$\mho$-Resolution: An Inference Rule for Regular Multiple-Valued Logics*

Technical report 98-1
March, 1998

James J. Lu
jameslu@bucknell.edu
Department of Computer Science
Bucknell University
Lewisburg, PA 17837. U.S.A.

Neil V. Murray
nvm@cs.albany.edu
Department of Computer Science
State University of New York
Albany, NY 12222

Erik Rosenthal
brodsky@charger.newhaven.edu
Department of Mathematics
University of New Haven
West Haven, CT 06516

Abstract

The inference rule $\mho$-*resolution* for regular multiple-valued logics is developed. One important advantage of $\mho$-resolution is that linear, regular proofs are possible. That is, unlike existing deduction techniques, $\mho$-resolution admits input deductions while maintaining regular signs. More importantly, $\mho$-resolution proofs are at least as short as proofs for definite clauses gerated by the standard inference techniques — annotated resolution and reduction — and pruning of the search space occurs automatically.

Keywords: $\mho$-resolution, inference, signed and annotated logics

---

*This research was supported in part by the National Science Foundation under grants CCR-9225037, CCR-9404338 and CCR-9202013.

1 Introduction

Signed logics [18, 10] provide a general automated reasoning framework for reasoning about multiple-valued logics (MVL's). They evolved from, among others, the following work on non-standard computational logics [2, 3, 5, 6, 8, 14, 13, 16, 20, 23]. The key feature is attaching *signs* — subsets of the set of truth values — to formulas in the MVL. Practically, this approach is appealing because it facilitates the utilization of classical techniques for the analysis of non-standard logics. Semantically, the ability to analyze non-standard logics through classical logic is attractive because it reflects the essentially classical nature of human reasoning. That is, regardless of the domain of truth values associated with a logic, at the meta-level, humans interpret statements about the logic to be either $true$ or $false$.

This paper focuses on the class of *regular* signed logics. Regular signed logics are of interest in the knowledge representation and logic programming communities due to their correspondence to the class of paraconsistent logics known as *annotated logics*, introduced by Subrahmanian [22], Blair and Subrahmanian [2], and Kifer et al. [11, 12, 24]. In [18], regular signed logics were also shown to capture fuzzy logics, but in this paper, *regular signed logics* will refer to annotated logics. In particular, the focus is on the definite Horn subset of annotated logic, widely applied within logic programming. The inference rule $\mho$-*resolution* is developed for regular signed logics, and its relative advantages with respect to the standard inference rules for annotated logic programs (ALP's) — annotated resolution and reduction — are described. These include the fact that linear, regular proofs are possible; furthermore, $\mho$-resolution proofs are at least as short as annotated resolution proofs, and pruning of the search space occurs automatically. We begin with a summary of the basic ideas of signed formulas; greater detail can be found in [20].

2 Signed Logics

We assume a language Λ consisting of (finite) logical formulas built in the usual way from a set $\mathcal{A}$ of atoms (predicates and terms at the first order level), a set of connectives, and a set of logical constants. For the sake of completeness, we define a formula in Λ as follows: Atoms are formulas; if Θ is an n-ary connective and if $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n$ are formulas, then so is $\Theta(\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n)$.

Associated with Λ is a set Δ of truth values, and an interpretation for Λ is a function from $\mathcal{A}$ to Δ; i.e., an assignment of truth values to every atom in Λ. A connective Θ of arity n denotes a function $\Theta : \Delta^n \rightarrow \Delta$. Interpretations are extended in the usual way to mappings from the set of formulas to Δ. Alternatively, a formula $\mathcal{F}$ of Λ can be regarded as denoting a mapping from interpretations to Δ.

A *sign* is a subset of Δ, and a *signed formula* is an expression of the form $S\!:\!\mathcal{F}$, where S is a sign and $\mathcal{F}$ is a formula in Λ. If $\mathcal{F}$ is an atom in Λ, we call $S\!:\!\mathcal{F}$ a *signed literal.*

Signed formulas may be thought of as a formalization of meta-reasoning over MVL's [20]. A natural interpretation of the signed formula $S\!:\!\mathcal{F}$ is the query, "Is the truth value of $\mathcal{F}$ in S?" The answer to such a query is yes or no; that is, either the formula evaluates to some element in S or does not. Observe that both the query and the answer are at the meta-level; observe also that the question cannot even be formulated at the object level. On the other hand, the question, "What is the the truth value of $\mathcal{F}$?" may be interpreted at the object level since the answer is an element of Δ.

For example, let Δ be the interval [0, 1], where elements of Δ represent the degree of belief of some fixed reasoning agent X. Thus, $\{1\}:P$ can be interpreted as, "Is X certain of the proposition P?" and $[0,.1]:P$ asks, "Is X quite doubtful of P?" These are yes or no questions.

To answer arbitrary queries, we map formulas in Λ to formulas in a classical logic Λ_S, *the language of signed formulas*; it is defined as follows: The literals are signed formulas and the connectives are (classical) conjunction and disjunction. It should be emphasized that a signed formula $S:\mathcal{F}$ is a literal in Λ_S regardless of the size or complexity of $\mathcal{F}$ and thus has no component parts in the language Λ_S. The set of truth values is $\{true, false\}$.

An arbitrary interpretation for Λ_S may make an assignment of *true* or *false* to any signed formula (i.e., to any literal) in the usual way. Our goal is to focus attention only on those interpretations that relate to the sign in a signed formula. To accomplish this we restrict attention to Λ-*consistent interpretations*. An interpretation I over Λ assigns to each literal, and therefore to each formula $\mathcal{F}$, a truth value in Δ, and the corresponding Λ-consistent interpretation I_c is defined by $I_c(S{:}\mathcal{F}) =$ *true* if $I(\mathcal{F}) \in S$; $I_c(S{:}\mathcal{F}) =$ *false* if $I(\mathcal{F}) \notin S$. Note that this correspondence between the set of all interpretations over Λ and the set of Λ-consistent interpretations over Λ_S is 1–1. Intuitively, Λ-consistent means an assignment of *true* to all signed formulas whose signs are simultaneously achievable via some interpretation over the original language. Restricting attention to Λ-consistent interpretations yields a new consequence relation: If $\mathcal{F}_1$ and $\mathcal{F}_2$ are formulas in Λ_S, we write $\mathcal{F}_1 \models_\Lambda \mathcal{F}_2$ if whenever I_c is a Λ-consistent interpretation and $I_c(\mathcal{F}_1) = true$, then $I_c(\mathcal{F}_2) = true$. Two formulas $\mathcal{F}_1$ and $\mathcal{F}_2$ in Λ_S are Λ-*equivalent* if $I_c(\mathcal{F}_1) = I_c(\mathcal{F}_2)$ for any Λ-consistent interpretation I_c; we write $\mathcal{F}_1 \equiv_\Lambda \mathcal{F}_2$. The following lemma is immediate.

