Theorem Proving via Uniform Proofs

by

Alberto Momigliano

September 1993

Report CMU-PHIL-41

Philosophy
Methodology
Logic

Pittsburgh, Pennsylvania 15213-3890

Theorem Proving via Uniform Proofs[1]

Alberto Momigliano
Department of Philosophy
Carnegie Mellon University
15213, Pittsburgh, PA
mobile@lcl.cmu.edu

Abstract

Uniform proofs systems have recently been proposed [Mil91] as a proof-theoretic foundation and generalization of logic programming. In [Mom92a] an extension with constructive negation is presented preserving the nature of abstract logic programming language. Here we adapt this approach to provide a complete theorem proving technique for minimal, intuitionistic and classical logic, which is totally goal-oriented and does not require any form of ancestry resolution. The key idea is to use the Gödel-Gentzen translation to embed those logics in the syntax of Hereditary Harrop formulae, for which uniform proofs are complete. We discuss some preliminary implementation issues.

1. Introduction

In this paper we present a novel approach to general theorem-proving that exploits the notion of uniform proof, introduced by [Mil91] as a proof-theoretic foundation and generalization of logic programming. Uniform proofs formalize the notion of goal-oriented, syntax-directed derivations: every time a compound goal is inferred, its main connective is introduced. If operations on atoms are restricted to backchaining, we obtain a language that very elegantly extends Prolog.

Whilst uniform proofs have extensively been studied and used in the logic programming environment, at first sight they seem unfit to form the basis of theorem-proving in full (classical) logic, since their syntax is restricted to Hereditary Harrop formulae ('fohh' in their first-order incarnation, which is the object of this paper); moreover, the provability relation involved is essentially intuitionistic (if not minimal).

---

[1]The materials contained in this paper have been presented in several seminars at the Department of Engineering, University of Salerno, Department of Philosophy, University of Bologna, Department of Computer Science, University of Milano, Italy and at the Summer School on "Proofs and Computations", Marktoberdorf, Germany.

The philosophy advocated here is that general theorem proving, as resolution with subsumption [Wos91], may not be effective for every class of problems. For certain classes of formulae, specialized procedure may be more productive. This is the case, most of the times, with Horn logic, which admits the very efficient goal-oriented, totally input procedure that underlies Prolog. We aim to extend this approach first to fohh. Then we would like to follow the spirit of Stickel's PTTP [Sti88] and Loveland's Near-Horn Prolog [Lov88], enlarging the deductive power of the calculus, without losing the uniformity property. This is achieved in two steps:

1. The original formulation was deprived of any form of negation. In [Mil89] and [Mom92a] an extension with constructive negation in the sense of [Tro88] is presented preserving the nature of abstract logic programming language. It comes in three flavors: minimal negation, which requires essentially no extension to the operational semantics of uniform proofs, intuitionistic and classical negation: the latter instead involve a significant departure to the way we are inclined to view logic programming languages, since it introduces a new way of proving atoms, namely by deriving a contradiction.

2. The final ingredient is then to embed full logic in fohh with negation through the Gödel-Gentzen negative translation [Tro88]. We thus obtain a completely input, goal-oriented procedure in the style of logic programming for minimal, intuitionistic and classical logic, with no need of factoring or any form of ancestry resolution, at the cost of essentially incorporating the law of "reductio ad absurdum" (RAA), in the terminology of [Tro88], though restricted to atoms.

To be more precise, the negative translation maps classical and intuitionistic derivability into minimal, for which no extension to uniform proofs is required. Nonetheless, this entails double-negating every atom. This is equivalent, from a logical point, to having RAA as a new rule to be applied only to atoms. Moreover the latter can, in some circumstances, be less redundant from a procedural standpoint. Hence in this paper we shall implement classical logic through uniform proofs with RAA.

