MECHANICAL PROOF PROCEDURES FOR MANY-VALUED LATTICE-BASED LOGIC PROGRAMMING

by

V.S. SUBRAHMANIAN

Abstract

Recent results of Blair, Brown and Subrahmanian [2] and independently, M. Fitting [10,11] have shown that the declarative semantics of logic programs when interpreted over sets of truth values possessing some simple lattice theoretic properties shows remarkably little change. We prove here that the operational semantics (i.e., proof procedures) for such languages also show remarkably little change. The principal result is that under a natural condition of support a straightforward generalization of SLD-resolution is sound and complete w.r.t. processing of queries over these differing logics.

1 INTRODUCTION

There has been an intense effort over the last year directed primarily at generalizing the declarative semantics of logic programs.

For some time, there was a proliferation of logic programming languages based on different logics (e.g., many-valued, temporal, modal, epistemic, intuitionistic, etc.). For instance, many different many-valued logics were proposed, and in each case, the model theoretic semantics for programs was characterizable in terms of the prefixed-points of a monotone operator from interpretations to interpretations. This led different researchers to realize that the declarative semantics of logic programming languages based on these dif-

---

1 This work was supported by U.S. Air Force Contract F30602-85-C-0008.
fering logics were closely linked together. A simple investigation yielded the existence of a unifying framework – this was discovered independently by M.Fitting [10,11] and Blair, Brown and Subrahmanian [2]. There are some differences between these two formalizations, but the different is not important for our purposes.

What this attempt at generalization did not yield was a generalization of the operational mechanism of SLD-resolution to these many-valued logics. That there is some such generalization seems intuitively apparent. For instance, Blair and Subrahmanian [6] showed that a version of SLD-resolution called SLDa-resolution is complete for a class of programs (the so-called covered programs) in logics whose set of truth values form a complete lattice. Subrahmanian [14] showed how a version of SLD-resolution can be used to yield a complete proof procedure for quantitative rule sets.

In both the above cases, the modification of SLD-resolution depended intimately on the syntactic structure of annotated programs and QRSs respectively. These two formalisms are syntactically very different. Annotated programs are clauses of the form

\[ A_i : \mu_i \leftarrow A_1 : \mu_1 \& \ldots \& A_n : \mu_n \]

where each \( A_i, 0 \leq i \leq n \) is an atom. Here, a truth value (the \( \mu_i \)'s) is attached to each atom in the clause. On the other hand, a QRS is a set of clauses of the form:

\[ \mu : A_i \leftarrow A_1 \& \ldots \& A_n \]

Here, the truth value (i.e. \( \mu \)) is associated not with each atom in the clause, but with the clause itself.

The question we are concerned with is: How can we liberate the completeness results for lattice-based logic programming from the syntactic way in which truth values occur in these lattice-based programs? We show that under a natural condition requiring support for truth value assignments, the generalization can be achieved.

The paper is organized as follows: in the next section, we precisely define some of the terminology used so far. In Section 3, we specify a Scheme for SLD-Resolution. Instantiating this scheme to different logics will yield a proof procedure that is sound and complete w.r.t. processing of queries to logic programs interpreted over
these logics. We prove soundness and completeness theorems.

2 DECLARATIVE SEMANTICS

In this section, we describe the model-theoretic meaning and the fixed-point semantical foundations of multi-valued logic programming.

DEFINITION 1 If $p$ is an $n$-ary predicate symbol, and $t_1, \ldots, t_n$ are terms, then $p(t_1, \ldots, t_n)$ is an atom. If $A$ is an atom, then $A$ and $\neg A$ are literals.

DEFINITION 2 If $A_0, \ldots, A_n$ are atoms, then

$$A_0 \leftarrow A_1 \& \cdots \& A_n$$

is a clause. Any variable symbol occurring in a clause is assumed to be implicitly universally quantified. $A_0$ is called the head of this clause, while $(A_1 \& \cdots \& A_n)$ is called the body.

EXAMPLE 1 The sentence

$$rich(X) \leftarrow salary(X, Sal) \& Sal > 100000.$$ 

is a clause that says that if $X$ has a salary of over 100,000 dollars, then $X$ is rich.

We assume that we are interested in a fixed, but arbitrary many-valued logic whose set of truth values, denoted $\mathbb{T}$, forms a complete lattice under the ordering $\leq$. Thus far, there have been two proposals in the literature for syntactically extending the definition of a clause to accommodate many-valued logic programming. The first is due to Ishizuka and Kanai [9], while the second is due to Subrahmanian [13]. Both depend critically upon $\mathbb{T}$.

DEFINITION 3 A $c$-clause is a sentence of the form:

$$\mu : A_0 \leftarrow A_1 \& \cdots \& A_n$$

where $\mu \in \mathbb{T}$.
Intuitively, $\mu$ is a truth value that is associated with the clause. Thus, for instance, if $T$ is the unit interval $[0,1]$ of reals, and we say:

$$0.3 : \text{rich}(X) \leftarrow \text{has} - \text{big} - \text{house}(X)$$

then we may mean one of many different things. Two possibilities are:

1. we could mean that the degree of belief in the sentence "if $X$ has a big house, then $X$ is rich" is $30\%$ (this is the view taken in [14]) or

2. if the degree of belief in the proposition "X has a big house" is $\alpha$, then the degree of belief in the proposition "X is rich" is at least $\delta(0.3, \alpha)$. Here $\delta$ is some unspecified combination function. (This view is taken by Van Emden [16], but his notion is too restrictive as he allows $\delta$ to be only multiplication. There is, in fact, no hope of using multiplication over arbitrary lattices.) A generalization of this was proposed by Subrahmanian [15].

The two possibilities listed above depend crucially on $T$ being the unit interval $[0,1]$ of reals. Our aim is to generalize this considerably. The other kind of clause defined in the literature is of this form:

**DEFINITION 4** If $A_0, \ldots, A_n$ are atoms, and $\mu_0, \ldots, \mu_n$ are in $T$, then

$$A_0 : \mu_0 \leftarrow A_1 : \mu_1 \& \ldots \& A_n : \mu_n$$

is an $a$-clause (short for annotated-clause).

Clauses of this kind have been proposed by Subrahmanian [13,6] for many-valued logic programming. As we shall show later, one can obtain completeness results for logic programs consisting of each of the above kinds of clauses.