Lemma 1 Let I_c be a Λ-consistent interpretation, let A be an atom and $\mathcal{F}$ a formula in Λ, and let S_1 and S_2 be signs. Then:
1. $I_c(\emptyset{:}\mathcal{F}) = false$;
2. $I_c(\Delta{:}\mathcal{F}) = true$;
3. $S_1 \subseteq S_2$ if and only if $S_1{:}\mathcal{F} \models_\Lambda S_2{:}\mathcal{F}$ for all formulas $\mathcal{F}$;
4. There is exactly one $\delta \in \Delta$ such that $I_c(\{\delta\}{:}A) = true$. □

The usual results involving Robinson's Unification Theorem are unaffected by this development, and techniques at the ground level can generally lift to the general level. As a result, attention is mostly restricted to the ground case for the remainder of the paper.

2.1 Signed Inference

In this section, we describe a method for adapting resolution to produce an inference rule for Λ_S. Similar notions have been developed by Baaz and Fermüller [1] and by Hähnle [9]. Many classical inference rules begin with links (complementary pairs of literals). Such rules typically deal only with formulas in which all negations are at the atomic level. Similarly, the inference techniques described below require that signs be at the "atomic level." To that end, we call a formula in Λ_S Λ-*atomic* if it has the property that whenever $S:A$ is a literal in the formula, then A is an atom in Λ; i.e., if the only atoms in the formula are signed literals. Often, one assumes that all formulas are Λ-atomic (annotated logic formulas, for example).

The inference rule defined below is based on the notion of complementary literals, which is generalized in the next lemma. The lemma is immediate from part 4 of Lemma 1.

Lemma 2 [(*The Reduction Lemma*)] Let $S_1{:}A$ and $S_2{:}A$ be Λ-atomic atoms in Λ_S; then $S_1:A \wedge S_2:A \equiv_\Lambda (S_1 \cap S_2):A$ and $S_1:A \vee S_2:A \equiv_\Lambda (S_1 \cup S_2):A$. □

Consider a Λ-atomic formula $\mathcal{F}$ in Λ_S in conjunctive normal form (CNF). Let $C_j, 1 \le j \le r$, be clauses in $\mathcal{F}$ that contain, respectively, Λ-atomic literals $\{S_j{:}A\}$. Thus we may write $C_j = K_j \vee \{S_j{:}A\}$. Then the resolvent R of the C_j's is defined to be the clause

$$\left(\bigvee_{j=1}^{r} K_j\right) \bigvee \left(\left(\bigcap_{j=1}^{r} S_j\right) : A\right).$$

The rightmost disjunct is called the *residue* of the resolution; observe that it is unsatisfiable if its sign is empty and satisfiable if it is not.

We must augment this definition with the following obvious simplification rules that stem from the Reduction Lemma. First, if the residue is unsatisfiable it may simply be deleted from R. Secondly, whenever R contains literals $S_i{:}B$, $1 \le i \le k$, we *merge* them into the literal $\bigcup_{i=1}^{k} S_i{:}B$; if $\bigcup_{i=1}^{k} S_i = \Delta$, then R is a tautology and may be deleted from $\mathcal{F}$. Such merging is essentially a generalization to MVL's of ordinary classical ground merging.

The classical notion of subsumption also generalizes to Λ_S: Clause C subsumes clause D if, for every literal $S{:}A \in C$, there is a literal $S'{:}A \in D$ such that $S \subseteq S'$. Observe that if $S \subseteq S'$, and if two clauses are resolved on the literals $S{:}A$ and $S'{:}A$, then the residue will be $S{:}A$ (after all, $S \cap S' = S$), so the clause containing $S{:}A$ must subsume the resolvent. This proves

Lemma 3 The resolvent produced by resolving on two literals in which the sign of one contains the sign of the other is superfluous in the sense that the resolvent is necessarily subsumed by one of its parents. □

2.2 Regular Signed Formulas and Annotated Logics

Assume now that the set of truth values Δ is not simply an unordered set of objects but instead forms a complete lattice under some ordering $\preceq$. The greatest and least elements of Δ are denoted $\top$ and $\bot$, respectively, and Sup and Inf denote, respectively, the supremum (least upper bound) and infimum (greatest lower bound) of a subset of Δ.

Let $(P; \preceq)$ be any partially ordered set, and let $Q \subseteq P$. Then $\uparrow Q = \{y \in P | (\exists x \in Q)\ x \preceq y\}$. Note that $\uparrow Q$ is the smallest *upset* containing Q (see [4]). If Q is a singleton set $\{x\}$, then we simply write $\uparrow x$. We say that a subset Q of P is *regular* if for some $x \in P$, $Q = \uparrow x$ or $Q = (\uparrow x)'$ (the set complement of $\uparrow x$). We call x the *defining element* of the set. In the former case, we call Q *positive*, and in the latter *negative*. Observe that both Δ and $\emptyset$ are regular since $\Delta = \uparrow\bot$ and $\emptyset = \Delta'$. Observe also that if $z = \text{Sup}\{x, y\}$, then $\uparrow x \cap \uparrow y = \uparrow z$. A signed formula is regular if every sign that occurs in it is regular. By Part 1 of Lemma 1, we may assume that no regular signed formulas have any signs of the form $(\uparrow x)'$, where $x = \bot$, since in that case $(\uparrow x)' = \emptyset$.

The diagram in Figure 1 is a good way of thinking of regular signs. The larger diamond represents a lattice, and the point x is an element in the lattice. The inner diamond is the set $\uparrow x$, while the complement is the set $(\uparrow x)'$.

An annotated logic is a signed logic in which the only allowable signs are regular. Since a regular sign is completely characterized by its defining element, say x, and its *polarity* (either positive or negative), we may write a regular signed atom as $x : \mathcal{F}$ if its sign is positive, or as $\sim x : \mathcal{F}$ if its sign is negative. [1] Historically, annotated logics

[1] The annotated logic community typically writes $\mathcal{F} : x$ and $\sim\ \mathcal{F} : x$. We will stay with this convention when dealing with regular signed logics.

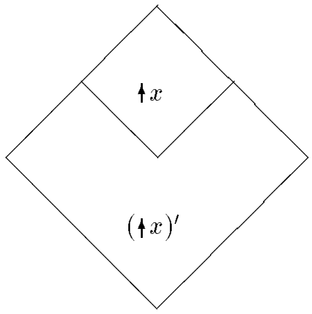

Figure 1: Regular Signs

have been restricted to Λ-atomic formulas. Though this restriction is unnecessary, we will follow it for the remainder of the paper to simplify the presentation.

2.3 Signed Resolution for Annotated Logics

A sound and complete resolution proof procedure was defined for clausal annotated logics in [14]. The procedure contains two inference rules that we will refer to as *annotated resolution* and *reduction*.[2] These two inference rules correspond to disjoint instances of signed resolution. Two annotated literals L_1 and L_2 are said to be *complementary* if they have the respective forms $A{:}\mu$ and $\sim(A{:}\rho)$, where $\mu \geq \rho$.