Historically most of research on theorem-proving has concentrated on methods to derive contradictions from set of clauses [Rob65]. While this may be suitable in many cases, as pointed out for instance in [And81], the distribution laws may create combinatorial explosions in the number of clauses (see for example Andrews' Challenge, problem 34 in [Pel86]). Although these redundancies can be partly eliminated with the introduction of new

predicate symbols, the original structure of the goal with all the information contained therein is irremediably lost. On the other hand, non-clausal methods [Ble77], for instance based on natural deduction, are experiencing different kind of riddles. As far as the latter is concerned, a significant part of the problem lies in the management of the v-E and $\exists$-E rules, due to the well-known problems with permutability [Kle52]; for example, while the theory in [Sie92] dictates the use of the classical v-E, the implementation uses a form of resolution. This is not the case, since the pure fragment of natural deduction (&, $\supset$,$\forall$) can be dealt within the framework of logic programming, as already implicit in [Gab84].

Our approach aims to find a convincing balance between the two; due to the rather generous syntax of fohh, we need a very limited amount of clausification: namely removing from the formulae in the set of axioms disjunction and existential in favor of conjunction, universal and negation by way of the well-known equivalencies. The structure of the proof remains as close as possible to those in logic programming (i.e., it is a uniform proof), with the notable difference that atoms are proved not only through backchaining, but with the required rule for negation. Furthermore, for classical logic, skolemization can reduce the amount of conversion, and relative negative complexity of the problem. As in the case of Near-Horn Prolog, the technique is best suited for near Harrop problems, i.e. where the majority of clauses is in the allowed syntax and only a limited number of disjunctions is present. Performances do degrade gracefully in proportion to the amount of clausification. Thus for something like the pigeon-hole problem this method would not be the first choice.

After this introduction, the rest of the paper is organized as follows. Section 2 is devoted to preliminaries, while in section 3 we demonstrate the method. In section 4 we briefly review related systems. Finally we conclude with some preliminary remarks on implementation issues and limitations.

2. Preliminaries on Uniform Proofs and Negation

Minimal logic (M) can be formalized in natural deduction by assuming a language with no negation connective and a primitive constant that denotes a not provable formula, say $\perp$. Then, given a first-order formula B, $\neg$B is defined as B$\supset\perp$: the usual rules for the other logic operators are retained, while no other property of the negation sign is assumed other than those inherited as specialization from modus ponens and implication introduction.

$$\dfrac{\dfrac{\begin{array}{c}[B]\\ \Pi\\ \bot\end{array}}{B \supset \bot}}{\neg B} \qquad\qquad \dfrac{\neg B \qquad B \supset \bot \quad B}{\bot}$$

To obtain intuitionism from minimal logic one adds the so called Duns Scotus law (DSL):

$$\dfrac{\bot}{B}$$

It is well known that classical logic can be obtained adding RAA:

$$\dfrac{\begin{array}{c}[\neg B]\\ \Pi\\ \bot\end{array}}{B}$$

Due to monotonicity, now DSL is a derived rule. What is remarkable is that in general the adjoining of RAA, whenever restricted to atoms, to a constructive[1] system preserves this property; this fact is an immediate consequence of the normalization theorem.

Uniform proofs can be roughly characterized in natural deduction as derivations whose branches are all in expanded normal form [Pra65]. This allows us to see a proof as a search procedure that works its way out in a bottom-up fashion from compound to atomic formulas successively eliminating the main connective. The idealized interpreter for compound goals is described by the following search instructions: any triple ‹ D,G,|- ›, whose provability relation satisfies those principles, is called an *abstract logic programming language* [Mil91].

AND:	P|- G1&G2	only if P|-G1 and P|-G2
OR:	P|- G1 v G2	only if P|-G1 or P|-G2
INSTANCE:	P|- ∃xG	only if there is a term t s.t. P|-[t/x]G
AUGMENT:	P|- D⊃G	only if D,P|-G
GENERIC:	P|-∀xG	only if P|-[a/x]G for a new parameter 'a'

[1] we identify constructive systems with those logics satisfying OR and INSTANCE (see below). This is probably limiting. See [Tro88].

