PARALLEL LOGIC PROGRAMMING WITH EXTENSIONS

Luís Moniz Pereira and José C. Cunha
Project Coordinators
Departamento de Informática
Universidade Nova de Lisboa
P–2825 Monte da Caparica, Portugal
Luís Damas
Project Coordinator
Laboratório de Inteligência Artificial e
Ciência dos Computadores
Universidade do Porto
R. Campo Alegre 823, 4100 Porto, Portugal

Abstract

A programming language is a tool and a vehicle for applications.

The need to promote the evolution of Prolog towards a more expressive new logic programming language.

A large medium-term national research project.

Extensions to Logic Programming:

new forms of computational reasoning, with explicit negation, constraint programming, parallelism and distribution support.

The resulting language: PROLOPPE, will integrate the above aspects.

A trully efficient implementation:

innovative execution model joint implicit and explicit parallelism, distribution over heterogenous multiple processor architectures.

A wide variety of applications: desision support systems natural language processing diagnosis scheduling, robot cooperation.

This paper is an edited version, by the authors, of part of the original proposal of the PROLOPPE Project which included contributions from C. Damásio, F. Menezes, J. C. Cunha, J. J. Alferes, L. M. Pereira, P. Barahona, P. D. Medeiros, S.P. Abreu, from the Universidade Nova de Lisboa, and A. Matos, A. P. Tomás, F. Silva, J. P. Leal, L. Damas, L. Lopes, M. Filgueiras, N. Moreira, R. Reis and V. S. Costa from the Universidade do Porto, and Miguel Calejo from Servisoft.

1 Introduction

Prolog has a number of limitations both at the language and at the execution levels

Prolog supports a subset of Logic only, it does not include explicit negation in the facts and in the rules.

A language augmenting the expressive and computational power of Prolog

by including explicit negation as well as default negation, exploiting new forms of non-monotonic reasoning handling contradictions and automated methods for belief revision

Implementation of these functionalities requires the support of construtive negation, integrity constraints, and disjunction.

Constraint-based programming methods for will be integrated in the implementation and will support rapid prototyping of efficient applications in multiple domains (e.g. scheduling and time-tabling).

Currently, these applications demand highly specialized programs which are hard to specify and mantain.

Integrating forms of non-monotonic reasoning and numerical constraints in a logic language opens possibilities for innovation in the area of Decision Support Systems.

An efficient implementation including such extensions will be developed: aiming at overcoming several limitations of current logic programming systems, regarding problem solving in non-trivial AI applications, and in distributed AI.

with the development of new techniques concerning the following aspects: optimized Prolog compilation, execution models for the proposed extensions, joint exploitation of implicit and explicit parallelism over heterogeneous multiple processor architectures, and tools for program development with sequential and parallel execution, with support for performance measurement, debugging and visualization.

A diversity of complex application domains requiring a programming system with: great expressiveness declarativity efficiency intelligence in the execution strategies.

2 Semantics for negation

Recently, several authors have underscored the importance of extending logic programming (LP) with a second kind of negation $\neg$, for use in knowledge representation, deductive databases and nonmonotonic reasoning (NMR) [GL90, GL92, Ino91, Kow90, KS90, PW90, PAA91b, PAA91d, PAA92b, PDA93b, PDA93c, PDA93a, PAA93, Wag91]. [BG93] makes an overview of the use of such programs in knowledge representation and NMR. Different semantics for extended logic programs with $\neg$–negation (ELP) have appeared [DR91, GL90, KS90, PA92, PAA91a, PAA92a, Prz90, Prz91a, Sak92, Wag91]. Each of these semantics is a generalization for ELP of either the stable models semantics (SM) [GL88], or the well–founded semantics (WFS) [GRS91] of normal programs.

In [Prz90, Dix91, Dix92] SM and WFS are contrasted, and it is argued that, by its structural properties, WFS is more suitable for an implementation (unlike SM, it is possible to define for WFS both bottom–up and top–down procedures [PAA91c, CW92, Ros92, BD93]). To deal with the problem of floundering, the top–down procedures need to treat negation as failure goals by means of procrastination, common to that of constraints and deterministic priority, and/or constructive methods [TS86, Prz89, KT88, CW92, Ros92, BD93, Bol93].

[AP92] contrasts some of these semantics regarding their use of ¬–negation, where distinct meanings of ¬ are identified (explicit, strong and classical), and argues that, by its properties, explicit negation is preferable. By being a generalization of WFS for ELP which uses explicit negation, WFSX [PA92] appears as a natural candidate for the base semantics of our ELP PROLOPPE implementation. [Alf93] make an extensive study on WFSX, and its relationship with several NMR formalisms. In [AP93b] WFSX is defined in terms of a "logic of belief and provability', and compared with the works of [Che93, MT93, LS93]. This open the way for contradiction removal (cf. below), and for generalizing WFSX to disjunctive programs, in the spirit of [BLM90, Prz91c, Prz91b, Prz93], where several semantics for disjunctive normal programs are defined.