Definition. (Annotated Resolution) Given the annotated clauses $(L_1 \vee D_1)$ and $(L_2 \vee D_2)$, where L_1 and L_2 are complementary, then the *annotated resolvent* of the two clauses on the annotated literals L_1 and L_2 is $D_1 \vee D_2$.

Two clauses can be so resolved only if the annotation of the positive annotated literal that is resolved upon is greater than or equal to the annotation of the negative literal resolved upon. In that case the two clauses are said to be *resolvable* on the annotated literals L_1 and L_2.

Definition. (Reduction) Suppose $(A:\mu_1 \vee E_1)$ and $(A:\mu_2 \vee E_2)$ are two annotated clauses in which μ_1 and μ_2 are incomparable. Then the annotated clause $(A{:}\mathrm{Sup}\ \{\mu_1, \mu_2\}) \vee E_1 \vee E_2$ is called a *reductant* of the two clauses, and we say that the two clauses are *reducible* on the annotated literals $A{:}\mu_1$ and $A{:}\mu_2$.

It is straightforward to see that the two inference rules are both captured by signed resolution. In particular, annotated resolution corresponds to an application of signed resolution (to regular signed clauses) in which the signs of the selected literals are disjoint. Reduction on the other hand, correspond to an application of signed resolution in which the signs of the selected literals are both positive. The next theorem is now immediate.

Theorem 4 Suppose that $\mathcal{F}$ is a set of annotated clauses and that $\mathcal{D}$ is a deduction of $\mathcal{F}$ using annotated resolution and reduction. Then $\mathcal{D}$ is a signed deduction of $\mathcal{F}$. In particular, if $\mathcal{F}$ is an unsatisfiable set of first order annotated clauses, then there is a signed refutation of $\mathcal{F}$. $\Box$

Hence, signed resolution provides a general method for implementing annotated logics, but, as is often the case in theorem proving, there is a trade-off between generality and efficiency. Since annotated resolution and reduction are instances of signed

[2]Kifer and Lozinskii refer to their first inference rule simply as resolution. However, since we are working with several resolution rules in this paper, appropriate adjectives will be used to avoid ambiguity.

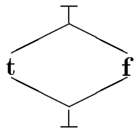

Figure 2: The Complete Lattice FOUR.

resolution, and since there exist signed resolutions for which no corresponding annotation resolutions or reductions exist, the search space induced by signed resolution will typically be larger than the search space induced by annotated resolution and reduction. The following theorem, proved in [18], characterizes that class of signed deductions which corresponds to the deductions obtainable via annotated resolution and reduction.

Theorem 5 Suppose $S_1, \ldots, S_n$ are regular signs whose intersection is empty, and suppose that no proper subset of $\{S_1, \ldots, S_n\}$ has an empty intersection. Then exactly one sign is negative; i.e., for some j, $1 \leq j \leq n$, $S_j = (\uparrow x_j)'$, and for $i \neq j$, $S_i = \uparrow x_i$, where $x_1, \ldots, x_n \in \Delta$. □

The intersection of a positive regular and a negative regular sign is regular if and only if it is empty, and two negative signs can have a regular intersection if and only if one is a subset of the other. In view of Lemma 3, the latter situation need never be considered, so we define a signed deduction to be *regular* if every sign that appears in the deduction is regular and if no residue sign is produced by the intersection of two negative signs. The next two theorems are immediate. Theorem 7 states that the class of regular signed deductions is precisely the class of deductions using annotated resolution and reduction. As a result, restricting signed resolution to regular clauses captures annotated resolution and reduction without increasing the search space.

Theorem 6 A signed deduction of a regular formula is regular if and only if the sign of every consistent residue is produced by the intersection of two positive regular signs. □

Theorem 7 Let $\mathcal{D}$ be a sequence of annotated clauses. Then $\mathcal{D}$ is an annotated deduction if and only if $\mathcal{D}$ is a regular signed deduction. □

Corollary. (Completeness of Regular Signed Resolution) Suppose $\mathcal{F}$ is an unsatisfiable set of regular signed clauses. Then there is a regular signed deduction of the empty clause from $\mathcal{F}$. □

2.4 Linear Signed Resolution and Horn Sets

Logic programming[3] may be thought of as providing a procedural interpretation of Horn clauses through linear resolution. Similarly, annotated Horn clauses may be interpreted procedurally through linear signed resolution. In light of Theorem 6 and the requirement that each step in a linear resolution involves the most recent resolvent, linear signed resolution may not in general be regular. Annotated logics, on the other hand, require regularity, which is incompatible with the linear restriction. This creates a difficulty if one desires to employ annotated logics in a logic programming setting. This is illustrated by the next simple example.

[3] Several researchers have explored annotated logics with an interest in logic programming. A more detailed account can be found in [19].

Consider the annotated logic program (ALP) $\mathcal{P}$, [4] written over the lattice FOUR pictured in Figure 2 (see [2]):

p:$\mathbf{t} \leftarrow$

p:$\mathbf{f} \leftarrow$.

Consider now the query, $\leftarrow p:\top$. It is easy to see $\mathcal{P} \models p:\top$. However, linear annotated resolution alone does not admit a refutation since the reductant p:$\top \leftarrow$ must be computed from the two program clauses. This clause would resolve with the query to yield a refutation, but this clause cannot be inferred with the linear restriction because the goal clause must be used in every step.

Kifer and Subrahmanian circumvent this difficulty by specifying that a deduction consist only of applications of annotated resolution. However, any inference may involve a resolution with an annotated clause obtained by implicit applications of reduction. In the last example, a proof could be obtained by a single deduction between the original query $\leftarrow p$:$\top$ and the reductant p:$\top \leftarrow$.

Unfortunately, implicit use of reduction creates new problems. For example, a proof that consists of only annotated resolution steps may contain steps that include clauses not in the original program. This makes the proof difficult to read. Moreover, application of the reduction rule can be expensive since it can occur at any time during a deduction, significantly expanding the search space.

Signed resolution, on the other hand, consists of a single rule of inference, and is therefore amenable to a linear restriction, thus making an SLD-like procedure implementable. Naturally, a regular proof will not in general be produced. Equivalently, any resolvent so produced cannot be annotated with a simple annotation. Hence, to directly apply such a procedure, an extension of the syntax of annotated atoms to accommodate more general annotations would be required.