Note that this only partially specifies the behavior of the interpreter: uniform proofs are not required to describe a specified course of action for atoms: every constructive move is generally welcomed. Thus there is no reason to built some particular choice in the system, although in overwhelmingly many cases this is best described in terms of backchaining: atoms either succeed because they are instances of clauses in the program or when an implicational clause, i. e., a "rule" with matching head is come across; then the procedure is reentered. For complete theorem proving, on the other hand, we have to supplement the traditional method to prove atoms with negative reasoning.

Following [Mom92a], we adopt the following language:

$$D ::= A \mid \bot \mid G \supset D \mid \forall x D \mid D1 \,\&\, D2 \mid D1 \leftrightarrow D2 \mid \neg G$$

$$G ::= A \mid \bot \mid G1 \,\&\, G2 \mid G1 \vee G2 \mid \forall x G \mid \exists x G \mid D \supset G \mid D1 \leftrightarrow D2 \mid \neg D$$

Negation, being camouflaged implication, is executed through the AUGMENT/backchain operation (see below); the evaluation of $\neg D$ consists in the assumption of D and in the attempt to prove falsehood from the enlarged program. Therefore the complement of a negated goal has to be a definite clause since it will dynamically join the program: analogously negative definite clauses are negation of goals since in backchaining mode are bound to become new goals. What follows is a suitable formulation of the uniform proof system, where we explicit the rules for minimal negation and enclose RAA.

Formally, atoms are dealt with through the notion of *immediate implication* that avoids the necessity of referring to an infinitary notion of program as resulting from its closure under substitution with all the terms of the language. Alternatively, the immediate implication rules correspond to the left rules in the sequent calculus for conjunction, implication and universal quantification, restricted to an atomic succedent.

$$\frac{P \vdash G_1 \quad P \vdash G_2}{P \vdash G_1 \,\&\, G_2} \qquad \frac{P \vdash G_i, i = 1,2}{P \vdash G_1 \vee G_2} \qquad \frac{P \vdash [a / x]G \text{ 'a' eigen variable}}{P \vdash \forall x G}$$

$$\frac{P \vdash [t / x]G}{P \vdash \exists x.G} \qquad \frac{D, P \vdash G}{P \vdash D \supset G} \qquad \frac{D, P \vdash \bot}{P \vdash \neg D}$$

$$\frac{P|\!-D_1 \supset D_2 \; P|\!-D_2 \supset D_1}{P|\!-D_1 \leftrightarrow D_2} \qquad \frac{\neg A, P|\!-\bot}{P|\!-A}$$

$$\frac{}{P|\!-A >> A} \qquad \frac{P|\!-D >> A, \quad D \in P}{P|\!-A} \qquad \frac{P|\!-D >> \bot, \quad D \in P}{P|\!-\bot}$$

$$\frac{P|\!-[t/x]D >> A}{P|\!-\forall x D >> A} \qquad \frac{P|\!-D_i >> A}{P|\!-D_1 \& D_2 >> A} \qquad \frac{P|\!-G}{P|\!-\!-G >> \bot} \qquad \frac{P|\!-G \quad P|\!-D >> A}{P|\!-(G \supset D) >> A}$$

For a very simple illustration of the method, which nevertheless requires both minimal and classical negations, we prove Pellettier's no. 4 (for the sake of brevity, we omit the immediate implication steps).

$p, \neg p, \neg q, (\neg p \supset q) \vdash p$
$p, \neg p, \neg q, (\neg p \supset q) \vdash \bot$
$\neg p, \neg q, (\neg p \supset q) \vdash \neg p$
$\neg p, \neg q, (\neg p \supset q) \vdash q$
$\neg p, \neg q, (\neg p \supset q) \vdash \bot$
$\neg q, (\neg p \supset q) \vdash p$
$(\neg p \supset q) \vdash (\neg q \supset p)$
$\vdash (\neg p \supset q) \supset (\neg q \supset p)$

Of course in real life, we would have stopped the computation before the last two steps, declaring success when any kind of goal, not only atoms, is also found on the left of the turnstile.