**DEFINITION 5** An $a$-logic program (resp. $c$-logic program) is a finite set of $a$-clauses (resp. $c$-clauses). We use the abbreviation ALP for an $a$-logic program and ACLP for a $c$-logic program.
In logic programming, we are primarily interested in Herbrand models, and therefore, we will restrict our interest to interpretations that are Herbrand-like. The Herbrand Base of an ACLP $P$, denoted by $B_P$, is the set of all variable-free atomic formulas that can be constructed from the constant, function and predicate symbols occurring in $P$. We make the assumption that $P$ contains at least one constant symbol.

**DEFINITION 6** Suppose $\mathcal{T}$ is a complete lattice under the ordering $\leq$. An Herbrand interpretation $I$ of the language $\mathcal{L}_P$ of the program $P$ is a map from $B_P$ to $\mathcal{T}$.

If $P$ is an ALP, we assume that there is a binary relation $=$ between interpretations of $\mathcal{L}_P$ and variable-free atomic formulas of $\mathcal{L}_P$. Once such a relation has been selected, it is extended to a binary relation on wffs as follows: (in the sequel, $F_1, F_2$ are sentences)

\[ A1 \quad I \models F_1 \& F_2 \text{ iff } I \models F_1 \text{ and } I \models F_2 \]

\[ A2 \quad I \models F_1 \lor F_2 \text{ iff } I \models F_1 \text{ or } I \models F_2 \]

\[ A3 \quad I \models F_1 \leftarrow F_2 \text{ iff } I \models F_1 \text{ or } I \not\models F_2 \]

\[ A4 \quad I \models (\forall x) F \text{ iff } I \models F(x/t) \text{ for all variable free terms } t. \text{ (Here } F(x/t) \text{ is the substitution of the variable-free term } t \text{ for all free occurrences of } x \text{ in } F.) \]

\[ A5 \quad I \models (\exists x) F \text{ iff } I \models F(x/t) \text{ for some variable free terms } t. \]

Thus a $\mathcal{T}$-valued logic is simply an association of such a binary relation between interpretations and the Herbrand Base of $P$. Once this binary relation has been defined, it is extended to arbitrary (negation-free) formulas in the natural way as specified by (A1)-(A5) below.

**DEFINITION 7** Suppose $\mathcal{T}$ is a complete lattice under the ordering $\leq$. We extend this ordering point-wise to the set of interpretation of the language $\mathcal{L}_P$ of $P$ as follows: $I_1 \leq I_2$ iff $I_1(A) \leq I_2(A)$ for all
\( A \in B_P \).

**DEFINITION 8** We say that a logic \( L \) is *suitable for programming* iff for all \( A \in B_P, I_1 \leq I_2 \) and \( I_1 \models A \) implies \( I_2 \models A \).

Unless explicitly stated otherwise, throughout this paper, we assume that all logics we are dealing with are suitable for programming.

**EXAMPLE 2** Suppose \( P \) is an ACLP and \( \mathcal{T} \) is the unity interval \([0,1]\). For each \( 0 \leq r \leq 1 \), we define a relation \( \models_r \) such that \( I \models_r A \) iff \( I(A) \geq r \) where \( I \) is an interpretation and \( A \in B_P \). It is easy to verify that the logic determined by the relation \( \models_r \) is suitable for programming.

**EXAMPLE 3** Suppose \( P \) is an ALP (relative to the set \( \mathcal{T} \) of truth values). Define \( \models \) to be the relation: \( I \models A : \mu \) iff \( I(A) \geq \mu \). The logic thus determined is suitable for programming.

**DEFINITION 9** Suppose \( P \) is an ALP. Then we associate with \( P \), an operator \( T_P \) from interpretations to interpretation such that \( T_P(I)(A) = \sqcup \{ \mu \mid A : \mu \leftarrow B_1 : \mu_1 \land \cdots \land B_n : \mu_n \ \text{is a variable-free instance of an a-clause in} \ P \ \text{and} \ I \models B_1 : \mu_1 \land \cdots \land B_n : \mu_n \} \).

**THEOREM 1** Suppose \( P \) is an ALP. Then:

1. \( I \) is a model of \( P \) iff \( T_P(I) \leq I \).
2. \( T_P \) is monotone.
3. \( T_P \) has a least fixed-point denoted \( lfp(T_P) \), and \( lfp(T_P) = T_P \uparrow \omega \). \( T_P \uparrow 0 \) is the interpretation that assigns \( \bot \) to all \( A \in B_P \) where \( \bot \) is the least element of \( \mathcal{T} \), \( T_P \uparrow \alpha = T_P(T_P \uparrow (\alpha - 1)) \) where \( \alpha \) is any successor ordinal, and \( T_P \uparrow \lambda = \sqcup_{\beta < \lambda} (T_P \uparrow \beta) \) where \( \lambda \) is a limit ordinal).
4. If \( A \) is in \( B_P \) and \( \nu \in \mathcal{T} \), then \( P \models A : \mu \) iff \( T_P \uparrow \omega(A) \geq \mu \).
5. If \( T_P \uparrow \omega(A) \geq \mu \), then there is some integer \( n \) such that \( T_P \uparrow n(A) \geq \mu \).
PROOF. (1) Suppose \( I \) is a model of \( P \) and \( T_P(I)(A) = \mu \). If \( \mu = \bot \), then \( T_P(I)(A) \leq I(A) \) is trivially true. Otherwise there is a finite set \( \Gamma \) of ground instances of clauses in \( P \) where \( \Gamma = \{ A : \mu_1 \leftarrow B_{1}^{1} : \rho_{1}^{1} \& \cdots \& B_{m_1}^{1} : \rho_{m_1}^{1} \}
\)
\[
\cdots
\]
\[
A : \mu_k \leftarrow B_{1}^{k} : \rho_{1}^{k} \& \cdots \& B_{m_k}^{k} : \rho_{m_k}^{k}
\}
\]
such that \( \mu = \sqcup \{\mu_1, \ldots, \mu_k\} \) and \( I = B_{1}^{i} : \rho_{1}^{i} \& \cdots \& B_{m_i}^{i} : \rho_{m_i}^{i} \), for all \( 1 \leq i \leq k \). But \( I \) is a model of \( P \), and hence, \( I(A) \geq \mu \), for all \( 1 \leq i \leq k \). Thus, \( I(A) \geq \sqcup \{\mu_1, \ldots, \mu_k\} = T_P(I)(A) \).