By generalizing LP with either explicit negation or integrity constrains, a new issue arises: how to deal with contradiction. [DR91, Jon91, PA93b, PAA91a, PAA92a] present several proposal for that issue. [AP93c, PA93a] distinguish between two generic approaches to deal with contradiction: one consists in *avoiding* it; the other in *removing* it. The definition on procedures for removing contradiction has been generalized to deal with two valued revisions [PDA93b], and to deal with preference among revisions [DNP93], with application to diagnosis, updates, and debugging [PDA93c].

Despite all the above mentioned effort on the theoretical study on ELP semantic and its application domains, to date there is no efficient implementation of these semantics, nor even a formal specification of their procedures. The ELP implementation of this proposal is intended to fill in this gap, and to allow for a practical application of ELP for problems of the domains studied.

3 Constraint Logic Programming

The characteristics of LP, namely its declarative nature, makes it particularly suitable to the specification of a large number of constraint satisfaction problems. Nevertheless, the resolution principle, the basis of LP, is insufficient to handle efficiently these problems, since it does not take advantage from the specificity of some domains (namely numeric) nor from the characteristics of the operations defined on these domains. Several extensions have thus been proposed to LP in the last few years that, without jeopardizing its declarative nature, allow a much better performance in solving these problems. In general, these languages extend LP to Constraint Logic Programming (CLP), by replacing the resolution principle by more powerful constraint solving method in some specialised domain.

Solving linear constraints on finite domains may also be done by exploring the equivalence to the problem of solving systems of linear equations over the natural numbers (Diophantine equations) and using the specific methods developed for it. Most of the recent research work on Diophantine equations is related with the development of algorithms for unification of terms with associative and commutative functors (AC-unification) and with the field of Term Rewriting Systems [Dom91]. The use in the implementation of a CLP system of one of the methods for solving a system of Diophantine equations is under research [Con93]. Other recent results, for a single equation, are described in [TFar] and [FT93] and correspond to the fastest methods known to date.

The topic of constraints over algebras of rational trees extends term unification, in a decidable way [Mah88, CL89], to the resolution of first order formulas with equality as unique predicate symbol. Extensions to Prolog in this line, are Prolog II [Col82], Prolog III, and more recent, systems as $CLP(\mathcal{FT})$ [Smi91], where universally quantified disequalities are used to allow logic programs with constructive negation. On the other hand, as was pointed out in [DMV93], the standard algebra of rational trees has a close relationship with the standard model for features logics, [Smo89], which were establish in order to formalize feature based grammar formalisms that have emerged in the Computational Linguistics community over the past few years. From a practical point of view, the fact that the satisfiability problem (in these domains) is NP-hard tends to manifest itself in a dramatic way in practical applications, motivated several specialized algorithms to minimize this problem [Kas87, ED88, MK91].

In [DV92] it was argued that any practical approach to the satisfiability problem should use factorization techniques to reduce the size of the input formulae to which any complete algorithm for satisfiability is applied, since such factorization can reduce by an exponential factor the overall computational cost of the process. In [DMV93, DMB93] were described more factorization techniques and a complete rewrite system for satisfiability was provided.

4 Compilers for Prolog with Extensions

Prolog adapts well to conventional computer architectures. Prolog's selection function and search rule are simple operations. Moreover, the fact that Prolog only uses terms means that the state of the computation can be coded quite efficiently. The basis for most of the current implementations of logic programming languages is the Warren Abstract Machine [War83], or *WAM*, an "abstract machine" useful as a target for the compilation of Prolog because it can be implemented very efficiently in most conventional architectures. Recent efforts on the implementation of Prolog have tried to improve further the performance by using direct compilation to native code and global analysis [Van90, Tay90]. Native code systems gain performance by by-passing the emulator. They can also perform machine-level optimisations. Global analysis provides information on how arguments are actually used during execution. Its most common uses are in the further specialisation of unification and in more sophisticated indexing.

The above-mentioned systems are not portable, that is, they usually depend on knowledge of some computer architectures. A portable approach, direct generation of "C" code, has been proposed by Debray among others, but can lead to the generation of huge and hard to compile "C" programs. We believe that such work could benefit from well-established work on portable C compilers.

4.1 New execution models

From the very beginnings of logic programming there has been a desire to obtain good execution models [Kow79] [PP79] [Col86] [Nai85]. More recently the need for such models has been made even clearer by the new goal of exploring parallelism in logic programs. One such important model is the Warren's Basic Andorra Model [War88], used in the the Andorra-I system [SCWY91b, SCWY91c]. In this model determinate goals (that is, goals for which at most a single clause can match) can be selected first and run in and-parallel. When no such goals are available, the system can try the several alternatives to a non-determinate goal in or-parallelism. Besides the parallelism, the selection functions most natural to the Basic Andorra Model have a very useful form of implicit coroutining [SC93], which has been exploited in several Andorra-I applications [Yan89, GY92] and in the Pandora language [Bah93]. Note that Andorra-I can only exploit and-parallelism between determinate goals. Warren's Extended Andorra Model (EAM) [War89] lifts this restriction and allows a general form of and-parallelism. The EAM gives a set of general rewrite rules for logic programs, which can be subject to different control schemes. The EAM was a basis for the Kernel Andorra Prolog (KAP) [HJ90] framework which is instantiated in the AKL language, proposed by Janson and Haridi [JH91]. In these languages, guards (such as commit guards, cut guards and wait guards) are used to control computation, which may be nondeterministic. Both or-parallelism, and and-parallelism between non-determinate (and determinate) goals can be exploited. Moreover, the search space can be much reduced over traditional Prolog systems.