From now on we denote this provability relation without RAA with '|-', otherwise with '$|\!-_{RAA}$'. It is easy to show a proof-theoretic Soundness and Completeness wrt natural deduction for the minimal calculus. Completeness instead relies on the normalization theorem for M. Everything is easily extendable to uniform proofs with DSL w.r.t. intuitionistic logic, provided we may reduce every application of DSL to an atomic inference.

For proofs of results stated here, please refer to [Mom92b].

Theorem 2.1 (Soundness and Completeness). P⊢-G iff there is a natural derivation Π in minimal logic of G from assumptions P, denoted Π:: Γ⊢-$_M$G.

Clearly '⊢-$_{RAA}$' is sound but not complete; for example no uniform proof of A v ¬A exists. We remedy this difficulty in the next section.

3. Extending Uniform Proofs to Full (Classical) Logic

Thus, we have slightly extended Miller's original system, which was limited to fohh (hohh) without negation and intuitionistic provability. Yet what we have is a proof system for a weird logic, which is essentially the restriction to the allowed syntax of the intermediate constructive logic obtained adding atomic RAA to intuitionism [Mig89]. We are still fairly distant from a complete method. Yet, due to the availability of negation we are able to adopt the Gödel-Gentzen negative translation [Tro88] of classical logic into intuitionism to show an embedding of minimal, intuitionistic and classical logic into fohh. Note that we cannot simply replace disjunctions and existentials because the provability relation underlying uniform proofs is not classical.

We define a mapping *: Form --> Form as follows

$\bot^* = \bot$
$A^* = T_L(A)$
$(M\&N)^* = M^*\&N^*$
$(M \vee N)^* = \neg(\neg M^* \& \neg N^*)$
$(M \supset N)^* = M^* \supset N^*$
$(M \leftrightarrow N)^* = M^* \leftrightarrow N^*$
$(\forall x M)^* = \forall x M^*$
$(\exists x M)^* = \neg\forall x \neg M^*)$

$T_C(A)=A$
$T_M(A)=T_I(A)=\neg\neg A$

In C atoms are preserved, while for M and I, we adopt the double negation translation. This is due to the unavailability of RAA. A variant (Gentzen's) would cancel implications:

$(M \supset N)^* = \neg(M^* \& \neg N^*)$. This may be useful in compiling away implications with 'D' head, as with the equivalencies reported in [Mil91], thus simplifying the immediate implication steps; in addition, it may ease the location of contradictory pairs (see the example below).

Several optimizations are possible: we may discard n+3 negations for n+1. This shows that double negation can be restricted to positive goals. Indeed, if we are dealing with theories with decidable atoms, it is possible to set $A^*=A$, also for M and I. Moreover, as far as C is concerned, skolemization may be used to remove existentials without increasing the 'negative' complexity of the formula.

Now, before giving an example, we outline the (proof-theoretic) completeness proof.

Fact. For all first-order formula M, M* is a fohh clause.

Lemma 3.1 If $\Pi :: \Gamma \vdash_L N$, then $\Pi^* :: \Gamma^* \vdash_M N^*$.

Theorem. 3.2 $\Pi :: \Gamma \vdash_{M,I} N$ implies $\Gamma^* \vdash N^*$. $\Pi :: \Gamma \vdash_C N$ implies $\Gamma^* \vdash_{RAA} N^*$.