2.5 A linear procedure with annotations

There is a means of implementing a linear strategy for signed resolution using annotations since, in view of Theorem 5, it is never necessary to resolve on literals with two negative signs. A trick is to associate with each active goal two annotations: a negative one α and a positive one β. The negative one is always the goal's initial annotation. The positive one, which changes during the deduction, is the supremum of the annotations of the positive literals against which the goal has been resolved. The two annotations α and β represent the sign $(\uparrow\alpha)' \cap \uparrow\beta$. More precisely, if $\leftarrow G:\alpha$ is the goal, then the initial positive annotation is $\perp$. At each stage, if the positive annotation is β, and if the goal is resolved against a rule whose head is $B:\rho$, there are two cases to consider. First, if $\alpha \leq \mathrm{Sup}\{\beta,\rho\}$, then the goal may be deleted. Otherwise, we replace the positive annotation β with $\mathrm{Sup}\{\beta,\rho\}$, and the goal remains active. This procedure is sound by the Reduction Lemma and is complete (See [18]).

Although this technique does provide linearity, reductions are nevertheless implicitly performed and represented via the second goal annotation. Of course, any method up until now that is complete for ALP's must either allow irregular signs for linearity or must capture reduction in some sense. Unfortunately, many redundant reduction steps are possible. As we shall see, $\mho$-resolution can avoid many such redundancies, and provide a cleaner input-style deduction in which no extra annotations are necessary; i.e., only strictly regular input deductions are required for ALP's.

[4] We use the left arrow to represent definite clauses in the standard way.

3 Decomposition and $\mho$-Resolution

To summarize the issues related to deduction so far, we have shown that

- in the case of regular signed logics (equivalently annotated logics) signed resolution provides generality and annotated resolution with reduction provides efficiency.
- in the case of definite Horn regular signed logics (i.e. annotated logic programs), signed resolution is amenable to the linear (and thus input) restriction while annotated resolution and reduction is not. Signed resolution does not, however, guarantee regularity and thus certain non-regular steps must be allowed.

The problem is to somehow simultaneously reconcile the shortcomings of these two approaches. When the goal annotation μ is not less than or equal to the program clause annotation ρ, then the intersection of $(\uparrow \mu)'$ and $\uparrow \rho$ is simply not regular and cannot be represented by an annotation. Signed resolution would produce the goal $((\uparrow \mu)' \cap \uparrow \rho) : p$. However, if we are able to soundly deduce this goal, then we may also soundly deduce any goal of the form $S : p$, where $S \supseteq ((\uparrow \mu)' \cap \uparrow \rho)$. Intuitively, $\mho$-resolution works by determining an annotation μ_0 such that $(\uparrow \mu_0)'$ is the smallest regular set that contains $((\uparrow \mu)' \cap \uparrow \rho)$. Thus we may soundly infer the new goal $p : \mu_0$ without losing regularity or linearity. Of course, fortuitously, completeness is not lost either.

What follows in the paper is the development of $\mho$-resolution. We first introduce a more general rule — *decomposition* — for inferencing over annotated logics. Decomposition by itself does not really address the main deduction issues successfully, but it leads naturally to $\mho$-resolution which is the rule of real interest. Consider the following motivating example.

Example 8 Let Δ be the lattice FOUR of Figure 2 and let Q be the query $\leftarrow p : \top$. Further, let P be the program

P_1 $\quad p : V \leftarrow q : V$
P_2 $\quad p : \mathbf{t} \leftarrow$
P_3 $\quad q : \mathbf{f} \leftarrow$.

Note that P_1 is a clause schema in which the variable V stands for all possible instantiations of elements in Δ. Such a shorthand has been widely used and provides only a convenience.

Clearly, $P \models p : \top$. Thus the query $\leftarrow p : \top$ should be provable. Using the technique of carrying two annotations, we may arrive at a proof using signed resolution in several ways. First, resolve the query against $p : \top \leftarrow q : \top$. The resolution step yields the new query $\leftarrow q : \top$. Next, resolve against P_2 to obtain $\leftarrow q : \top^{\mathbf{t}}$ where the superscript $\mathbf{t}$ indicates the positive annotation associated with q. Finally, resolving against P_3 produces $\leftarrow q : \top^{\top}$, or the empty clause.

As an alternative, suppose the one schema instance $p : \top \leftarrow q : \top$ is not in the program. The initial query is provable still by first resolving the query against $p : \mathbf{t} \leftarrow q : \mathbf{t}$ to produce the query $\leftarrow p : \top^{\mathbf{t}}, q : \mathbf{t}$. Resolving next against $p : \mathbf{f} \leftarrow q : \mathbf{f}$ yields $\leftarrow p : \top^{\top}, q : \mathbf{t}, q : \mathbf{f}$ which simplies to $\leftarrow q : \mathbf{t}, q : \mathbf{f}$. This query may now be proved through two simple resolution steps against P_2 and P_3.

3.1 Decomposition

Now instead of maintaining two annotations, suppose after we have determined that the initial query cannot resolve (using standard annotated resolution) with any program clause, we decompose the initial query into the two goal query: $\leftarrow p : \mathbf{t}, p : \mathbf{f}$.

Then the two goals can be resolved through annotated resolution against the program clauses in a straightforward way. In other words, since one way to show $p:\top$ is to show both $p:\mathbf{t}$ and $p:\mathbf{f}$, and since each of the latter two can be proven more easily, we may set up a rule of inference which performs such a decomposition whenever a suitable clause cannot be found for annotated resolution.

Definition. (Decomposition) Let Q be the query $\leftarrow A_1 : \mu_1, ..., A_m : \mu_m$. Given $A_i : \mu_i$ from the query Q, suppose $A_i : \rho_1$ and $A_i : \rho_2$ are literals such that $\mu_i \preceq \text{Sup}\{\rho_1, \rho_2\}$. Then we say $A_i : \mu_i$ *decomposes* to

$$A_i : \rho_1, A_i : \rho_2$$

and the decomposition Q^d of Q with respect to A_i is

$$\leftarrow A_1 : \mu_1, ..., A_{i-1} : \mu_{i-1}, A_i : \rho_1, A_i : \rho_2, A_{i+1} : \mu_{i+1}, ..., A_m : \mu_m.$$

Theorem 9 (Soundness of Decomposition) Suppose Q is a ground annotated query. Let Q^d be a decomposition of Q and I_c be a Λ_S-interpretation. If $I_c(Q) = true$, then $I_c(Q^d) = true$.

Proof. Let Q be the query $\leftarrow A : \mu$ where $A : \mu$ can be decomposed to $A : \rho_1, A : \rho_2$. Then the decomposed query Q^d is the query $\leftarrow A : \rho_1, A : \rho_2$. As $I_c(Q) = true$, $I_c(A : \mu) = false$. Hence $\mu \not\preceq I(A)$ where I is the Λ-interpretation that corresponds to I_c. It suffices to show that $I_c(Q^d) = true$, i.e. $\rho_1 \not\preceq I(A)$ or $\rho_2 \not\preceq I(A)$.

