A. Voronkov (Ed.)

Logic Programming and Automated Reasoning

International Conference LPAR '92
St. Petersburg, Russia, July 15-20, 1992
Proceedings

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Series Editor

Jörg Siekmann
University of Saarland
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, W-6600 Saarbrücken 11, FRG

Volume Editor

Andrei Voronkov
European Computer-Industry Research Centre (ECRC)
Arabellastraße 17, W-8000 München 81, FRG

CR Subject Classification (1991): F.4.1, I.2.3

ISBN 3-540-55727-X Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-55727-X Springer-Verlag New York Berlin Heidelberg

Typesetting: Camera ready by author/editor

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

Preface

LPAR'92 is organized by Eurobalt Inc. and St. Petersburg Institute of Electrical Engineering in cooperation with the Russian Association for Logic Programming. It aims at bringing together researchers interested in logic programming and automated reasoning. The research in logic programming grew out of the research in automated reasoning of early 1970s. Later, the implementation techniques used in logic programming have been used in implementing theorem proving systems. Recently, results from both fields were used in deductive databases. Although the two fields have much in common, there was no common conference for them. I hope that the initiative of the Russian Association for Logic Programming to organize this conference will result in a regular internationally recognized series of conferences.

LPAR'92 is the successor of the 1st and 2nd Russian Conferences on Logic Programming held in Irkutsk in 1990 and in St. Petersburg in 1991. The proceedings of these two conferences were published in Volume 592 of Lecture Notes in Artificial Intelligence. The scientific program of LPAR'92 includes 6 invited talks, 35 talks and a session on system descriptions. All the papers from this volume were selected from 102 submitted papers. In addition, Steffen Hölldobler, Ewing Lusk and Jack Minker made it possible to prepare a written version of their invited talks for this volume. The session on system descriptions aims at initiating discussion on computer implementation of logical concepts. During the conference several systems implemented on IBM PC and compatibles will be demonstrated.

There are many people involved in the organization of LPAR'92. I wish to personally thank Gérard Comyn (ECRC), Peter Gotzmann (ECRC), Michel Parigot (University of Paris 7), Priscilla Rasmussen (Rutgers University), Tania Rybina (SINTEL), George Selvais (IRI Inc.) and IJCAI board of trustees.

I gratefully acknowledge financial sponsorship by IJCAI Inc. and ECRC GmbH.

Munich, May 1992 — Andrei Voronkov

Program Committee:

Dmitri Boulanger (Catholic University of Leuven)
François Bry (ECRC, Munich)
Mats Carlsson (SICS, Kista)
Philippe Codognet (INRIA Rocquencourt)
Eugene Dantsin (Electrical Engineering Institute, St. Petersburg)
Robert Freidson (Electrical Engineering Institute, St. Petersburg)
Steffen Hölldobler (Technische Hochschule Darmstadt)
Nikolai Ilinski (Institute of Physics Engineering, Moscow)
Alexander Leitsch (Technical University of Vienna)
Vladimir Lifschitz (University of Texas)
Ewing Lusk (Argonne National Laboratory)
Bill McCune (Argonne National Laboratory)
Dale Miller (University of Pennsylvania)
Gregory Mints (Stanford University)
Marek Sergot (Imperial College, London)
Jörg Siekmann (Universität des Saarlandes, Saarbrücken)
Petr Stepanek (Charles University, Prague)
Mark Stickel (SRI International, Menlo Park)
Konstantin Vershinin (Institute for Cybernetics, Kiev)
Andrei Voronkov (ECRC, Munich) — chairman

Organizing Committee:

Eugene Dantsin (Electrical Engineering Institute, St. Petersburg)
Robert Freidson (Electrical Engineering Institute, St. Petersburg) — chairman
Nikolai Ilinski (Institute of Physics Engineering, Moscow)
Andrei Voronkov (ECRC, Munich)

Invited Speakers:

Pascal van Hentenryck (Brown University)
Steffen Hölldobler (Technische Hochschule Darmstadt)
Vladimir Lifschitz (University of Texas)
Ewing Lusk (Argonne National Laboratory)
Jack Minker (University of Maryland)
Gregory Mints (Stanford University)

Conference Sponsors:

IJCAI Inc.
ECRC GmbH

Reviewers:

S. Abreu (INRIA Rocquencourt)
Khayri A. M. Ali (SICS, Kista)
Jean-Marc Andreoli (ECRC, Munich)
Matthias Baaz (Teschnische Universität Wien)
F. Barthélemy (INRIA Rocquencourt)
Damian Chu (Imperial College, London)
Keith Clark (Imperial College, London)
Jürgen Cleve (Universität des Saarlandes, Saarbrücken)
Anatoli Degtyarev (University of Kiev)
Marc Denecker (Catholic University of Leuven)
D. Diaz (INRIA Rocquencourt)
Michel Dorochevsky (ECRC, Munich)
B. Dumant (INRIA, Rocquencourt)
Uwe Egly (Technische Hochschule Darmstadt)
Norbert Eisinger (ECRC, Munich)
C. Fermüller (Teschnische Universität Wien)
Torkel Franzén (SICS, Kista)
Alessandro Giacalone (ECRC, Munich)
Gerd Grosse (Technische Hochschule Darmstadt)
Hans Hansson (SICS, Kista)
Dieter Hutter (Universität des Saarlandes, Saarbrücken)
Nobuyuki Ichiyoshi (ICOT, Tokyo)
Noboru Iwayama (ICOT, Tokyo)
Sverker Janson (SICS, Kista)
Michael Kohlhase (Universität des Saarlandes, Saarbrücken)
Per Kreuger (SICS, Kista)
Hirofumi Kumeno (Mitsubishi Research Institute)
Rainer Manthey (ECRC, Munich)
Gerd Neugebauer (Technische Hochschule Darmstadt)
Remo Pareschi (ECRC, Munich)
Michel Parigot (University of Paris 7)
D. Rayko (Institute for Cybernetics, Kiev)
Igor Romanenko (Institute for Cybernetics, Kiev)
V. Rudenko (Institute for Cybernetics, Kiev)
François Rouaix (INRIA Rocquencourt)
Gernot Salzer (Technische Universität Wien)
Claus Sengler (Universität des Saarlandes, Saarbrücken)
Peter H. Schmitt (Universität Karlsruhe)
Josef Schneeberger (Technische Hochschule Darmstadt)
Bent Thomsen (ECRC, Munich)
Mark Wallace (ECRC, Munich)
F. Winkler (RISC, Linz)
Jörg Würtz (DFKI, Saarbrücken)

Contents

Session 1: Semantics I

Soundness and Completeness of Partial Deductions for Well-Founded Semantics 1
Halina Przymusinska, Teodor Przymusinski, Hirohisa Seki

Session 2: Non-Resolution Theorem Proving I

On Deductive Planning and the Frame Problem 13
Steffen Hölldobler (invited lecture)

On Resolution in Fragments of Classical Linear Logic 30
James Harland, David Pym

A Procedure for Automatic Proof Nets Construction 42
Didier Galmiche, Guy Perrier

Session 3: Constraints

Free Logic and Infinite Constraint Networks 54
James Bowen, Dennis Bahler

Session 4: Data Bases and Knowledge Bases

Towards Probabilistic Knowledge Bases 66
Beat Wüthrich

Two-Level Grammar: A Functional/Logic Query Language for
Database and Knowledge-Base Systems 78
Barrett R. Bryant, Aiqin Pan

Extending Deductive Database Languages by Embedded Implications 84
Burkhard Freitag

Session 5: Resolution Theorem Proving

Controlling Redundancy in Large Search Spaces:
Argonne-Style Theorem Proving Through the Years 96
Ewing L. Lusk (invited lecture)

Resolution for Many-Valued Logics 107
Matthias Baaz, Christian G. Fermüller

An Ordered Theory Resolution Calculus 119
Peter Baumgartner

Application of Automated Deduction to the Search for
Single Axioms for Exponent Groups 131
William McCune, Larry Wos

Session 6: Theorem Proving and Complexity

Elementary Lower Bounds for the Lengths of Refutations 137
Hai-Ping Ko, Mark E. Nadel

Shortening Proofs by Quantifier Introduction 148
Uwe Egly

Session 7: Implementation Aspects

Reform Compilation for Nonlinear Recursion 160
Håkan Millroth

Pruning Infinite Failure Branches in Programs with Occur-Check 172
Ulrich Neumerkel

Session 8: Logical Frameworks

The Use of Planning Critics in Mechanizing Inductive Proofs 178
Andrew Ireland

$\lambda\mu$-Calculus: An Algorithmic Interpretation of Classical Natural Deduction 190
Michel Parigot

Building Proofs by Analogy via the Curry-Howard Isomorphism 202
Thierry Boy de la Tour, Christoph Kreitz

On the Use of the Constructive Omega-Rule Within Automated Deduction 214
Siani Baker, Andrew Ireland, Alan Smaill

Session 9: Parallel Theorem Proving and Logic Programming

OR-Parallel Theorem Proving with Random Competition 226
Wolfgang Ertel

Parallel Computation of Multiple Sets-of-Support 238
Christian B. Suttner

Towards Using the Andorra Kernel Language for
Industrial Real-Time Applications 250
Bogumil Hausman

Session 10: Unification and Equality I

Unification in a Combination of Equational Theories with Shared
Constants and its Application to Primal Algebras 261
Christophe Ringeissen

