to focus in particular on or-parallelism.

There are several reasons for focussing on or-parallelism as a first step. Briefly, in the short term, or-parallelism seems easier and more productive to exploit transparently than and-parallelism. However, none of these reasons precludes integrating and-parallelism at a later stage, and indeed this is our ultimate intention.

- **Generality.** It is relatively straightforward to exploit or-parallelism without restricting the power of our logic programming language. In particular, we retain the ability we have in Prolog to generate all solutions to a goal.
- **Simplicity.** It is possible to exploit or-parallelism without requiring any extra programmer annotation or complex compile-time analysis.