Because $\mu \not\preceq I(A)$ and $\mu \preceq \text{Sup}\{\rho_1, \rho_2\}$, we have $\text{Sup}\{\rho_1, \rho_2\} \not\preceq I(A)$. To show $\rho_1 \not\preceq I(A)$ or $\rho_2 \not\preceq I(A)$, assume the contrary condition $\rho_1 \preceq I(A)$ and $\rho_2 \preceq I(A)$. Then I(A) is an upper-bound of ρ_1 and ρ_2 by definition, which implies $\text{Sup}\{\rho_1, \rho_2\} \preceq I(A)$, and we obtain the necessary contradiction. □

Decomposition, together with annotated resolution, constitutes an SLD-style proof procedure and it does so using only regular signed literals. By decomposing an annotation into two "lower annotations", decomposition is a type of *divide-and-conquer* strategy. In the case of the previous example, it is straightforward to choose two annotations to which the query literal annotation $\top$ can be decomposed. However, programs over other types of lattices may require several guesses, as shown in the next example.

Example 10 Let Δ be the lattice of Figure 3 and let P be the program

P_1 $p:V \leftarrow q:V$
P_2 $q:1 \leftarrow$
P_3 $q:2 \leftarrow$.

Let Q be the query $\leftarrow p:\top$. Clearly $P \models p:\top$. If we chose to decompose $p:\top$ into the two literals $p:1$ and $p:3$, we observe that the corresponding query $\leftarrow p:1, p:3$ does not have a refutation using only annotated resolution. On the other hand, if we choose to decompose $p:\top$ into the two literals $p:1$ and $p:2$, we can obtain a refutation using annotated resolution.

The previous example demonstrates the difficulty of making a correct guess since the two annotations into which the original should be decomposed is dependent on the structure of Δ and on the program clauses. In the case of FOUR, the choice was immediate regardless of the annotations of the program clauses, whereas for the 1-2-3-lattice, the success of the choice depended on what information was contained within the program. Therefore, in general, decomposition may be expensive to apply

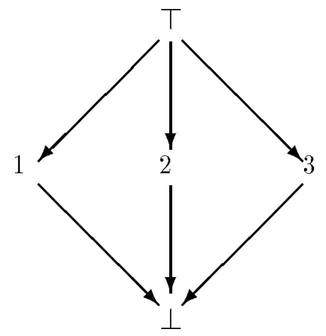

Figure 3: 1-2-3 Lattice

because the number of guesses may be large and most of them do not lead to a proof. However, there may be properties embedded in certain classes of lattices that can be used to reduce the number of guesses. In the next section, we define one class of lattices — *ordinary* lattices — and introduce a simple and elegant SLD-style proof procedure based on $\mho$-resolution for these lattices; $\mho$-Resolution is an enhancement to decomposition in which no guessing is performed.

3.2 Ordinary Lattices and $\mho$-Resolution

Decomposition will work for any lattice but as previously mentioned, its application may be expensive. One way to improve decomposition is to modify it to be a binary inference rule in which the selection of one component is based on the existence of an appropriate head literal with annotation ρ in a program clause. Based on μ and ρ, we may then guess for an annotation γ such that $\mu \preceq \mathrm{Sup}\{\gamma, \rho\}$. $\mho$-Resolution, described below, exploits this observation by further restricting the class of lattices to those with the property that for any two values μ, ρ of the lattice, there is a best choice for γ. We call such a collection of lattices ordinary lattices. First, given $\mu, \rho \in \Delta$, we use $\mathcal{M}(\mu, \rho)$ to denote the set of annotations for which the least upper bound of ρ with an element of the set is greater than or equal to μ. Formally, $\mathcal{M}(\mu, \rho) = \{\gamma \mid \gamma \in \Delta \text{ and } \mathrm{Sup}\{\gamma, \rho\} \succeq \mu\}$.

For the 1-2-3-lattice in Figure 3, if $\mu = \top$ and $\rho = 1$, $\mathcal{M}(\mu, \rho) = \{2, 3, \top\}$. For the lattice FOUR in Figure 2, if $\mu = \top$, and $\rho = \mathbf{t}$, $\mathcal{M}(\mu, \rho) = \{\mathbf{f}, \top\}$.

Definition ($\mho$ Operator) [5] Given $\mu, \rho \in \Delta$, we define $\mho(\mu, \rho)$to be $\mathrm{Inf}(\mathcal{M}(\mu, \rho))$.

Definition (Ordinary-Lattice) A lattice Δ is an *ordinary lattice* if $\forall \mu, \rho \in \delta$, $\mho(\mu, \rho) \in \mathcal{M}(\mu, \rho)$.

Notice that the 1-2-3-lattice is not an ordinary lattice because the set $\mathcal{M}(\mu, \rho) = \{2, 3, \top\}$ has no *least* element in the set. The lattice FOUR, however, is an ordinary lattice.

To simplify the notation, given $\mu, \rho_1, ..., \rho_m$, we write $\mho(\mu, \rho_1, ..., \rho_m)$ to represent $\mho(\mho(...\mho(\mho(\mu, \rho_1), \rho_2)...), \rho_m)$. The following lemma describes a property of ordinary

[5] The intended pronunciation is "mo", after the LaTeX symbol \mho.

lattices that is crucial for proving completeness of the $\mho$-resolution inference rule.

Lemma 11 Let Δ be an ordinary lattice. Then $\mho(\mu, \rho_1, ..., \rho_m) = \bot$ *iff* $\mu \preceq \rho$, where $\rho = \mathrm{Sup}\{\rho_1, ..., \rho_m\}$.

Proof. We begin by showing that if $\mu \preceq \rho$, where $\rho = \mathrm{Sup}\{\rho_1, ..., \rho_m\}$ then

$$\mho(\mu, \rho_1, ..., \rho_m) = \bot.$$

Suppose $\mu \preceq \rho$, then

$$\mu \preceq \mathrm{Sup}\{\rho_1, ..., \rho_m\} = \mathrm{Sup}\{\rho_1, \mathrm{Sup}\{\rho_2, ..., \rho_m\}\}.$$

By definition of $\mathcal{M}$,

$$\mu \preceq \mathrm{Sup}\{\rho_1, \mathrm{Sup}\{\rho_2, ..., \rho_m\}\} \textit{ iff } \mathrm{Sup}\{\rho_2, ..., \rho_m\} \in \mathcal{M}(\mu, \rho_1).$$