Further improvements to AKL's search rule have been performed by Abreu, Pereira and Codognet [APC92a]. The authors have studied failure-driven configuration reordering, which can be seen as an application of the first-fail principle to the unfolding of an AKL computation. This shows that And-Or tree Rewriting systems (AORS), which encompasses both AKL and the EAM, provide a fertile base for the exploitation of a-posteriori search-space pruning, i.e. pruning part of the search-space as a consequence of the execution of another portion of the program. This approach complements the a-priori search-space pruning that comes as a result of constraint propagation, another mechanism present in AKL.

The differences between Prolog and Andorra-I are more striking. Andorra-I does in fact inherit most of its implementation techniques from Parlog [Cra88] and KL1 [SSM+87]. Andorra-I's abstract machine and compiler are described by Santos Costa [SC93] (note that in practice much of the difficulties to be addressed in Andorra-I are due to parallelism support, handled by Yang's engine and by the several schedulers[BRSW91, Dut91]). Andorra-I incorporates some optimisations, the ones considered particularly important for Andorra-I's main goal to run real applications. The compiled Andorra-I is not as optimised as current Prolog systems, being a more complex and a newer system. Great improvements can be obtained by using the new techiques that are being developed for Prolog, plus the new techiques developed for the committed-choice languages [TB93].

The implementation of AKL and of the EAM also brings some new problems. Janson and Montelius have a prototype implementation [JM92], but again several optimisations will be needed for one such system to compete with current Prolog systems. Note that AKL (and EAM) can be described in terms of and-boxes, and or-boxes (several types may exist). These boxes are expanded during forward execution, but their configuration must be reorded upon failure. This can be made more effective through the guidance of the reordering scheme by a binding-dependency maintenance system; such is the object of the AKL/IP (for AKL with Intelligent Pruning) system outlined in [AP93a] and [APC92b], currently being worked on in Lisbon. AKL/IP is currently being implemented using Janson and Montelius' prototype [JM92] as a basis, being thus a sequential implementation. It can be argued that a computational model based on rewrite rules for and-or trees (as is the case with AORS's) is more suited to dependency-directed search-space pruning than systems using a Prolog-like selection rule, because the former provides a built-in[1] mechanism to describe suspension of goals and can cope more gracefully with changes to the relative ordering of goals at run-time.

5 Implicit and Explicit Parallelism

Parallel logic programming systems obtain high-performance by exploiting different forms of parallelism. Both implicit and explicit parallelism are available in logic programming languages. In explicit systems such as Delta Prolog [PMCA86] special types of goals (events and splits in Delta Prolog) are available to control parallelism. Implicit parallelism can be obtained through the parallel execution of several resolvents arising from the same query, *or-parallelism*, or through the parallel resolution

[1] i.e. intrinsic to the execution model, not as an add-on such as can be found in some Prolog systems, or in multi-sequential parallel implementations.

of several goals, *and-parallelism*. All these forms of parallelism can be explored according to very different strategies. We next discuss the most important techniques now available to exploit parallelism in logic programs.

5.1 Implicit Parallelism

Both or-parallelism and and-parallelism have been exploited successfully in logic programming systems. Whereas or-parallel systems exploit much the same parallelism, and differ mainly in the way they represent the search space, quite a few different forms of and-parallelism have been recognised. Arguably, some of the most important approaches are the following. Systems implementing the committed-choice languages, exploit parallelism between goals that have commited to a single clause. Independent and-parallelism systems, such as &-Prolog [HG90], only run independent goals, whose computation should not interfere, in parallel. Andorra-I [SCWY91a] exploits and-parallelism between goals that are *determinate*. The latter idea has been generalised in the Extended Model, and in the AKL language[JH91].

5.2 Parallel Implementation of the EAM and AKL

Andorra-I can only exploit and-parallelism between determinate goals. Both the EAm and AKL lift this restriction. AKL and the EAM thus share the property that the computation may be carried out in parallel more naturally than in other parallel logic programming systems: the approach of having the run-time data structures organized as a tree of and-boxes and choice-boxes, the requirements on quietness of pruning guards, together with the fact that and-boxes have their own store lead to potentially better locality properties, making such a system suitable for both coarse and fine-grained parallelism. Indeed, the gains on a parallel implementation of AKL are very attractive, and Moolenaar[VAMD91] has recently implemented a parallel prototype for of AKL.

Finally, the promising experimental results obtained with the preliminary prototype of the AKL/IP system prompt us to look into a true parallel implementation.

5.3 Implicit Parallelism for Distributed Memory Systems

Recently, new parallel architectures, namely distributed shared memory architectures, have been proposed and built (e.g. the KSR and EDS parallel machines). Although, in these architectures, the memory is physically distributed there is software and hardware support for a shared virtual memory computation model. These architectures combine the advantages of large number of processors (scalability) with the advantage of shared memory, and are therefore ideal targets for the parallel execution of Prolog programs. These architectures are quite recent and few parallel logic programming systems have been developed that understand the new issues.