Example: Let us sketch the derivation of Pelletier's no.30. The first step is the conversion to fohh (in the Gentzen format); thereafter the fixed premises are left implicit in the rest of the derivation.

$$\ldots$$
$$\neg Fa, \neg Ga\ \neg Ia, \ldots \vdash \neg Ia$$
$$\neg Fa, \neg Ga\ \neg Ia, \ldots \vdash Ga \supset \neg Ia$$
$$\neg Fa, \neg Ga\ \neg Ia, \ldots \vdash Fa$$
$$\neg Fa, \neg Ga\ \neg Ia, \ldots \vdash \bot$$
$$\neg Ia, \ldots \vdash \neg(\neg Fa \& \neg Ga)$$

$$\ldots$$
$$Ga, \neg Ia, \ldots \vdash \neg Ia$$
$$\neg Ia, \ldots \vdash Ga \supset \neg Ia$$
$$\neg Ia, \ldots \vdash Ha$$

$$\neg Ia, \ldots \vdash \neg(\neg Fa \& \neg Ga) \& Ha$$
$$\neg Ia, \ldots \vdash \bot$$
$$\forall x \neg(\neg(\neg Fx \& \neg Gx) \& Hx), \forall x((Gx \supset \neg Ix) \supset. Fx, \forall x((Gx \supset \neg Ix) \supset. Hx) \vdash Ia$$
$$\forall x \neg(\neg(\neg Fx \& \neg Gx) \& Hx), \forall x((Gx \supset \neg Ix) \supset. Fx, \forall x((Gx \supset \neg Ix) \supset. Hx) \vdash \forall x Ix$$
$$\forall x(Fx \vee Gx \supset. \neg Hx), \forall x((Gx \supset \neg Ix) \supset. Fx \& Hx) \vdash \forall x Ix$$

Thus we have a compilation technique for the aforementioned logics into a goal-oriented system, once we agree either to extend uniform proofs with RAA or to double-negate

atoms. Though clearly the proof system is not likely to have a feasible lower bound on the length of derivations, since it can polynomially simulate classical logic, this approach has some appeal; given this balance between the 'naturality' of the proof system and the minimal quantity of conversion to the required syntax, we manage to preserve most of the information given by the logical structure of the goal and axioms.

4. Related Work

Our approach to theorem proving can either be seen as an extension of logic programming or a refinement of sequent calculus. As the latter is concerned we are prescribing a goal-oriented bottom-up linear evaluation strategy enforced by the uniformity condition. By means of the negative translation, we are able to avoid the case analysis required by the v-L rule and its potentially high backtracking cost, as in the classical example p v q |- q v p. More in general, we avoid all the problems caused by the non-permutability of rules [Kle52], by restricting to a permutably problems' free fragment of the calculus.

For the former perspective, our method has apparent analogies with the recent attempts to import in the field of ATP techniques from logic programming. The seminal and more successful system is Stickel's PTTP [Sti88], which is intended as a brute-force, very fast in LIPS theorem-prover. It supplements SLD-resolution with the model elimination rule. This entails keeping track of the ancestors of the goal, loosing one of the key feature of Prolog. Moreover, due to the directionality of SLD-resolution it requires the inclusion of the contrappositives of the clauses, causing a notable increase in the size of the program. Loveland's nH-Prolog [Lov88] can be seen as a way of incorporating case analysis in SLD-resolution; the evaluation of non Horn problems demands each time the invocation of a restart rule, until the stack of unsolved (deferred) heads is empty. Without requiring contrappositives it simulates case analysis with different runs of essentially the Prolog engine. Unfortunately naive nH Prolog is incomplete and the new versions (Progressive nH and Inheritance Nh) have a less intuitive and efficient description. Plaisted [Pla88] instead is concerned with a difficult attempt to use versions of the cut rule backwards leading to a problematic guessing of its premises. For a comparison, see [Ree92].

Though developed independently, our method turned out to have a very tight relationship with N-Prolog [Gab84], which is a complete implementation of positive intuitionistic logic.

By defining disjunction classically and allowing a restart rule, Gabbay shows it to be complete for full classical logic as well.

Another close relative goes under the name of *disjunctive logic programming* (see [Min91] and references therein). It aims to deal with full clausal logic by extending the tools for Horn Logic. In particular the procedural semantic, called SLO-resolution, extends the SLD-step by having a selection function choose a clause in the current goal and trying to find a clause whose head subsumes it; if successful, the goal is incremented by the body, without deleting the selected clause. This approach, that seems fairly expensive, since the subsumption test is already NP-complete, has not been advocated, as far as I know, for theorem proving.