To see that \( T_P(I) \leq I \) implies that \( I \) is a model of \( P \), let
\[
A : \mu \leftarrow B_{1} : \rho_{1} \& \cdots \& B_{k} : \rho_{k}
\]
be a ground instance of a clause in \( P \) such that \( I \models B_{1} : \rho_{1} \& \cdots \& B_{k} : \rho_{k} \). Clearly, then \( T_P(I)(A) \geq \mu \). Thus, as \( I \geq T_P(I) \), we know that \( I(A) \geq \mu \), i.e., \( I \) is a model of the above clause. Therefore \( I \) is a model of \( P \). This completes the proof of (1).

(2) Straightforward.

(3) To see that \( T_P \uparrow \omega \leq \text{lfp}(T_P) \) is a straightforward consequence of the monotonicity of \( T_P \). So it suffices to show that \( \text{lfp}(T_P) \leq T_P \uparrow \omega \). Suppose \( A \) is any ground atom. Then:

\[
(D) \quad T_P \uparrow 0(A) \leq T_P \uparrow 1(A) \leq T_P \uparrow 2(A) \leq \cdots
\]
is an ascending chain as \( T_P \) is monotonic. Moreover, each \( T_P \uparrow n(A) \) is either \( \bot \) or the lub of a subset of the set \( \Lambda = \{ \mu \mid A' : \mu \text{ is the head of a clause in } P \text{ such that } A' \text{ and } A \text{ are unifiable} \} \). As \( \Lambda \) is finite, each \( T_P \uparrow n(A) \) is the lub of a finite subset of \( \Lambda \). As \( \Lambda \) is
finite, there are only finitely many such subsets, and hence there is an integer \( k \) such that

\[
T_P \uparrow k(A) = T_P \uparrow (k + 1)(A) = T_P \uparrow (k + 2)(A) = \ldots
\]

Thus, \( T_P \uparrow \omega(A) = T_P \uparrow k(A) = \emptyset \{\psi_1, \ldots, \psi_n\} \) where

\[
A : \psi_1 \leftarrow B_1^1 : \rho_1^1 \& \ldots \& B_{m_1}^1 : \rho_{m_1}^1
\]

\[ \ldots \]

\[
A : \psi_n \leftarrow B_n^n : \rho_1^n \& \ldots \& B_{m_n}^n : \rho_{m_n}^n
\]

are finitely many ground instances of clauses in \( P \) such that \( T_P \uparrow (k - 1) \models B_i^i : \rho_i^1 \& \ldots \& B_{m_i}^i : \rho_{m_i}^i \) for all \( 1 \leq i \leq n \). For each \( 1 \leq i \leq n \) and each \( 1 \leq j \leq m_i \), there is an integer \( r_{i,j} \) such that interpretation

\[
T_P \uparrow r_{i,j}(B_j^i) \geq \rho_j^i
\]

As there are only finitely many such \( B_j^i \)'s and as \( D \) is an ascending chain, there is some integer \( r \) such that

\[
T_P \uparrow r(B_j^i) \geq \rho_j^i
\]

for all \( 1 \leq i \leq n \) and \( 1 \leq j \leq m_i \). But then \( T_P \uparrow (r + 1)(A) \geq \emptyset \{\psi_1, \ldots, \psi_n\} = \mu = T_P \uparrow \omega(A) \). This completes the proof.

(4) If \( P \models A : \mu \), then clearly \( T_P \uparrow \omega(A) = \text{lfp}(T_P)(A) \geq \mu \) as \( \text{lfp}(T_P) \) is a model of \( P \). Suppose now that \( \text{lfp}(T_P)(A) \geq \mu \). But each model \( M \) of \( P \) is such that \( \text{lfp}(T_P) \leq M \). Hence, \( T_P \uparrow \omega(A) = \text{lfp}(T_P)(A) \leq M(A) \). Hence, \( P \models A : \mu \).

(5) Immediate from the proof of (3) above. \( \Box \)

How do the above results change for ACLPs? If the combination function \( \delta \) satisfies some simple conditions, then the same results hold for ACLPs.

**DEFINITION** 10 Suppose \( \mathcal{T} \) is a complete lattice of truth values
under the ordering \( \leq \). The binary function \( \delta : \mathbb{T} \times \mathbb{T} \to \mathbb{T} \) is monotone iff whenever \( x_1 \leq x_2 \) and \( y_1 \leq y_2 \) then \( \delta(x_1, y_1) \leq \delta(x_2, y_2) \).

**EXAMPLE 4** When \( \mathbb{T} \) is the unit interval \([0, 1]\), then multiplication is monotone. Thus, the combination function of Van Emden [16] is monotone.

For ACLPs, interpretations assign truth values to complex formulas as defined below:

\[
(C1) \quad I(F_1 \& F_2) = \inf(I(F_1), I(F_2))
\]

\[
(C2) \quad I(F_1 \lor F_2) = \sup(I(F_1), I(F_2))
\]

\[
(C3) \quad I(\exists x F) = \sup\{I(F(x/t)) \mid t \text{ is a ground term}\}
\]

We say an interpretation \( I \) is a model of the ground \( c \)-clause (w.r.t. the combination function \( \delta \))

\[
\mu : A \rightarrow B_1 \& \cdots \& B_n
\]

iff \( \delta(I(B_1 \& \cdots \& B_n), \mu) \geq I(A) \). An interpretation \( I \) is a model of a \( c \)-clause \( C \) iff \( I \) is a model of each ground instance \( C' \). And finally, \( I \) is a model of an ACLP \( P \) iff \( I \) is a model of each \( c \)-clause in \( P \).

**DEFINITION 11** Suppose \( P \) is an ACLP and \( \delta \) is a combination function. We associate with \( P \) an operator \( S_P \) from interpretations to interpretations such that \( S_P(I)(A) = \cup \{\delta(\alpha, r) \mid \alpha : A \rightarrow B_1 \& \cdots \& B_n \text{ is a ground instance of a clause in } P \text{ and } I(B_1 \& \cdots \& B_n) = r\} \).

**DEFINITION 12** An ACLP \( P \) is said to be strongly canonical iff for every \( n > 0 \) and every \( A \in B_L \), the set \( \{m \mid (m > n) \& S_P \uparrow m(A) \geq S_P \uparrow n(A)\} \) is finite.

Observe that if \( P \) is strongly canonical, then for all \( A \in B_F \), there is an integer \( n \) such that \( S_P \uparrow \omega(A) = S_P \uparrow n(A) \).
THEOREM 2 Suppose $P$ is an ACLP and $\delta$ is a monotone binary function. Then:

1. $I$ is a model of $P$ if and only if $S_P(I) \subseteq I$.