One of the first models has been designed and implemented by Silva [Sil93], and shown successful execution for or-parallelism in one of these architectures.

5.4 Explicit Parallelism

This approach consists in the definition of constructs for the explicit specification of sequenciality and concurrency, synchronization and non-determinism in a logic programming language.

The diversity of proposals that have arisen in the past ten years have two principal aims: (i) search for increased flexibility in the specification of parallelism, versus implicit parallelism as supported by a compiler and/or run-time system; (ii) the need for suitable constructs for the specification of distributed systems. A large number of problems are naturally modelled by multiple concurrent interacting processes where the data structures and/or the entities that solve the problem are spacially distributed.

6 Programming environment

Mostly since the 80s several efforts have been made to build powerful logic programming environments, in order to meet the expectations brought by the innovative Prolog language in the 70s [DCLY93]. One of the areas where more promising results were found, by capitalizing on logic programming's own caracteristics, was declarative debugging, which has grown into an area with autonomous scientific workshops, such as the recent [FN93], [Cal92]. So far one of the greatest eforts towards an environment integrating innovative tools has been the ESPRIT ALPES project [ANPR89], which went on between 1986 and 1988. From that work and from its sequel by the UNL team several prototype Prolog environments were developped (X-Prolog, MacLogic, StepOnProlog), associated to various Prolog implementations (Apple Computer's Logic Manager, UNL's own nanoProlog, Universidade do Porto's YAP, University of Edinburgh's C-Prolog), sometimes in the context of external R&D contracts (Apple, ENIDATA, Digital, Softlog/NeXT, among others).

In this project we are further improving the development tools in the environment, including low-level analyzers for sequential and parallel execution, declarative debugging, browsers, graphics tools and interface buiders.

7 Tasks Overview

This project represents a significant effort, at national scale, towards promoting the Logic Programming paradigm in several directions: theory, language, execution models, implementations and applications. It spans a large body of researchers, and it will stimulate a diversity of teaching and training activities.

Besides the semantic definitions for the new PROLOPPE language, we will produce the first usable implementation of the language and a development environment to be made available to the international academic community, promoting the exploitation of AI applications. This implementation will include and extend the results that the proponents have been achieving, regarding a better Prolog execution model, applied to the support of more advanced semantics and a joint exploitation of implicit and explicit parallelism. The tasks below will produce specifications of language extensions, models and prototypes, and the prototypes themselves, which will be demonstrated and made available during the project.

- Explicit Negation and Logic Programming with Non-Monotonic Reasonning, including contradiction removal, with constructive negation and disjunction.

- Constraint Logic Programming: constraint resolution methods are investigated and used in the implementation of constraint logic programming languages: (a) resolution methods for the linear Diophantines equation systems and other more general constraints concerning naturals and finite domains; (b) incremental hierarchical constraint solvers over natural and finite domains.

- Execution Models: optimized compilation of the Prolog model will be explored, since our past experience in the implementation of conventional Prolog shows that great improvements can still be achieved when implementing the language. This includes: (a) the design of an Intermediate Computer Description, oriented to the underneath architecture; (b) the implementation of a compiler with procedure and intra-procedure level optimization, featuring unfolding, choice points elimination and mode or sequentiality detection; (c) extensibility support to provide the proposed extensions to the logic language.

- We will develop other execution models and search strategies in the following aspects: (a) optimization of the search process based in "Intelligent Pruning", a method that resembles Prolog's intelligent backtracking, but applied to AKL (AKL/IP) execution model; (b) sequential and parallel implementation of the AKL/IP language; (c) application of the AKL/IP execution model to non-monotonic reasoning, as support to an implementation support of the previous extensions.

- Implicit and Explicit Parallelism: implicit parallelism of the OR and AND types will be explored over shared memory and distributed memory architectures, integrated with other forms of parallelism suitable to the support of distributed logic programming, its application in Distributed AI, and its implementation over heterogenous multiprocessors.

- Development Environment: we will integrate the acomplished extensions in an environment with a set of user support tools, such as: (a) low-level analyser with performance measuring tools, and sequential and parallel execution tracing; (b) declarative debugger; (c) browser; (d) graphic library and specification languages for system interaction; (e) visualization of distributed computations.

- Some applications will be developed to evaluate, test, promote PROLOPPE: (a) syntactic analyser for natural language (for the testing of system use in new formalisms based on constraints like the HPSG); (b) Constraints and Time-Tabling; (c) To diagnosis of distributed artificial intelligence and non–monotonic reasoning.

  Other aplications of non–monotonic reasoning are forseen, e.g. to planning and to natural language, not carried out within the project, but evaluated by institutional colleges of team members. The results of this evalutation will be report.

Acknowledgments. This research is nationally supported in part by Programa CIENCIA and project PROLOPPE of the PRAXIS XXI programme.

References

[Alf93] José Júlio Alferes. *Semantics of Logic Programs with Explicit Negation*. PhD thesis, Universidade Nova de Lisboa, October 1993.