Thus $\mathrm{Sup}\{\rho_2, ..., \rho_m\} \succeq \mho(\mu, \rho_1)$. By a similar argument,

$$\mathrm{Sup}\{\rho_2, \mathrm{Sup}\{\rho_3 ..., \rho_m\}\} \succeq \mho(\mu, \rho_1) \textit{ iff } \mathrm{Sup}\{\rho_3 ..., \rho_m\} \in \mathcal{M}(\mathcal{M}(\mu, \rho_1), \rho_2),$$

so

$$\mathrm{Sup}\{\rho_3 ..., \rho_m\} \succeq \mho(\mu, \rho_1, \rho_2).$$

By extension, it follows that

$$\mathrm{Sup}\{\rho_m\} \succeq \mho(\mu, \rho_1, \rho_2, ..., \rho_{m-1}).$$

Therefore,

$$\mathrm{Sup}\{\} \succeq \mho(\mu, \rho_1, \rho_2, ..., \rho_{m-1}, \rho_m).$$

By definition, $\mathrm{Sup}\{\} = \bot$, and since there is no element less than $\bot$, the above inequality implies

$$\bot = \mho(\mu, \rho_1, \rho_2, ..., \rho_m).$$

Proof of the reverse condition

$$\mho(\mu, \rho_1, ..., \rho_m) = \bot \text{ if } \mu \preceq \rho, \text{ where } \rho = \mathrm{Sup}\{\rho_1, ..., \rho_m\}$$

is easily shown by reversing the steps of the above proof. □

In the event that Δ is an ordinary lattice, we can see that given a query literal $p : \mu$ and a program clause head annotation $p : \rho$, the natural choice for the decomposition of $p : \mu$ based on $p : \rho$ is $p : \rho$ and $p : \mho(\mu, \rho)$. The first part of the decomposition is obvious, and the second part represents the "simplest" remaining condition that needs to be solved in order to solve the original query. This leads us to the desired inference rule.

Definition. ($\mho$-Resolution) Given an ALP over an ordinary lattice, suppose Q is a query

$$\leftarrow A_1 : \mu_1, ..., A_m : \mu_m$$

and C is a program clause $A : \rho \leftarrow \mathit{Body}$, where A_i is unifiable with A via mgu θ and $\mho(\mu_i, \rho) \prec \mu_i$. The $\mho$-resolvent of Q and C with respect to A_i is the query

$$\leftarrow (A_1 : \mu_1, ..., A_{i-1} : \mu_{i-1},\ A : \mho(\mu_i, \rho), \mathit{Body},\ A_{i+1} : \mu_{i+1}, ..., A_m : \mu_m)\theta.$$

A $\mho$-*deduction* of a query from a given ALP and an initial query using $\mho$-resolution is defined in the usual way. A $\mho$-deduction of a query from an ALP is a $\mho$-*proof* if

$$\leftarrow A_1 : \bot, A_2 : \bot, ..., A_n : \bot$$

is the last clause in the $\mho$-deduction.

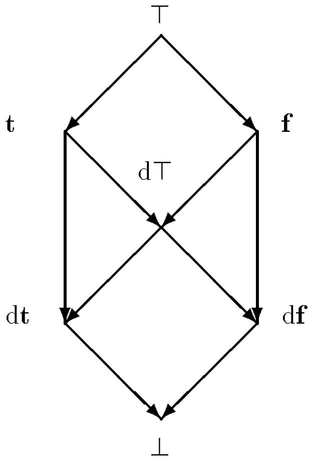

Figure 4: Lattice with Defaults

Lemma 12 (Soundness of $\mho$-Resolution) $\mho$-Resolution is sound for ordinary lattices.

Proof. Given an ALP over an ordinary lattice, suppose $Q \equiv \leftarrow A_1 : \mu_1, ..., A_m : \mu_m$ is a query, and $C \equiv A : \rho \leftarrow Body$ is a program clause, where A_i is unifiable with A via mgu θ. By the soundness of decomposition (Theorem 9), we can infer

$$Q^* \equiv \leftarrow A_1 : \mu_1, ..., A_{i-1} : \mu_{i-1},\ A : \mho(\mu_i, \rho), A : \rho,\ A_{i+1} : \mu_{i+1}, ..., A_m : \mu_m$$

since $\mu_i \preceq \mathrm{Sup}\{\rho, \mho(\mu_i, \rho)\}$. Applying annotated resolution to Q^* and C, we obtain

$$\leftarrow (A_1 : \mu_1, ..., A_{i-1} : \mu_{i-1},\ A : \mho(\mu_i, \rho), Body,\ A_{i+1} : \mu_{i+1}, ..., A_m : \mu_m)\theta.$$

which is precisely the $\mho$-resolvent of Q and C. □

The following example demonstrates the use of $\mho$-resolution over the ordinary lattice in Figure 4. Notice that the lattice consists of the values d**t**, d**f**, and d⊤, in addition to the four elements of FOUR. Ginsberg introduced this lattice as one useful for distinguishing between truth and default facts [7]. The symbols d**t** and d**f** may be regarded as "concluded **true** (resp. **false**) by default." Following this intuition, d⊤ stands for "**inconsistent** default conclusion" and is distinguished from the stronger inconsistency represented by ⊤.

Example 13 Let Δ be the lattice of Figure 4 and let P be the program

P_1 $\quad p : d\mathbf{t} \leftarrow$

P_2 $\quad p : d\mathbf{f} \leftarrow$

Let Q be the query $\leftarrow p : d\top$. Then a $\mho$-proof proceeds as follows.

$Q^* = Q_0 :$	$\leftarrow p : d\top$	Initial query
$Q_1 :$	$\leftarrow p : d\mathbf{f}$	$\mho$-resolvent of P_1 and Q_0: note $\mathcal{M}(d\top, d\mathbf{t}) = \{d\mathbf{f}, d\top\}$
$Q_2 :$	$\leftarrow p : \bot$	$\mho$-resolvent of P_2 and Q_1: note $\mathcal{M}(d\mathbf{f}, d\mathbf{f}) = \Delta$

We thus obtain a refutation. •

$\mho$-Resolution is a variant of the more general decomposition inference rule and it exploits properties intrinsic in the structure of the ordinary lattices where a unique guess for the decomposition exists. As such, $\mho$-resolution can be regarded as a semantically-based inference rule. Further, $\mho$-resolution allows a simple SLD-style proof procedure for annotated logic programs over ordinary lattices that eliminates the expensive reduction rule, yet does not require non-regular deduction.

Pruning of the search occurs naturally in $\mho$-resolution. For instance, consider the following program P over the lattice FOUR again.

Example 14

$$P = \{\underbrace{p:\mathbf{f} \leftarrow q:\mathbf{f}}_{C_1}, \underbrace{p:\mathbf{t} \leftarrow q:\mathbf{t}}_{C_2}, \underbrace{p:\mathbf{t}}_{C_3}, \underbrace{q:\mathbf{f}}_{C_4}\}.$$

Observe that $P \models p:\mathbf{t}$. However, the query $\leftarrow p:\mathbf{t}$ cannot $\mho$-resolve with the head of C_1, $p:\mathbf{f}$, since $\mho(\mathbf{t},\mathbf{f}) = \mathbf{t}$. Thus, C_2 and C_3 are acceptable candidates for $\mho$-resolution with the query, but C_1 is not. •

In essence, C_1 is not required for solving the query, and detection of such irrelevancies is built into $\mho$-resolution. In the same example, other inference techniques will not prevent resolution of the query with C_1, or the use of C_1 for reduction. This may represent considerable savings, since the body of C_1 could be arbitrarily large and/or could lead to an arbitrarily large but unnecessary deduction. Indeed, proof space reduction is a primary motivation for the introduction of $\mho$-resolution. In the next section, we show that since $\mho$-resolution exploits properties intrinsic in the structure of the ordinary lattices where unique guesses for performing decompositions exist, $\mho$-resolution guarantees not only a "smaller" proof space, but one in which only the "best" proofs are admitted. The basic result shows that for each proof of a query using annotated resolution and reduction, there is a $\mho$-proof of the same query using the same or fewer inferences. An example showing how search space pruning is built in to $\mho$-resolution is also given.