Non-Clausal Resolution and Superposition with
Selection and Redundancy Criteria 273
Leo Bachmair, Harald Ganzinger

Relating Innermost, Weak, Uniform and Modular Termination
of Term Rewriting Systems 285
Bernhard Gramlich

Session 11: Semantics II

A Two Step Semantics for Logic Programs with Negation 297
Maurizio Gabbrielli, Giorgio Levi, Daniele Turi

Generalized Negation as Failure and Semantics of Normal Disjunctive Logic Programs 309
Chitta Baral

General Model Theoretic Semantics for Higher-Order Horn Logic Programming 320
Mino Bai, Howard A. Blair

Session 12: Extensions of Logic Programming

Disjunctive Deductive Databases 332
José Alberto Fernández, Jack Minker (invited lecture)

Netlog -- A Concept Oriented Logic Programming Language 357
Alexander Voinov

From the Past to the Future: Executing Temporal Logic Programs 369
Michael Fischer, Richard Owens

Session 13: Non-Resolution Theorem Proving II

Computing Induction Axioms 381
Christoph Walther

Session 14: Specification and Verification

Consistency of Equational Enrichments 393
Valentin Antimirov, Anatoli Degtyarev

A Programming Logic for a Verified Structured Assembly Language 403
Paul Curzon

Session 15: Unification and Equality II

The Unification of Infinite Sets of Terms and its Applications 409
Gernot Salzer

Unification in Order-Sorted Type Theory 421
Michael Kohlhase

Infinite, Canonical String Rewriting Systems Generated by Completion 433
Andrea Sattler-Klein

System Descriptions

Spes: A System for Logic Program Transformation 445
Francis Alexandre, Khaled Bsaïes, Jean Pierre Finance, Alain Quéré

Linear Objects: A Logic Framework for Open System Programming 448
Jean-Marc Andreoli, Remo Pareschi

ISAR: An Interactive System for Algebraic Implementation Proofs 451
Bernhard Bauer, Rolf Hennicker

Mathpert: Computer Support for Learning Algebra, Trig and Calculus 454
Michael Beeson

MegaLog — A Platform for Developing Knowledge Base Management Systems 457
Jorge Bocca, Michael Dahmen, Michael Freeston

SPIKE, an Automatic Theorem Prover 460
Adel Bouhoula, Emmanuel Kounalis, Michaël Rusinowitch

An Application to Teaching in Logic Course of ATP
Based on Natural Deduction 463
Li Dafa

A Generic Logic Environment 466
Mark Dawson

ElipSys. A Parallel Programming System Based on Logic 469
Michel Dorochevsky, Liang-Liang Li, Mike Reeve, Kees Schuerman, André Véron

Opium — A High Level Debugging Environment 472
Mireille Ducassé

An Inductive Theorem Prover Based on Narrowing 475
Ulrich Fraus, Heinrich Hussmann

A Cooperative Answering System 478
Terry Gaasterland, Parke Godfrey, Jack Minker, Lev Novik

MIZ-PR: A Theorem Prover for Polymorphic and Recursive Functions 481
Javier Leach, Susana Nieva

ProPre. A Programming Language with Proofs 484
Pascal Manoury, Michel Parigot, Marianne Simonot

FRIENDLY-WAM: An Interactive Tool
to Understand the Compilation of Prolog 487
Julio García Martín, Juan José Moreno-Navarro

SEPIA: A Basis for Prolog Extensions 490
Micha Meier

The External Database in SICStus Prolog 493
Hans Nilsson

The KCM System: Speeding-up Logic Programming
Through Hardware Support 496
Jacques Noyé

Logician's Workbench 499
Igor Romanenko

EUODHILOS: A General Reasoning System for a Variety of Logics 501
Hajime Sawamura, Toshiro Minami, Kyoko Ohashi

The EKS-VI System 504
*Laurent Vielle, Petra Bayer, Volker Küchenhoff, Alexandre Lefebvre,
Rainer Manthey*

CHIP and Propia 507
Mark Wallace, Thierry Le Provost

One of the important semantics introduced recently to logic programming is the so called *well-founded semantics* [VGRS90]. It appears to be a natural semantics for logic programs which extends the *perfect model semantics* of stratified programs, eliminates some of the drawbacks of Clark's predicate completion semantics and, in general, behaves in a more regular fashion (see, e.g., [PP90, PW91]). For *datalog* programs with negation well-founded models can be computed in polynomial (quadratic) time. While SLDNF-resolution is still *sound* with respect to the well-founded semantics, a new resolution procedure, called *SLS-resolution*, has been introduced for well-founded semantics [Prz89a, Ros89] and shown to be *sound and complete* (for non-floundering queries) with respect to this semantics. Recently, D. S. Warren developed an elegant Prolog *meta-interpreter* and introduced the *Extended Warren Abstract Machine* (XWAM) for the well-founded semantics (cf. [PW91]).