[ANPR89] J.A.S. Alegria, A. Natali, N. Preston, and C. (editors) Ruggieri. Alpes final report (esprit p973 project). Technical report, Universidadade Nova de Lisboa, 1989.

[AP92] J. J. Alferes and L. M. Pereira. On logic program semantics with two kinds of negation. In K. Apt, editor, *Int. Joint Conf. and Symp. on LP*, pages 574–588. MIT Press, 1992.

[AP93a] Salvador Abreu and Luís Moniz Pereira. Design for akl with intelligent pruning. In Roy Dyckhoff, editor, *Workshop on Extensions of Logic Programming*, pages 1–6. University of St. Andrews, Scotland, 1993.

[AP93b] J. J. Alferes and L. M. Pereira. Belief, provability, and logic programs (draft). Technical report, CENTRIA, 1993. Submitted to KR'94.

[AP93c] J. J. Alferes and L. M. Pereira. Contradiction: when avoidance equal removal. Part I. In R. Dyckhoff, editor, *4th Int. Ws. on Extensions of LP*, pages 7–16. Univ. of St. Andrews, 1993.

[APC92a] Salvador Abreu, Luís Moniz Pereira, and Philippe Codognet. Improving backward execution in the andorra family of languages. In Krzysztof Apt, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming*, pages 384–398, Washington, USA, 1992. The MIT Press.

[APC92b] Salvador Abreu, Luís Moniz Pereira, and Philippe Codognet. Improving Backward Execution in Non-deterministic Concurrent Logic Languages. In Ryuzo Hasegawa and Mark Stickel, editors, *Workshop on Automated Deduction, Fifth Generation Computer Systems*. Institute for New Generation Computing, 1992.

[Bah93] Reem Bahgat. *Non-Deterministic Concurrent Logic Programming in Pandora*. World Scientific, 1993.

[BD93] R.N. Bol and L. Degerstedt. Tabulated resolution for well–founded semantics. In *Proc. Int. Logic Programming Symposium'93*. MIT Press, 1993.

[BG93] C. Baral and M. Gelfond. Logic programming and knowledge representation (draft). Technical report, University of Texas at El Paso, 1993.

[BLM90] C. Baral, J. Lobo, and J. Minker. Generalized disjunctive well–founded semantics. In C. Delobel, M. Kifer, and Y. Masunaga, editors, *10th Int. Conf. on Automated Deduction*, pages 102–116. LNAI 449, Spriger–Verlag, 1990.

[Bol93] R.N. Bol. Loop checking and negation. *Journal of Logic Programming*, 15(2):147–175, 1993.

[BRSW91] Anthony Beaumont, S Muthu Raman, Péter Szeredi, and David H. D. Warren. Flexible Scheduling of OR-Parallelism in Aurora: The Bristol Scheduler. In *PARLE91: Conference on Parallel Architectures and Languages Europe*, volume 2, pages 403–420. Springer Verlag, June 1991.

[Cal92] Miguel Calejo. *A Framework for Declarative Prolog Debugging*. PhD thesis, Universidade Nova de Lisboa, March 1992.

[Che93] J. Chen. Minimal knowledge + negation as failure = only knowing (sometimes). In L. M. Pereira and A. Nerode, editors, *2nd Int. Ws. on LP & NMR*, pages 132–150. MIT Press, 1993.

[CL89] H. Comon and Pierre Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7:317–425, 1989.

[Col82] Alain Colmerauer. Prolog and infinite trees. In S. A. Tarnlund, editor, *Logic Programming*. AC, 1982.

[Col86] Alain Colmerauer. Theoretical Model of Prolog II. In Michel van Caneghen and David H. D. Warren, editors, *Logic Programming and its Applications*, pages 3–31. Ablex Publishing Corporation, 1986.

[Con93] Evelyne Contejean. Solving linear diophantine constraints incrementally. In D. S. Warren, editor, *Logic Programming: Proceedings of the 10th International Conference on Logic Programming*. MIT Press, 1993.

[Cra88] J. A. Crammond. *Implementation of Committed Choice Logic Languages on Shared Memory Multiprocessors*. PhD thesis, Heriot-Watt University, Edinburgh, May 1988. Research Report PAR 88/4, Dept. of Computing, Imperial College, London.

[CW92] W. Chen and D. S. Warren. A goal–oriented approach to computing well–founded semantics. In K. Apt, editor, *Int. Joint Conf. and Symp. on LP*, pages 589–603. MIT Press, 1992.

[DCLY93] M. Ducassé, B. Charlier, Y. Lin, and Ü. Yalcinalp, editors. *Post Conf. Ws. on Logic Programming Environments*. ILPS'93, 1993.

[Dix91] J. Dix. Classifying semantics of logic programs. In A. Nerode, W. Marek, and V. S. Subrahmanian, editors, *LP & NMR*, pages 166–180. MIT Press, 1991.

[Dix92] J. Dix. A framework for representing and characterizing semantics of logic programs. In B. Nebel, C. Rich, and W. Swartout, editors, *3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann, 1992.

[DMB93] Luís Damas, Nelma Moreira, and Sabine Broda. Resolution of constraints in algebras of rational trees. In Miguel Filgueiras and Luís Damas, editors, *Progress in Artificial Intelligence — 6th Portuguese Conference on Artificial Intelligence*, volume 727 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1993.