2. $S_P$ is monotone.

3. If $P$ is strongly canonical, then $\text{lfp}(S_P) = S_P \uparrow \omega$.

4. Suppose $P$ is strongly canonical and $A \in B_P$ and $\mu \in \mathbb{R}$. Then $P \models_\mu A$ if and only if $S_P \uparrow \omega(A) \geq \mu$. (Here, $P \models_\mu A$ if for every model $I$ of $P$, it is the case that $I(A) \geq \mu$.)

PROOF:

(1) ($\to$) Suppose $I$ is a model of $P$ and that $S_P(I)(A) = \mu \neq \bot$. Let $C_1, C_2, \ldots$ be all ground instances of clauses in $P$ such that each $C_i$ has $A$ as the head. The each $C_i$ is of the form:

$$\rho_i : A \leftarrow B_1^i \land \cdots \land B_m^i.$$  

Moreover,

$$\mu = \bigsqcup_{i \geq 1} \delta(I(B_1^i \land \cdots \land B_m^i), \rho_i).$$

Thus, as $I$ is a model of $P$, and hence of each $C_i$, we know that

$$I(A) \geq \delta(I(B_1^i \land \cdots \land B_m^i), \rho_i)$$

for all $i \geq 1$. Hence,

$$I(A) \geq \left( \bigsqcup_{i \geq 1} \delta(I(B_1^i \land \cdots \land B_m^i), \rho_i) \right) = \mu.$$

(\leftarrow) Suppose $S_P(I) \subseteq I$. Let

$$\rho : A \leftarrow B_1 \land \cdots \land B_n$$

be a ground instance of a clause in $P$. Then, by definition of $S_P$,

$$\delta(I(B_1 \land \cdots \land B_n), \rho) \leq S_P(I)(A).$$
But $S_{\tau}(I)(A) \leq I(A)$. Hence,

$$\delta(I(B_1 \& \cdots \& B_n), \rho) \leq S_{\tau}(I)(A) \leq I(A).$$

Hence, $I$ is a model of $P$.

(2) Suppose $I_1 \leq I_2$ and $S_{\tau}(I_1)(A) = \mu \neq \bot$. Let $C_1, C_2, \ldots$ be all ground instances of clauses in $P$ such that each $C_i$ has $A$ as the head. Then each $C_i$ is of the form:

$$\rho_i : A \leftarrow B_i^{i_1} \& \cdots \& B_{m_i}^{i_1}.$$  

Moreover,

$$\mu = \bigsqcup_{i \geq 1} \delta(I_1(B_1^{i_1} \& \cdots \& B_{m_1}^{i_1}), \rho_i).$$

As $\delta$ is monotone, and as $I_1 \leq I_2$,

$$\delta(I_1(B_1^{i_1} \& \cdots \& B_{m_1}^{i_1}), \rho_i) \leq \delta(I_2(B_1^{i_1} \& \cdots \& B_{m_1}^{i_1}), \rho_i)$$

for all $i \geq 1$. Hence,

$$S_{\tau}(I_2)(A) \geq \delta(I_2(B_1^{i_1} \& \cdots \& B_{m_1}^{i_1}), \rho_i)$$

for all $i \geq 1$. Hence,

$$S_{\tau}(I_2)(A) \geq \bigsqcup_{i \geq 1} \delta(I_2(B_1^{i_1} \& \cdots \& B_{m_1}^{i_1}), \rho_i) \geq$$

$$\geq \bigsqcup_{i \geq 1} \delta(I_1(B_1^{i_1} \& \cdots \& B_{m_1}^{i_1}), \rho_i) = S_{\tau}(I_1)(A).$$

(3) Suppose $P$ is strongly canonical. As $S_{\tau}$ is monotonic, clearly $S_{\tau} \uparrow \omega \leq lfp(S_{\tau})$. We only need show that $S_{\tau} \uparrow \omega$ is a fixed-point of $S_{\tau}$. Suppose $S_{\tau} \uparrow (\omega + 1)(A) = \mu$. If $\mu = \bot$, then $S_{\tau} \uparrow \omega(A) = S_{\tau} \uparrow (\omega + 1)(A)$ and we are done. So assume $\mu \neq \bot$. Then, by strong canonicality of $P$, there is some integer $n$ such that $S_{\tau} \uparrow n(A) = \mu$. Thus, for each clause in $P$ having a ground instance of the form:

$$\rho : A \leftarrow B_1 \& \cdots \& B_m$$
we know that
\[ \delta(S_P \uparrow \omega(B_1 \& \cdots \& B_m), \rho) \leq S_P \uparrow \omega(A) = S_P \uparrow n(A). \]
Hence, \( S_P \uparrow (\omega + 1)(A) = S_P(S_P \uparrow n)(A) = S_P \uparrow (n + 1)(A) = S_P \uparrow \omega(A). \)

(4) Immediate consequence of parts (1) and (3) of the above theorem. \( \square \)

EXAMPLE 5 The above theorem generalizes Van Emden's declarative semantics for QRSs [16]. Here, \( \mathcal{T} \) is a complete lattice, and certainly it is true that multiplication is commutative, associative and monotonic.

DEFINITION 13 An ALP \( P \) is said to be supported iff for all \( A \in B_P \) such that \( T_P \uparrow \omega(A) \neq \perp \), there is an a-clause in \( P \) having a ground instance of the form:
\[ A : \mu \leftarrow B_1 : \mu_1 \& \cdots \& B_n : \mu_n, \]
and \( T_P \uparrow \omega = B_1 : \mu_1 \& \cdots \& B_n : \mu_n \) and \( \mu > T_P \uparrow \omega(A) \).

\[ \begin{array}{c}
\top \\
\downarrow \\
f \\
\downarrow \\
\bot \\
\uparrow \\
t \\
\uparrow \\
\end{array} \]

Figure 1: Four Valued Lattice
EXAMPLE 6 Suppose $\tau$ is the four valued complete lattice shown in Figure 1, and $P$ is the following ALP:

$$A : f \leftarrow$$

$$A : t \leftarrow$$

Then $T_P \uparrow \omega(A) = \tau$, but clearly this program is not supported.

It may appear from the above example that supportedness is a restriction on ALPs, but this is not true.