3.3 Proof/Search Space Considerations

In this section, we view deductions as sequences of inferences rather than as sequences of clauses. This is merely a matter of convenience. Standard annotated logic programming deductions consist of both annotated resolution and reduction steps, and we wish to easily refer to the particular inference rule applied at one step.

It is impractical (and unnecessary) to deal with the space of *all* annotated resolution/reduction deductions, mainly for two reasons. First, given any deduction, a different deduction can be obtained merely by adding redundant steps. Thus we will consider deductions in which every inference contributes; i.e., we assume that deductions are minimal in the sense that removal of any one step results in a sequence that does not formally comprise a deduction. Note that this in no way implies that such deductions are *shortest* in any sense.

Secondly, the steps of a given deduction may often be reordered in many ways. For example, the literals in the goal clause of an ALP computation may be solved in different orders. This amounts to reordering the annotated resolution steps of a deduction, and we consider such a reordering to yield a distinct deduction. However, a given annotated resolution may involve a reductant and thus be enabled by a series of reduction steps that produced that reductant. Obviously, these reductions (treated as implicit in annotated logic programming) can be done in any order and at any point prior to the resolution. Although we treat different reduction orders as different deductions, we do assume that, in aggregate, all reductions necessary to enable a resolution occur immediately prior to that resolution step. It is obvious that any deduction, say $\mathcal{D}'$, not allowed under these assumptions is in fact a redundant variation of one, say $\mathcal{D}$, that is allowed, and $\mathcal{D}'$ is no shorter than $\mathcal{D}$.

Therefore, we assume that an annotated logic programming deduction $\mathcal{D}$ is a sequence of annotated resolution steps: $\mathcal{S}_1, \mathcal{S}_2, \ \ldots, \ \mathcal{S}_n$. Furthermore, consider a given step $\mathcal{S} = \mathcal{S}_i$; $\mathcal{S}$ has the form $\mathcal{R}_1, \mathcal{R}_1, \ \ldots, \ \mathcal{R}_k, \mathcal{R}, \ k \geq 0$, where the $\mathcal{R}_i$'s are reductions and $\mathcal{R}$ is annotated resolution. Let $A:\mu$ be the goal removed by step $\mathcal{R}$, and let $A:\rho_0, A:\rho_1, \ \ldots, \ A:\rho_k$ be the heads of the $k+1$ program clauses used in $\mathcal{S}$. We will show that corresponding to these k reductions followed by a resolution, there is a sequence of $k{+}1$ $\mho$-resolutions, the last of which produces the goal $A:\perp$. (Starting with the goal clause, both deductions involve exactly the $k{+}1$ program clauses used for the k reductions. In both cases, the resulting goal list will be expanded to include all goals in the bodies of those program clauses. So the goal resolved away by annotated resolution is the only one of concern.)

The series of $k{+}1$ $\mho$-resolutions will, by definition yield the goal $A:\mho(\mu, \rho_0, ..., \rho_k)$. By the definition of annotated resolution, $\mu \preceq \mathrm{Sup}(\rho_0, \rho_1, \ \ldots, \ \rho_k)$. But then by Lemma 11, $\mho(\mu, \rho_0, ..., \rho_k) = \perp$, and we have proved:

Theorem 15 ($\mho$-Resolution Simulates Annotated Resolution) Let $\mathcal{P}$ be an annotated logic program, let $\mathcal{Q}$ be a query, and let $\mathcal{D}_{\mathcal{AR}}$ be a proof of $\mathcal{Q}$ from $\mathcal{P}$ by annotated resolution and reduction. Then there is a proof $\mathcal{D}_{\mho}$ of $\mathcal{Q}$ from $\mathcal{P}$ by $\mho$-resolution that is no longer than $\mathcal{D}_{\mathcal{AR}}$. □

Corollary. (Completeness of $\mho$-Resolution) Suppose $\mathcal{F}$ is an unsatisfiable set of regular signed Horn clauses. Then there is a $\mho$-resolution input deduction of the empty clause from $\mathcal{F}$. □

3.4 Characterizing Ordinary Lattices

$\mho$-Resolution works well for annotated logic programs, however its importance depends on the size of the class of ordinary lattices. If there are few applications which use ordinary lattices, $\mho$-resolution loses its significance as a new inference rule. In this section, we give two subclasses of ordinary lattices.

First, it is easily shown that all linear lattices are ordinary lattices. This class of lattices is particularly important to applications of annotated logic for quantitative reasoning. Of course, when the truth domain is linearly ordered, $\mho$-resolution just specializes to annotated resolution, and reduction is not necessary.

A much richer class of truth domains is the class of distributive lattices. A number of applications of annotated logics involve distributive lattices [17]. We show that all finite distributive lattices are ordinary lattices. We conjecture that this result generalizes to infinite distributive lattices.

Definition 16 (Distributive Lattices [4]) A lattice Δ is said to be *distributive* if it satisfies the **distributive law**.

$$(\forall \alpha, \beta, \gamma \in \Delta) \ \ \mathrm{Inf}\{\alpha, \ (\mathrm{Sup}\{\beta, \gamma\})\} = \mathrm{Sup}\{(\mathrm{Inf}\{\alpha, \beta\}), \ (\mathrm{Inf}\{\alpha, \gamma\})\}$$

The following is an equivalent condition.

$$(\forall \alpha, \beta, \gamma \in \Delta) \ \ \mathrm{Sup}\{\alpha, \ (\mathrm{Inf}\{\beta, \gamma\})\} = \mathrm{Inf}\{(\mathrm{Sup}\{\alpha, \beta\}), \ (\mathrm{Sup}\{\alpha, \gamma\})\}$$

The default-lattice of Figure 4 is an ordinary lattice and it can be shown that it also is a distributive lattice. The following theorem relates finite distributive lattices to ordinary lattices.

Theorem 17 All finite distributive lattices are ordinary lattices.