[DMV93] Luís Damas, Nelma Moreira, and Giovanni B. Varile. The formal and computational theory of complex constraint solution. In C. Rupp, M. Rosner, and R. Johnson, editors, *Constraints, Language, and Computation*. Academic Press, London, 1993.

[DNP93] C. V. Damásio, W. Nejdl, and L. M. Pereira. REVISE: An extended logic programming system for revising knowledge bases (draft). Technical report, CENTRIA and RTHW–Aachen, 1993. Submitted to KR'94.

[Dom91] Eric Domenjoud. *Outils pour la Déduction Automatique dans les Théories Associatives-Commutatives*. PhD thesis, Université de Nancy I, 1991.

[DR91] P. M. Dung and P. Ruamviboonsuk. Well founded reasoning with classical negation. In A. Nerode, W. Marek, and V. S. Subrahmanian, editors, *LP & NMR*, pages 120–132. MIT Press, 1991.

[Dut91] Inês Dutra. A Flexible Scheduler for the Andorra-I System. In *LNCS 569, ICLP'91 Pre-Conference Workshop on Parallel Execution of Logic Programs*, pages 70–82. Springer-Verlag, June 1991.

[DV92] Luís Damas and Giovanni B. Varile. On the satisfiability of complex constraints. In *Coling'92*, Nantes, France, 1992.

[ED88] A. Eisele and J. Dörre. Unification of disjunctive feature descriptions. In *26th Annual Meeting of the Association for Computational Linguistics*, Buffalo, New York, 1988.

[FN93] P. Fritzson and H. Nilsson, editors. *1st Int. Ws. on Automatic Algorithmic Debugging, AADEBUG'93.* Preproceedings by Linkoping Univ., 1993. To appear in Springer–Verlag LNCS.

[FT93] Miguel Filgueiras and Ana Paula Tomás. Fast methods for solving linear Diophantine equations. In M. Filgueiras and L. Damas, editors, *Proceedings of the 6th Portuguese Conference on Artificial Intelligence*, pages 297–306. LNAI 727, Springer–Verlag, 1993.

[GL88] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. A. Bowen, editors, *5th Int. Conf. on LP*, pages 1070–1080. MIT Press, 1988.

[GL90] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In Warren and Szeredi, editors, *7th Int. Conf. on LP*, pages 579–597. MIT Press, 1990.

[GL92] M. Gelfond and V. Lifschitz. Representing actions in extended logic programs. In K. Apt, editor, *Int. Joint Conf. and Symp. on LP*, pages 559–573. MIT Press, 1992.

[GRS91] A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.

[GY92] Steve Gregory and Rong Yang. Parallel Constraint Solving in Andorra-I. In *International Conference on Fifth Generation Computer Systems 1992*, pages 843–850. ICOT, Tokyo, Japan, June 1992.

[HG90] M. V. Hermenegildo and K. Greene. &-Prolog and its Performance: Exploiting Independent And-Parallelism. In *Proceedings of the Seventh International Conference on Logic Programming*, pages 253–268. MIT Press, June 1990.

[HJ90] Seif Haridi and Sverker Jansson. Kernel Andorra Prolog and its Computational Model. In D.H.D. Warren and P. Szeredi, editors, *Proceedings of the Seventh International Conference on Logic Programming*, pages 31–46. MIT Press, 1990.

[Ino91] K. Inoue. Extended logic programs with default assumptions. In Koichi Furukawa, editor, *8th Int. Conf. on LP*, pages 490–504. MIT Press, 1991.

[JH91] Sverker Janson and Seif Haridi. Programming Paradigms of the Andorra Kernel Language. In *Logic Programming: Proceedings of the International Logic Programming Symposium*, pages 167–186. MIT Press, October 1991.

[JM92] Sverker Janson and Johan Montelius. Design of a Sequential Prototype Implementation of the Andorra Kernel Language. Sics research report, in preparation, Swedish Institute of Computer Science, 1992.

[Jon91] K. Jonker. On the semantics of conflit resolution in truth maintenance systems. Technical report, Univ. of Utrecht, 1991.

[Kas87] R. T. Kasper. Unification method for disjunctive feature descriptions. In *27th Annual Meeting of the Association for Computational Linguistics*, Standford, CA, 1987.

[Kow79] Robert A. Kowalski. *Logic for Problem Solving.* Elsevier North-Holland Inc., 1979.

[Kow90] R. Kowalski. Problems and promises of computational logic. In John Lloyd, editor, *Computational Logic*, pages 1–36. Basic Research Series, Springer–Verlag, 1990.

[KS90] R. Kowalski and F. Sadri. Logic programs with exceptions. In Warren and Szeredi, editors, *7th Int. Conf. on LP*. MIT Press, 1990.

[KT88] D. B. Kemp and R. W. Topor. Completeness of a top–d query evaluation procedure for stratified databases. In Kowalski and Bowen, editors, *Proc. of the Fifth Int. Conf. and Symposium on Logic Programming*, pages 178–194. ALP, MIT Press, 1988.