We prove that partial deductions based on *SLS-resolution* preserve the *well-founded semantics* of logic programs. More precisely, we show that if P is a program and if P' is obtained from P by an SLS-partial deduction then the well-founded semantics of the initial program P coincides with the well-founded semantics of the derived program P'. This result proves that the *declarative semantics* of logic programs is preserved by SLS-partial deductions and shows that partial deductions based on SLS-resolution can be safely used *without alternating in any way* the original meaning of the program.

Due to the soundness and completeness of SLS-resolution for well-founded semantics, we immediately obtain that SLS-partial deductions also preserve the *procedural semantics* of logic programs, i.e., that the set of SLS-computed answer substitutions obtained from the initial program P is equivalent to the set of SLS-computed answer substitutions obtained from the partially deduced program P'. It is important to point out that in both results we are allowed to choose *arbitrary* computation rules used with programs P and P'.

Analogous results for *Clark's predicate completion semantics* and for partial deductions based on *SLDNF-resolution* are only partially true. Lloyd and Shepherdson proved in [LS87] (see also [Kom81]) that SLDNF-partial deductions preserve the *procedural semantics* of programs, i.e., that the set of SLDNF-computed answer substitutions obtained from the initial program P is equivalent to the set of SLDNF-computed answer substitutions obtained from the partially deduced program P'. This result requires, however, a *careful selection* of, possibly different, computation rules used with programs P and P' and thus is difficult to apply in practice.

On the other hand, as pointed out in [LS87], SLDNF-partial deductions *do not* preserve the *declarative semantics* of programs, i.e., the predicate completion semantics of the initial program P can, in general, be different from the predicate completion semantics of the derived program P'.

Our results underscore one more time the naturality of well-founded semantics and the regularity of its behavior vis-a-vis predicate completion semantics. They show that partial deduction is a completely *safe query optimization procedure* for logic programs with the well-founded semantics.

The invariance of well-founded semantics under partial deductions is proved under the assumption of the so called *$\mathcal{A}$-closedness* of the program, which was originally introduced and used by Lloyd and Shepherdson [LS87]. In the last section we define the notion of *constrained partial deduction* and prove the invariance of well-founded semantics under constrained partial deduction, without the assumption of $\mathcal{A}$-closedness. This result generalizes our previous results.

The paper is organized as follows. In the next section we define the terminology used in the paper, including the notions of SLS-resolution and SLS-partial deduction. In Section 3 we show the invariance of well-founded semantics under SLS-partial deductions. In the last Section 4 we show the invariance of well-founded semantics under constrained partial deductions, without the assumption of $\mathcal{A}$-closedness.

2 Notation and Definitions

Definition 1. By a *logic program* P we mean a finite set of universally quantified clauses of the form

$$A \leftarrow L_1, ..., L_m$$

where $m \geq 0$, A is an atom and L_i's are literals. □

The *alphabet* of a program P consists of all the constant, predicate, variable and function symbols that appear in P and – in addition – the *equality* predicate = (which does not appear in program clauses[4]), infinitely but countably many variable symbols and (possibly) countably many additional constant and/or function symbols. It is assumed that there is at least one constant in the alphabet and that the alphabet also contains the usual punctuation symbols, connectives ($\wedge, \vee, \neg$) and quantifiers ($\exists, \forall$). The *language* L_P of P consists of all the well-formed formulae of the so obtained first order theory.

Throughout this paper we assume the so called *Clark's Equational Theory* axioms (CET) ([Llo87]), i.e., instead of considering the program P itself we consider its extension $CET(P) = P + CET$.

Definition 2. By the *well-founded semantics* $WF(P)$ of a program P we mean the set of all closed formulae (sentences) which hold in all (Herbrand or not) well-founded models of P satisfying the Clark Equality Theory axioms CET (see [VGRS90, PP90]).

If a sentence F belongs to $WF(P)$ then we say that F is implied by the well-founded semantics and we write:

$$WF(P) \models F. \; \square$$

2.1 SLS-resolution

In this section we recall the definition of *SLS-resolution*, defined originally in [Prz89a] (see also [Ros89]) and subsequently modified in [PW91].

Suppose that P is any logic program. By a *goal* G we mean a headless clause $\leftarrow L_1 ,..., L_k$, where k≥0 and L_i's are literals. We also write, $G = \leftarrow Q$, where Q = $L_1 ,..., L_k$ is called a *query*.

[4] The extension to the more general case with equality literals appearing in the premises of program clauses and goals is fairly straightforward and is discussed in Section 4.