DEFINITION 14 Suppose $C_1, C_2$ are clauses (standardized apart) of the form:

$$A^1 : \mu \leftarrow B^1_1 : \rho^1_1 \land \cdots \land B^1_k : \rho^1_k$$

$$A^2 : \mu \leftarrow B^2_1 : \rho^2_1 \land \cdots \land B^2_k : \rho^2_k$$

and there is a renaming substitution $\theta$ (i.e. $\theta$ replaces variable symbols by variable symbols only) such that

1. $A^1_\theta = A^2_\theta$ and

2. for each $1 \leq i \leq k$, there is a $1 \leq j \leq k$ such that: $B^1_i \theta = B^2_j$ and $\rho^1_i \leq \rho^2_j$.

then $C_1$ is said to subsume $C_2$. $C_1, C_2$ are subsumption equivalent iff $C_1$ subsumes $C_2$ and $C_2$ subsumes $C_1$.

Subsumption-equivalence is clearly decidable. Intuitively, if $C_1$ is subsumption equivalent to $C_2$, it just means that $C_2$ is identical to $C_1$ in all senses except that: (1) the variables of $C_1$ are renamed and (2) the atoms occurring in the body of $C_2$ possibly occur in an order that is different from the order in which the corresponding atoms occur in $C_1$.

THEOREM 3 Suppose $P$ is an ALP. Then there is an ALP $P'$ such that $P'$ is supported and such that $T_P \uparrow \omega = T_{P'} \uparrow \omega$.

PROOF. Suppose $C_1, C_2$ are (standardized apart) clauses in $P$ of the form:
MECHANICAL PROOF PROCEDURES FOR ... 

\[ \begin{align*}
A^1 : & \mu_1 \leftarrow B^1_1 : \rho^1_1 \& \cdots \& B^1_k : \rho^1_k \\
A^2 : & \mu_2 \leftarrow B^2_1 : \rho^2_1 \& \cdots \& B^2_k : \rho^2_k
\end{align*} \]

such that \( A^1, A^2 \) are unifiable via \text{mgu} \( \theta \) and \( \mu_1 \) and \( \mu_2 \) are incomparable (i.e. \( \mu_1 \not\leq \mu_2 \) and \( \mu_2 \not\leq \mu_1 \)). Then the \textit{mutant} of \( C_1, C_2 \), denoted \( M(C_1, C_2) \) is the clause:

\[ A^1 \theta \square \{ \mu_1, \mu_2 \} \leftarrow (B^1_1 : \rho^1_1 \& \cdots \& B^1_k : \rho^1_k B^2_1 : \rho^2_1 \& \cdots \& B^2_k : \rho^2_k)^\theta \]

Now define an operator \( \mathcal{C} \) from programs to programs as follows:

\[ \mathcal{C}(P) = \{ M(C_1, C_2) \mid C_1, C_2 \text{ are clauses in } P \text{ such that the head of } C_1 \text{ is annotated with } \mu_1, \text{ the head for } C_2 \text{ is annotated with } \mu_2 \text{ and } \mu_1 \not\leq \mu_2 \text{ and } \mu_2 \not\leq \mu_1 \text{ and } M(C_1, C_2) \text{ is defined and is not subsumption equivalent to any clause in } P \} \]

Now define the following iteration operator \( \mathcal{I} \):

\[ \begin{align*}
\mathcal{I}_0(P) & = P \\
\mathcal{I}_{n+1}(P) & = \mathcal{C}(\mathcal{I}_n(P)) - \mathcal{I}_n(P)
\end{align*} \]

It is easy to verify, as \( P \) has only finitely many clauses, that for some integer \( r \), \( \mathcal{I}_r(P) = \emptyset \). Call the least such \( r \) the \textit{closure integer} of \( \mathcal{I} \) and define the \textit{closure} \( CL(P) \) of \( P \) to be:

\[ CL(P) = \bigcup_{0 \leq r < r'} \mathcal{I}_{r'}(P) \]

It is easy to see that \( CL(P) \) is supported. Moreover, \( T_P = T_{\cup L(P)} \) and hence it follows that \( T_P \uparrow \omega = T_{\cup L(P)} \uparrow \omega \cdot CL(P) \) may then be taken as the desired program \( P' \).

\[ \Box \]

\textbf{EXAMPLE 7} Suppose \( P \) is the program considered in Example 6.
then the closure of \( P \) is \( P \cup \{ p : \top \leftarrow \} \).

\textbf{DEFINITION 15} Suppose \( P \) is an ACLP and \( \delta \) is monotone, associative and commutative. \( P \) is \textit{supported} (w.r.t. \( \delta \)) iff for all \( A \in B_P \) such that \( S_P \uparrow \omega(A) \neq \bot \), there is a clause in \( P \) having a ground instance of the form:
\[ \alpha : A \leftarrow B_1 \land \ldots \land B_n \]

and such that \( S_P \uparrow \omega(A) = \delta(\alpha, S_P \uparrow k(B_1 \land \ldots \land B_n)) \) for some \( k < \omega \).

**EXAMPLE 8** Van Emden's QRSs are all supported.

The remaining part of this paper is devoted to the design of a proof procedure (or query answering procedure) for logic programs (either ACLPs or ALPs) that are supported.

### 3 QUERY ANSWERING PROCEDURES

The fundamental thesis of logic programming is this: A logic program is a set of sentences in a logic. A query is satisfied by the program if it is a logical consequence of the program. Thus, in order to check whether a particular query is satisfied by a program, consider the query as a sentence which we are trying to prove from the program. If the logic we are considering is complete, then should be possible.

However, as logic programming has developed over the years, different proof procedures have been proposed by the individual re-searches. These proof procedures are "custom-made" for the new non-classical logics that are proposed for logic programming. Some of these are fundamentally different from SLD-resolution which is used in classical logic programming. We believe that *whenever possible*, this situation should be avoided.

We justify this belief in the following way. From a *pragmatic* point of view, it must be possible to process queries quickly and efficiently. If we rely on a mechanical proof procedure to do this, this means that this proof procedure must be efficiently implementable. SLD-resolution is, at the time of writing of this paper, undoubtedly the best theorem proving method for answering queries to *classical* logic programs. It avoids the pitfalls that lead to the inefficiency of automatic theorem provers based on other kinds of proof procedures. Now, non-classical logics certainly serve an important epistemological role. They provide us with a way of reasoning about beliefs. Such beliefs may be "fuzzy" or may indeed be inconsis-
tent [7]. Reasoning about irrational people is a daily feature of our lives. Our main point here is that this ability to reason in diverse ways about rational or irrational reasoning agents need not come at the expense of computational efficiency. It is precisely to emphasize this point that we now present a generalized Scheme for doing SLD-resolution in ACLPs and ALPs. We emphasize that all the many-valued logic programming languages that we are aware of at this point in time are either special cases of ACLPs or of ALPs. Thus, this Scheme is robust enough to provide a proof theory for each of these logic programming languages. Moreover, these are all precisely SLD-resolution like proof procedures.