[LS93] V. Lifschitz and G. Schwarz. Extended logic programs as autoepistemic theories. In L. M. Pereira and A. Nerode, editors, *2nd Int. Ws. on LP & NMR*, pages 101–114. MIT Press, 1993.

[Mah88] Michael J. Maher. Complete axiomatizations of the algebras of finite, rational and infinite trees. Technical report, IBM Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, U.S.A., 1988.

[MK91] John T. Maxwell and Ronald M. Kaplan. A method for disjunctive constraint satisfaction. In Massaru Tomita, editor, *Current Issues in Parsing Technology.* Kluwer Academic Publishers, 1991.

[MT93] V. Marek and M. Truszczynski. Reflexive autoepistemic logic and logic programming. In L. M. Pereira and A. Nerode, editors, *2nd Int. Ws. on LP & NMR*, pages 115–131. MIT Press, 1993.

[Nai85] Lee Naish. *Negation and Control in Prolog.* Lecture notes in Computer Science 238. Springer–Verlag, 1985.

[PA92] L. M. Pereira and J. J. Alferes. Well founded semantics for logic programs with explicit negation. In B. Neumann, editor, *European Conf. on AI*, pages 102–106. John Wiley & Sons, 1992.

[PA93a] L. M. Pereira and J. J. Alferes. Contradiction: when avoidance equal removal. Part II. In R. Dyckhoff, editor, *4th Int. Ws. on Extensions of LP*, pages 17–26. Univ. of St. Andrews, 1993.

[PA93b] L. M. Pereira and J. J. Alferes. Optative reasoning with scenario semantics. In D. S. Warren, editor, *10th Int. Conf. on LP*, pages 601–615. MIT Press, 1993.

[PAA91a] L. M. Pereira, J. J. Alferes, and J. N. Aparício. Contradiction Removal within Well Founded Semantics. In A. Nerode, W. Marek, and V. S. Subrahmanian, editors, *LP & NMR*, pages 105–119. MIT Press, 1991.

[PAA91b] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Counterfactual reasoning based on revising assumptions. In Ueda and Saraswat, editors, *Int. LP Symp.*, pages 566–577. MIT Press, 1991.

[PAA91c] L. M. Pereira, J. N. Aparício, and J. J. Alferes. A derivation procedure for extended stable models. In *Int. Joint Conf. on AI*. Morgan Kaufmann, 1991.

[PAA91d] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Nonmonotonic reasoning with well founded semantics. In Koichi Furukawa, editor, *8th Int. Conf. on LP*, pages 475–489. MIT Press, 1991.

[PAA92a] L. M. Pereira, J. J. Alferes, and J. N. Aparício. Contradiction removal semantics with explicit negation. In *Applied Logic Conf.* Preproceedings by ILLC, Amsterdam, 1992. To appear in Springer–Verlag LNAI.

[PAA92b] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Logic programming for nonmonotonic reasoning. In *Applied Logic Conf.* Preproceedings by ILLC, Amsterdam, 1992. To appear in Springer–Verlag LNAI.

[PAA93] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Non–monotonic reasoning with logic programming. *Journal of LP. Special issue on Nonmonotonic reasoning*, 17(2), 1993.

[PDA93a] L. M. Pereira, C. Damásio, and J. J. Alferes. Debugging by diagnosing assumptions. In P. Fritzson and H. Nilsson, editors, *1st Int. Ws. on Automatic Algorithmic Debugging, AADEBUG'93*. Preproceedings by Linkoping Univ., 1993. To appear in Springer–Verlag LNCS.

[PDA93b] L. M. Pereira, C. Damásio, and J. J. Alferes. Diagnosis and debugging as contradiction removal. In L. M. Pereira and A. Nerode, editors, *2nd Int. Ws. on LP & NMR*, pages 316–330. MIT Press, 1993.

[PDA93c] L. M. Pereira, C. Damásio, and J. J. Alferes. Diagnosis and debugging as contradiction removal in logic programs. In M. Filgueiras and L. Damas, editors, *6th Portuguese AI Conf.* Springer–Verlag, 1993.

[PMCA86] Luís Moniz Pereira, Luís Monteiro, José Cunha, and Joaquim N. Aparício. Delta Prolog: a distributed backtracking extension with events. In Ehud Shapiro, editor, *Third International Conference on Logic Programming, London*, pages 69–83. Springer-Verlag, 1986.

[PP79] L. M. Pereira and A. Porto. Intelligent Backtracking and Sidetracking in Horn Clause Programs - the Theory. Report 2/79, Departamento de Informatica, Universidade Nova de Lisboa, October 1979.

[Prz89] T. Przymusinski. Every logic program has a natural stratification and an iterated fixed point model. In *8th Symp. on Principles of Database Systems*. ACM SIGACT-SIGMOD, 1989.

[Prz90] T. Przymusinski. Extended stable semantics for normal and disjunctive programs. In Warren and Szeredi, editors, *7th Int. Conf. on LP*, pages 459–477. MIT Press, 1990.

[Prz91a] T. Przymusinski. A semantics for disjunctive logic programs. In Loveland, Lobo, and Rajasekar, editors, *ILPS'91 Ws. in Disjunctive Logic Programs*, 1991.