It has been recently observed, among others, by Murthy [Mur93] that the double-negation/A embedding can be seen as a CPS-translation of the correspondent proof terms. Although he rules out the Gödel-Gentzen interpretation as too complicated, a simple proof relying on the admissibility of the Markov principle shows that this it actually induces a CPS-translation. In this context the CPS-translation is in charge of the transformation of any normalized proof Π::F, for any F to a uniform proof Π^*::F*. Making uniform is CPS-translating. If this is true, we may then use the CPS-translation backwards to recover the original natural deduction format of the proof. But this has more important consequences from a type-theoretic perspective: we may use uniform proofs to derive mechanically specifications and have programs (proofs) already in a machine-oriented form as the CPS style. We have to investigate this together with the relationship with the so-called "direct method" [Sch92]; here programs are extracted from classical proofs (in atomically decidable theories), without any roundabout, from the ($\forall, \supset$-fragment) of minimal logic, with the other connectives being defined as usual.

5. Implementation Issues and Conclusions

While the procedure is clearly absolutely deterministic (at least as Prolog is considered to be), the main problematic point is the detection of contradictory pairs to fire the negation rule. Clearly the notion of goal-oriented-ness tends to break down as we move away from the Harrop format: every negative sentence is a candidate for backchaining and the indexing of predicates that proved to be so effective in Prolog is not achievable. Thus, it seems that we are lead back to the original problem of resolution. On the other hand let me remark that

this situation is much more local to the proof. Moreover, since uniform proofs are per se normalized, the subformula property holds [Pra65]. Contradictory pairs have only to be searched among the subformulae of the current program and goal. Although this is not a conclusive answer, heuristics can be developed to locate the more promising ones; some (extraction strategies) have already been tested in the context on the intercalation calculus [Sie92], others (weighting) can be inherited from resolution theorem provers.

A possible limitation of the method could be detected in the need to reach the atomic level in every proof. Arguably, though sensible from a programming point of view, where atoms are procedures, sometimes we would rather stay at higher level: for instance, if dealing with elementary part of set theory, we may wish not to unfold the definition, say, of 'subset' and hold it to its property of being a partial order. Nevertheless, nothing prevents us, on the responsibility of the user, to declare those definitions as atomic and forbid their expansion.

Another questionable feature may seem our neglect for intuitionistic proofs. In first, sometimes we may recover the latter from a classical one after execution, just by checking whether the assumptions of RAA have been used, as the following trivial example shows; here q is assumed, but not used and thus it can be canceled:

$$\neg q, p, \neg p \vdash p$$
$$\neg q\ p, \neg p \vdash \bot$$
$$p, \neg p \vdash q.$$
$$p \vdash \neg p \supset q.$$
$$\vdash p \supset . \neg p \supset q$$

Moreover the whole procedure seems to be neatly amenable of some parallel implementation in the spirit of or-parallelism. Looking for a proof of G from P, we may run a uniform proof for G* from P*; every time we hit an atom, for which there is a backchaining clause, we branch three different processes looking respectively for a minimal, intuitionistic or classical uniform proof. Of course, this can be done sequentially too, by making more complicated the backtracking procedure.

The proof system is very easily implementable on the top of every logic programming language that supports fohh (hohh), as λProlog or Elf [Pfe91]. Indeed, the current SML (naive) implementation is a simple modification of the interpreter for λProlog described in [Ell91]. We follow Stickel's steps modifying the interpreter by building in ∀-unification

with the extended occurs check for soundness and depth-first iterative-deepening search instead of unbounded depth-first to make the search strategy complete. The application of the method to higher-order logic is next.

Acknowledgments

I would like to thank Frank Pfenning, Wilfried Sieg and Mario Ornaghi.

6. References

[And81] Andrews, P. Theorem Proving via General Matings, *JACM*, Vol.28, pp. 193-214, 1981.