DEFINITION 16 The logic program \( LP(P) \) associated with \( P \) (where \( P \) is either an ALP or an ACLP) is the pure logic program obtained by deleting all truth values occurring in clauses in \( P \).

EXAMPLE 9 Suppose \( P \) is the ALP containing the single clause:

\[
A : \mu \leftarrow B_1 : \mu_1 \& \cdots \& B_k : \mu_k
\]

Then \( LP(P) \) is:

\[
A \leftarrow B_1 \& \cdots \& B_k
\]

If \( P' \) is the ACLP containing the single clause:

\[
\mu : A \leftarrow B_1 \& \cdots \& B_k
\]

then \( LP(P) \) is:

\[
A \leftarrow B_1 \& \cdots \& B_k
\]

When \( P = \{C\} \) is an ALP or ACLP consisting of only one clause, then we use \( LP(C) \) to denote the single clause in \( LP(P) \). We now briefly recall the notion of SLD-resolution in classical logic programming.

DEFINITION 17 A query an existentially closed conjunction of atoms, i.e. if \( A_1, \ldots, A_n \) are atoms (or atoms annotated with truth values), then \( (\exists)(A_1 \& \cdots \& A_n) \) is a query. We will often write a query as just \( A_1 \& \cdots \& A_n \) and assume the existential quantification to be implicit.
DEFINITION 18 Suppose $Q \equiv A_1 \land \cdots \land A_n$ is a query, and $C \equiv
$ $A \leftarrow B_1 \land \cdots \land B_k$

is a clause such that $A$ and $A_i$ are unifiable (and their most general unifier, mgu for short, is $\theta$). Then

$$(A_1 \land \cdots \land A_{i-1} \land B_1 \land \cdots \land B_k \land A_{i-1} \land \cdots \land A_n)\theta$$

is a resolvent of $C$ and $Q$. (We assume here that $C$ and $Q$ share no common variable symbols – this can be easily achieved by renaming all bound variables.) When the restriction that $\theta$ is a most general unifier is weakened to require only that $\theta$ be a unifier, then

$$(A_1 \land \cdots \land A_{i-1} \land B_1 \land \cdots \land B_k \land A_{i-1} \land \cdots \land A_n)\theta$$

is a called an unrestricted resolvent of $C$ and $Q$.

DEFINITION 19 An SLD-deduction of the query $Q_0$ w.r.t. the logic program $P$ is a sequence

$$< Q_0, C_0, \theta_0 >, \ldots, < Q_i, C_i, \theta_i >$$

where $Q_{i+1}$ is a resolvent of the query $Q_i$ and the clause $C_i \in P$ via mgu $\theta_i$. When $Q_{i+1}$ need only be an unrestricted resolvent, then the above sequence is called an unrestricted SLD-deduction. An SLD-refutation of the query $Q_0$, from $P$ is a finite SLD-deduction

$$< Q_0, C_0, \theta_0 >, \ldots, < Q_n, C_n, \theta_n >$$

such that the resolvent of $Q_n$ and $C_n$ is the empty query, denote $\Box$. An unrestricted SLD-refutation is defined in the obvious way.

LEMMA 1 (MGU Lemma for SLD-Refutation) (Lloyd, [12]) Suppose $P$ is a logic program and $Q_0$ a query such that

$$< Q_0, C_0, \theta_0 >, \ldots, < Q_n, C_n, \theta_n >$$
is an unrestricted refutation of $Q_0$ from $P$. Then there is an SLD-Refutation

\[
< Q_0, C_0, \sigma_0 >, \ldots, < Q_n, C_n, \sigma_n >
\]

from $P$ and a substitution $\gamma$ such that

\[
\sigma_0 \cdots \sigma_n \gamma = \theta_0 \cdots \theta_n
\]

THEOREM 4 Suppose $P$ is an ALP and $\mathcal{T}$ is the complete lattice of truth values on which $P$ is based. If $lfp(T_P)(A) \neq \perp$, then there is an SLD-refutation of $A$ from $LP(P)$.

PROOF. Suppose $A$ is a ground atom such that $T_P \uparrow \omega(A) \neq \perp$. Then by the proof of Theorem 1 part (5), for some $n < \omega, T_P \uparrow n(A) = \mu > \perp$. We proceed by induction on $N$.

Base Case. ($n = 1$) Then $A : \mu' \leftarrow$ is a ground instance of a clause in $P$ such that $\perp < \mu' \leq \mu$. Clearly, then $A' \leftarrow$ is a unit clause in $LP(P)$ and Figure 2 shows an SLD-refutation of $A$ from $LP(P)$.

Inductive Case. ($n + 1$) Then

\[
A : \mu' \leftarrow B_1 : \psi_1 \& \cdots \& B_m : \psi_m
\]
is a ground instance of a clause $C$ in $P$ such that $\bot < \mu' \leq \mu$ and $T_P \vdash n \models B_1 : \psi_1 \& \cdots \& B_m : \psi_m$. By the inductive assumption, there is an SLD-refutation $\mathcal{R}_i$ of each $B_i$ from $P$. Hence, the $\mathcal{R}_i$'s can be combined into SLD-refutation $\mathcal{R}$ of $B_1 \& \cdots \& B_m$ from $P$. Now

$$A \leftarrow B_1 \& \cdots \& B_m$$

is a ground instance of the clause $LP(C)$ in $P$. $LP(C)$ is of the form

$$A' \leftarrow B'_1 \& \cdots \& B'_m$$

such that for some substitution $\theta$,

$$(A' \leftarrow B'_1 \& \cdots \& B'_m)\theta$$

is exactly

$$A \leftarrow B_1 \& \cdots \& B_m$$

Hence, there is an unrestricted refutation of $A$ from $P$ shown in Figure 3.