[Prz91b] T. Przymusinski. Stable semantics for disjunctive programs. *New Generation Computing*, 9:401–424, 1991.

[Prz91c] T. Przymusinski. Stationary semantics for normal and disjunctive programs. In C. Delobel, M. Kifer, and Y. Masunaga, editors, *2nd Int. Conf. on DOOD*. LNCS 566, Spriger–Verlag, 1991.

[Prz93] T. Przymusinski. Static semantics for normal and disjunctive programs. Technical report, Dep. of Computer Science, Univ. of California at Riverside, 1993.

[PW90] D. Pearce and G. Wagner. Reasoning with negative information I: Strong negation in logic programs. In L. Haaparanta, M. Kusch, and I. Niiniluoto, editors, *Language, Knowledge and Intentionality*, pages 430–453. Acta Philosophica Fennica 49, 1990.

[Ros92] K. Ross. A procedural semantics for well–founded negation in logic programs. *Journal of Logic Programming*, 13(1):1–22, 1992.

[Sak92] C. Sakama. Extended well–founded semantics for paraconsistent logic programs. In *Fifth Generation Computer Systems*, pages 592–599. ICOT, 1992.

[SC93] V. Santos Costa. *Compile-Time Analysis for the Parallel Execution of Logic Programs in Andorra-I*. PhD thesis, University of Bristol, August 1993.

[SCWY91a] V. Santos Costa, D. H. D. Warren, and R. Yang. Andorra-I: A Parallel Prolog System that Transparently Exploits both And- and Or-Parallelism. In *Third ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming PPOPP*, pages 83–93. ACM press, April 1991. SIGPLAN Notices vol 26(7), July 1991.

[SCWY91b] V. Santos Costa, D. H. D. Warren, and R. Yang. The Andorra-I Engine: A parallel implementation of the Basic Andorra model. In *Proceedings of the Eighth International Conference on Logic Programming*, pages 825–839. MIT Press, June 1991.

[SCWY91c] V. Santos Costa, D. H. D. Warren, and R. Yang. The Andorra-I Preprocessor: Supporting full Prolog on the Basic Andorra model. In *Proceedings of the Eighth International Conference on Logic Programming*, pages 443–456. MIT Press, June 1991.

[Sil93] Fernando M. A. Silva. *An Implementation of Or-Parallel Prolog on a Distributed Shared Memory Architecture*. PhD thesis, Dept. of Computer Science, Univ. of Manchester, September 1993.

[Smi91] Donald Smith. Constraint operations for clp($\mathcal{FT}$). In Koichi Furukawa, editor, *Logic Programming: Proceedings of the 8th International Conference*. MIT Press, 1991.

[Smo89] Gert Smolka. Feature constraint logics for unification grammars. Technical report, IBM Wissenschafliches Zentrum, Institut für Wissensbasierte Systeme, 1989. IWBS Report 93.

[SSM+87] M. Sato, H. Shimizu, A. Matsumoto, K. Rokusawa, and A. Goto. KL1 Execution Model for PIM Cluster with Shared Memory. In Jean-Louis Lassez, editor, *Proceedings of the Fourth International Conference on Logic Programming*, MIT Press Series in Logic Programming, pages 338–355. University of Melbourne, "MIT Press", May 1987.

[Tay90] A. Taylor. LIPS on a MIPS: Results from a Prolog Compiler for a RISC. In *Proceedings of the Seventh International Conference on Logic Programming*, pages 174–185. MIT Press, June 1990.

[TB93] E. Tick and C. Banerjee. Performance evaluation of Monaco compiler and runtime kernel. In *ICLP93*, pages 757–773, 1993.

[TFar] Ana Paula Tomás and M. Filgueiras. Solving linear diophantine equations. *Reports of the Institute of Cybernetics, Tallinn*, to appear.

[TS86] H. Tamaki and T. Sato. Old resolution with tabulation. In *Proc. of 3rd International Conference on Logic Programming*, pages 84–98, 1986.

[VAMD91] Henk Van Acker, Remco Moolenaar, and Bart Demoen. A parallel implementation of AKL. Presented at the ILPS workshop on Parallel Logic Programming, October 1991.

[Van90] P. Van Roy. *Can Logic Programming Execute as Fast as Imperative Programming?* PhD thesis, University of California at Berkeley, November 1990.

[Wag91] G. Wagner. A database needs two kinds of negation. In B. Thalheim, J. Demetrovics, and H-D. Gerhardt, editors, *Mathematical Foundations of Database Systems*, pages 357–371. LNCS 495, Springer–Verlag, 1991.

[War83] David H. D. Warren. An Abstract Prolog Instruction Set. Technical Note 309, SRI International, 1983.

[War88] David H. D. Warren. The Andorra model. Presented at Gigalips Project workshop, University of Manchester, March 1988.

[War89] David H. D. Warren. Extended Andorra model. PEPMA Project workshop, University of Bristol, October 1989.

[Yan89] Rong Yang. Solving Simple Substitution Ciphers in Andorra-I. In *Proceedings of the Sixth International Conference on Logic Programming*, pages 113–128. MIT Press, June 1989.