[Ble77] Bledsoe W. Non Resolution Theorem Proving. *Artificial Intelligence*,9, pp.1-36, 1977.

[Ell91] Elliot C. & Pfenning F. A Semi-Functional Implementation of a Higher-Order Logic Programming Language, available by anonymous ftp from alonzo.tip.cs.cmu.edu.

[Gab84] Gabbay D. M. & Reyle U. , N-Prolog: an Extension of Prolog with Hypothetical Implications.1 *Journal of Logic Programming*, 4, pp. 319-355, 1984.
[Kle52] Kleene S.C. Permutability of Inferences in Gentzen's Calculi LK and LJ. *Memoirs of the American Mathematical Society*, 10, pp. 1-26, 1952.

[Lov88] Loveland D. Near-Horn Prolog and Beyond CS-1988-25 Duke University

[Mig89] Miglioli P., Moscato U., Ornaghi M., Quazza S., Usberti G. Some Results on Intermediate Constructive Logics, *Notre Dame Journal of Formal Logic*, v.30 n.4, 1989.

[Mil89] Miller D. A Logical Analysis of Modules in Logic Programming, *Journal of Logic Programming*, pp. 79-108, 1989.

[Mil91] Miller D. , Nadathur G. , Pfenning F. , Scedrov A. Uniform Proofs as a Foundation for Logic Programming, *The Annals of Pure and Applied Logic*, v.51, pp. 125-157, 1991.

[Min 91] Minker J., Rajasekar A. & Lobo J. A Theory of Disjunctive Logic Programs, in *Computational Logic*, Lassez J.L. & Plotkin G. (eds.), MIT Press 1991.

[Mom92a] Momigliano A. Hereditary Harrop Formulae and Minimal Negation, to appear in *Logic Foundations of Computer Science*, Nerode A. (ed.) Springler-Verlag, 1992.

[Mom92b] Momigliano A. *Some Remarks on Uniform Proofs and Constructive Negation*, Master Thesis, Department of Philosophy, Carnegie Mellon University, 1992.

[Mur93] Murthy C. Classical Logic as a Programming Language: Typing Nonlocal Control. *Maniscript*, 1993.

[Pel86] Pellettier F. J. 75 Problems for Testing Automatic Theorems Provers, *JAR*, 2, pp. 191-216, 1986.

[Pfe91] Pfenning F. Logic Programming in the LF Logical Framework,*Logical Frameworks*, Huet & Plotkin (eds.) pp. 149-182, Cambridge University Press 1991.

[Pla88) Plaisetd D.A. Non Horn Logic Programming without Contrappositives. *JAR*, 4 pp. 287-325, 1988.

[Pra65] Prawitz D. *Natural Deduction,* Almqvist&Wiksell, Stockholm 1965.

[Ree92] Reed D. W. & Lovelannd W.D. A Comparison of three Prolog Extensions. *JLP*, v. 12, pp. 25-50, 1992.

[Rob65] Robinson J. A. A Machine-Oriented Logic Based on the Resolution Principle.*JACM* 12 pp. 23-41.

[Sch92] Schwichtenberg H. Proofs as Programs, in *Proof Theory. A Selectioin of papers from the Leeds Proof Theory Programme 1990,* S. Wainer (Ed.), Cambridge University Press 1992.

[Sie92] Sieg W. *Mechanism and Search. Aspect of Proof Theory.* Report CMU-Phil 30, 1992.

[Sti89] Stickel M. A Prolog Technology Theorem Prover : a New Exposition and Implementation in Prolog. 464 *SRI International.*, Technical note, 46, 4. Menlo Park, Calif., 1989.

[Tro88] Troesltra A.S. & van Dalen D. *Constructivism in Mathematics,* vol 1-2. North-Holland, Amsterdam, 1988.

[Wos91] Wos L. & McCune W. Automated Theorem Proving and Logic Programming: A Natural Symbiosis. JLP, 1-53, 1991.