By the Mgu Lemma for SLD-Refutation, it follows that there is an SLD-Refutation of $A$ from $P$. \qed
DEFINITION 20 The combination function $\delta : \mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}$ is
strict iff whenever either $\mu_1 = \bot$ or $\mu_2 = \bot$, then $\delta(\mu_1, \mu_2) = \bot$. $\delta$ is
compact iff whenever

$$\mu = \bigcup \{ \delta(\mu, \rho), \ldots, \delta(\mu, \rho), \ldots \}$$

then there is a finite subset $A \subseteq \{1, 2, \ldots\}$ such that

$$\mu = \bigcup \{ \delta(\mu_i, \rho) \mid i \in A \}.$$

DEFINITION 21 An ACLP $P$ is closed iff whenever

$$\mu_1 : A \leftarrow B_1 \& \ldots \& B_r,$$

$$\mu_2 : A \leftarrow D_1 \& \ldots \& D,$$

are ground instance sof clauses in $P$, then for each interpretation $I$, there is a clause $C$ in $P$ having a ground instance of the form

$$\rho : A \leftarrow E_1 \& \ldots \& E_w$$

such that

$$\delta(I(E_1 \& \ldots \& E_w), \rho) =$$

$$\delta(I(B_1 \& \ldots \& B_r), \mu_1) \sqcup \delta(I(D_1 \& \ldots \& D), \mu_2).$$

THEOREM 5 Suppose $P$ is a strongly canonical, closed ACLP, $\mathcal{T}$ is the complete lattice of truth values on which $P$ is based, and the combination function $\delta$ is strict and compact. If $\text{lfp}(S_P)(A) \neq \bot$, then there is an SLD-refutation of $A$ from $LP(P)$.

PROOF. As $P$ is strongly canonical, there is a least integer $n > 0$ such that $\text{lfp}(S_P)(A) = S_P \uparrow n(A)$. We proceed by induction on $n$.

Base Case. $(n = 1)$ Then there is a clause in $P$ having a ground instance

$$\mu : A \leftarrow B_1 \& \ldots \& B_n,$$

such that

$$\delta(S_P \uparrow 0(B_1 \& \ldots \& B_n), \mu) > \bot.$$
As $\delta$ is strict, $n$ must be 0. Then $A \leftarrow$ is a ground instance of a unit clause $C$ in $LP(P)$. Clearly, the query $A$ and $C$ resolve to yield the empty clause.

**Inductive Case.** $(n + 1)$ Suppose $S_{\mathcal{P}} \vdash (n + 1)(A) = \mu$. Then there is some integer $k$ and sets of ground clauses $\mathcal{F}_i (1 \leq i \leq k)$ where each clause in $\mathcal{F}_i$ is a ground instance of some clause in $P$ and is of the form:

$$\mu_i : A \leftarrow B_1^{i,j} \& \cdots \& B_{s_i}^{i,j}.$$ 

Thus, any two clauses in $\mathcal{F}_i$ have the same attenuation factor, denoted $\mu_i$. Thus,

$$\mu = \bigcup \left( \bigcup_{i=1}^{k} \left( \bigcup \{ \delta(S_{\mathcal{P}} \vdash n(B_1^{i,j} \& \cdots \& B_{s_i}^{i,j}, \mu_i) \} \right) \right).$$

As $\Box(X \cup Y) = (\Box X) \cup (\Box Y)$, we know that

$$\mu = (\Box X_1) \cup \cdots \cup (\Box X_k)$$

where $X_i = \{ \delta(S_{\mathcal{P}} \vdash n(B_1^{i,j} \& \cdots \& B_{s_i}^{i,j}, \mu_i) \mid \text{where } \mu_i : A \leftarrow B_1^{i,j} \& \cdots \& B_{s_i}^{i,j} \text{ is a ground instance of a clause in } \mathcal{F}_i \}$. Thus, each $X_i$ is a set of lattice elements of the form $\delta(-, \mu_i)$. Note here that the second argument is fixed. hence, as $\delta$ is compact, there is a finite subset $Y_i$ of $X_i$ such that $\Box X_i = \Box Y_i$. Hence,

$$\mu = (\Box Y_1) \cup \cdots \cup (\Box Y_k)$$

As $P$ is closed, and as the above is a lub over a finite set, there is a single clause in $P$ having a ground instance of the form

$$\kappa : A \leftarrow D_1 \& \cdots \& D_r,$$

such that $\mu = \delta(S_{\mathcal{P}} \vdash n(D_1 \& \cdots \& D_r), \kappa)$. By the induction hypothesis, and as $\delta$ is strict, $\kappa \neq \bot$ and as $S_{\mathcal{P}} \vdash n(D_i \neq \bot$ for all $1 \leq i \leq r$, we know that there is a refutation of $(D_1 \& \cdots \& D_r)$ from $LP(P)$. Clearly then there is an SLD-Refutation of $A$ from $LP(P)$. 
COROLLARY 1 Suppose $P$ is a supported ACLP and $\delta$ is strict. If $A$ is a ground atom such that $\text{lp}(SP)(A) \neq \bot$, then there is an SLD-Refutation of $A$ from $LP(P)$. \hfill \Box

We now show how to process queries to ALPs.

ALP Query Processing Algorithm

INPUT: An ALP $P$ and some variable free atom $A \in B_P$

1. Construct $LP(P)$
2. Apply SLD-Resolution w.r.t. $LP(P)$ and $A$
3. Non-deterministically select an SLD-Refutation $< Q_0, C_0, \theta_0 >$, $\ldots$, $< Q_n, C_n, \theta_n >$ of $Q_0 \equiv A$ from $P$
4. Associate a truth value $\nu(A_j, Q_i)$ with each atom $A_j$ occurring in $Q_i$, all $1 \leq i \leq n$ as follows:
   (a) Let $A$ be the solitary atom in $Q_n$ and let $C_n \in LP(P)$ be of the form $A' \leftarrow$ such that $A'\theta_n = A\theta_n$. Then $\nu(A, Q_n) = \cup\{\mu \mid A' : \mu \leftarrow \}$ is a clause in $P$.
   (b) Suppose $i < n$. Let $A_j$ be an atom in $Q_i$. If $A_j$ is the atom on which $Q_i$ and $C_i$ resolve, then $\nu(A_j, Q_i) = \cup\{\mu \mid A' : \mu \leftarrow B_i : \mu_1 \& \ldots \& B_k : \mu_k \text{ is a clause in } P \text{ such that } A' \text{ and } A \text{ are unifiable via mgu } \nu \text{ and } C_i \text{ is } A' \leftarrow B_1 \& \ldots \& B_k \}$ and $\nu(B_r, Q_i) \leftarrow \mu_r$ for all $1 \leq r \leq k$. If $A_j$ is not the atom on which $C_i$ and $Q_i$ resolve, then $\nu(A_j, Q_i) = \nu(A_j, Q_{i+1})$.