Proof. The proof is by contradiction. Assume Δ is a finite distributive lattice. Further, assume Δ is not an ordinary lattice, then for some $\mu, \rho \in \Delta$, $\mho(\mu, \rho) \notin \mathcal{M}(\mu, \rho)$. We know $\mid \mathcal{M}(\mu, \rho) \mid \geq 1$ since $\mu \in \mathcal{M}(\mu, \rho)$. Moreover, $\mid \mathcal{M}(\mu, \rho) \mid \geq 2$ since otherwise $\mathcal{M}(\mu, \rho)$ is a singleton and a least element exists. Since Δ is finite, $\mathcal{M}(\mu, \rho)$ is finite. Hence, by the assumption that Δ is not an ordinary lattice, there exist $\gamma_1, \gamma_2 \in \mathcal{M}(\mu, \rho)$ such that $\mathrm{Inf}\{\gamma_1, \gamma_2\} \notin \mathcal{M}(\mu, \rho)$. But Δ is distributive, so the following equation must hold.

$$\mathrm{Sup}\{\rho,\ (\mathrm{Inf}\{\gamma_1, \gamma_2\})\} = \mathrm{Inf}\{(\mathrm{Sup}\{\rho, \gamma_1\}),\ (\mathrm{Sup}\{\rho, \gamma_2\})\}. \qquad \text{Eq.(1)}$$

Since $\gamma_1, \gamma_2 \in \mathcal{M}(\mu, \rho)$, we know $\mathrm{Sup}\{\rho, \gamma_1\} \succeq \mu$ and $\mathrm{Sup}\{\rho, \gamma_2\} \succeq \mu$. Moreover $\mathrm{Inf}\{(\mathrm{Sup}\{\rho, \gamma_1\}), (\mathrm{Sup}\{\rho, \gamma_2\})\} \succeq \mu$. Thus substituting from Eq.(1), we have $\mathrm{Sup}\{\rho, (\mathrm{Inf}\{\gamma_1, \gamma_2\})\} \succeq \mu$, which implies $\mathrm{Inf}\{\gamma_1, \gamma_2\} \in \mathcal{M}(\mu, \rho)$, contradicting the premise. $\Box$

The finiteness restriction is important in the proof because we are not guaranteed that two minimal elements γ_1, γ_2 are in the set $\mathcal{M}(\mu, \rho)$ for infinite distributive lattices. This may occur, for example, when there exists an open interval I within $\mathcal{M}(\mu, \rho)$ and the least element of I does not exist in $\mathcal{M}(\mu, \rho)$.

4 Future work

A short range goal is to establish the conjecture that all distributive lattices are ordinary lattices. Further characterizing the class of ordinary lattices should provide useful insights into regular signed logics and implementation issues.

A more difficult problem is to investigate $\mho$-resolution in the context of general annotated logics, i.e., where non-Horn formulas are present, and non-linear deductions are allowed. In particular, $\mho$-resolution should be investigated with respect to systems in which reductants can be cached. Also, we plan to compare the *search* space of $\mho$-resolution with that of annotated resolution. This is important for practical implementation of annotated logic deduction systems.

References

[1] Baaz, M., and Fermüller, C.G., Resolution-based theorem proving for many-valued logics, *J. of Symbolic Computation*, 19(4):353–391, 1995.

[2] Blair, H.A. and Subrahmanian, V.S., Paraconsistent Logic Programming, *Theoretical Computer Science*, 68:135–154, 1989.

[3] Carnielli, W.A., Systematization of finite many-valued logics through the method of tableaux. *J. of Symbolic Logic* 52(2):473–493, 1987.

[4] Davey, B.A., and Priestley, H.A., *Introduction to Lattices and Order*, Cambridge Mathematical Textbooks, (1990)

[5] Doherty, P., *NML3 – A Non-Monotonic Formalism with Explicit Defaults.* Linköping Studies in Science and Technology. Dissertations No. 258. Department of Computer and Information Science, Linköping University, 1991.

[6] Fitting, M., First-order modal tableaux, *J. Automated Reasoning* 4:191-213, 1988.

[7] Ginsberg, M.L., Multivalued Logics, *Readings in Non-Monotonic Reasoning*, 251–255, 1987.

[8] Hähnle, R., *Automated Deduction in Multiple-Valued Logics*, International Series of Monographs on Computer Science, vol. 10. Oxford University Press, 1994.

[9] Hähnle, R., Short conjunctive normal forms in finitely-valued logics, *J. of Logic and Computation*, 4(6): 905–927, 1994.

[10] Hähnle, R., Tableaux methods for many-valued logics, *Handbook of Tableau Methods* (M d'Agostino, D. Gabbay, R. Hähnle and J. Posegga eds.), Kluwer, Dordrecht. To appear.

[11] Kifer, M., and Li, A., On the Semantics of Rule-based Expert Systems with Uncertainty, *Proceedings of the 2nd International Conference on Database Theory*, 102–117, 1988.

[12] Kifer, M., and Lozinskii, E.L., RI: A Logic for Reasoning in Inconsistency, *Proceedings of the Fourth Symposium of Logic in Computer Science*, Asilomar, 253-262, 1989.

[13] Kifer, M., and Subrahmanian, V.S., Theory of generalized annotated logic programming and its applications, the *J. of Logic Programming* 12, 335–367, 1992.

[14] Kifer, M., and Lozinskii, E., A logic for reasoning with inconsistency, *J. of Automated Reasoning* 9, 179–215, 1992.

[15] Leach, S.M., and Lu, J.J., Computing Annotated Logic Programs, *Proceedings of International Conference on Logic Programming*, MIT Press, 1994.

[16] Lu, J.J., Henschen, L.J., Subrahmanian, V.S., and da Costa, N.C.A., Reasoning in paraconsistent logics, *Automated Reasoning: Essays in Honor of Woody Bledsoe* (R. Boyer ed.), Kluwer Academic, 181–210, 1991.

[17] Lu, J.J., Nerode, A. and Subrahmanian, V.S. Hybrid Knowledge Bases, *IEEE Transactions on Knowledge and Data Engineering*, 8(5):773–785, 1996.

[18] Lu, J.J., Murray, N.V., and Rosenthal, E., A Framework for Automated Reasoning in Multiple-Valued Logics, *J. of Automated Reasoning.* To appear.

[19] Lu, J.J., Logic programming with signs and annotations, *J. of Logic and Computation*, 6(6):755 – 778, 1996.

[20] Murray, N.V., and Rosenthal, E., Adapting classical inference techniques to multiple-valued logics using signed formulas, *Fundamenta Informaticae* 21:237–253, 1994.

[21] Ramesh, A., Murray, N.V., Parameterized Prime Implicant/Implicate Computations for Regular Logics, To appear, *Mathware & Soft Computing*, Special Issue on Deduction in Many-Valued Logics.

[22] Subrahmanian, V.S., On the Semantics of Quantitative Logic Programs, in: *Proceedings of the 4th IEEE Symposium on Logic Programming*, Computer Society Press, 1987.

[23] Subrahmanian, V.S., Paraconsistent Disjunctive Databases, *Theoretical Computer Science*, 93, 115–141, 1992.

[24] Thirunarayan, K., and Kifer, M., A Theory of Nomonotonic Inheritance Based on Annotated Logic, *Artificial Intelligence*, 60(1):23–50, 1993.