End Algorithm

The first point to note here is that this algorithm may not always terminate. The reason for this is that step 2 above may not terminate. However, this is not particularly important. It is quite well known (cf. Blair [3,4]) that the set $SS(P) = \{ A \in B_P \mid P \equiv A \}$ where $P$ is a pure logic program may be a strictly recursively enumerable subset of $B_P$. Hence there is no algorithm at all that will
always terminate and return the appropriate answer. However, we will shortly show that this suffices as a sound and complete proof procedure. Before proceeding to establish soundness and completeness theorems, we present some simple example to demonstrate the working of the above algorithm.

EXAMPLE 10 Suppose $P$ is the ALP over the four valued lattice of Fig. 1. Let $P$ be the program consisting of two clauses:

\[
p : t \leftarrow \\
p : \neg p \leftarrow \\
\]

![Figure 4](image_url)

Then Figure 4 shows how truth values are associated with the queries in the SLD-tree from $LP(P)$:

EXAMPLE 11 Consider this slightly more complicated example. Let $T$ be the set of truth values $[0.1] \times [0.1]$ ordered as: $[x_1, y_1] \leq [x_2, y_2]$ iff $x_1 \leq x_2$ and $y_1 \leq y_2$. Let $P$ be the program:

(C1) $q(X) : [0.3, 0.3] \leftarrow p(X, Y) : [0.1, 0.4]$  
(C2) $q(a) : [0.2, 0.7] \leftarrow p(a, s(U)) : [0.1, 0.1]$  
(C3) $p(a, a) : [0.1, 0.7] \leftarrow$  
(C4) $p(a, s(Z)) : [0.3, 0.8] \leftarrow p(a, X) : [0.1, 0.1]$

Consider now the query $q(a)$. Its truth value in $T_P \uparrow \omega(A)$ is $[0.3, 0.7]$. One of the SLD-refutation trees associated with the atom $p(a)$, together with truth values associated via Step 4 of the ALP Query Processing Algorithm is shown in Figure 5. Note that this ALP is not supported.
THEOREM 6 (Soundness) Suppose $P$ is an ALP, $A$ is an atom, and the ALP Query Processing Algorithm, when started on input $P$ and $A$ returns $\nu(Q_0, A)$. Then $T_P \uparrow \omega(A) \geq \nu(Q_0, A)$.

PROOF. The proof proceeds by induction on the length $k$ of the SLD-Refutation selected in Step 3 of the ALP Query Processing Algorithm.

Base Case. ($k = 1$) Then the SLD-refutation is of the form:

$$< A, C_0, \theta_0 >$$

and so $C_0$ is a unit clause in $LP(P)$ of the form $A' \leftarrow$ where $\theta_0$ is the mgu of $A, A'$. Thus, there must be clauses in $P$ of the form:

$$(A' : \mu_i \leftarrow)_{1 \leq i \leq k}$$

where $\nu(Q_0, A) = \sqcup\{\mu_1, \ldots, \mu_k\}$. As $T_P \uparrow \omega$ is a model of $P, T_P \uparrow \omega(A) \geq \sqcup\{\mu_1, \ldots, \mu_k\}$. 

\[ \text{Figure 5} \]
Inductive Case. \((k + 1)\) Suppose the SLD-Refutation of \(A\) is of length \(+ 1\). Let
\[
< Q_0, C_0, \theta_0 >, \ldots, < Q_{k+1}, C_{k+1}, \theta_{k+1} >
\]
(where \(Q_0 = A\)) be the SLD-Refutation selected in Step 3 of the Algorithm. By the inductive assumption,
\[
< Q_1, C_1, \theta_1 >, \ldots, < Q_{k+1}, C_{k+1}, \theta_{k+1} >
\]
is an SLD-Refutation of \(Q_1\) from \(P\) of length \(k\) and so
\[
T_P \uparrow \omega(Q_1) \geq \nu(Q_1)
\]
But \(Q_1\) is the resolvent of \(Q_0\) and \(C_0\). Here, \(C_0\) is of the form
\[
A' \leftarrow B_1 \& \cdots \& B_m
\]
and \(Q_1 = (B_1 \& \cdots \& B_m) \theta_0\), where \(\theta_0\) is the mgu of \(A\) and \(A'\). Let
\[
A' : \mu_1 \leftarrow B_1 : \psi_1^1 \& \cdots \& B_m : \psi_m^1
\]
\[
\ldots
\]
\[
A' : \mu_r \leftarrow B_1 : \psi_1^r \& \cdots \& B_m : \psi_m^r
\]
be all the clauses in \(P\) from which \(C_i\) is derived. Then, as \(T_P \uparrow \omega(B_i) \geq \psi_i^j\) for all \(1 \leq j \leq r\) by the Inductive hypothesis, it follows by Step 4(b) of the ALP Query Processing Algorithm that
\[
\nu(Q_0, A) = \sqcup \{ \mu_1, \ldots, \mu_n \}
\]
But \(T_P \uparrow \omega(B_i) \geq \psi_i^j\) for all \(1 \leq i \leq m\) and \(1 \leq j \leq r\), (i.e. there are only finitely many such combinations of \(i, j\)), there is some integer \(r'\) such that \(T_P \uparrow r'(B_i) \geq \psi_i^j\) for all \(1 \leq i \leq m\). Thus, \(T_P \uparrow (r' + 1)(A) \geq \sqcup \{ \mu_1, \ldots, \mu_r \}\). Hence, \(T_P \uparrow \omega(A) \geq \sqcup \{ \mu_1, \ldots, \mu_r \}\). This completes the proof. \(\Box\)

THEOREM 7 (Completeness) Suppose \(P\) is a supported ALP such that \(\bot\) is not the annotation of any atom occurring in \(P\), \(A\) is an atomic query, and \(P \models A : \mu\) \((\mu \neq \bot)\). Then there is some selection of an SLD-Refutation in Step 3 of the ALP Query Processing Algorithm above such that \(\nu(Q_0, A) \geq \mu\).
PROOF. Suppose $P = A : \mu$. Then by Theorem 1 part (4), it follows that $T_P \uparrow \omega(A) \geq \mu$. Hence, by Theorem 1 part (5) there is some integer $n$ such that $T_P \uparrow n(A) \geq \mu$. We proceed by induction on $n$.

**Base Case.** $(n = 1)$ As $P$ is supported, and as $T_P \uparrow \omega(A) \geq \mu$, and as $\mu > \bot$, there is a single clause in $P$ having a ground instance of the form:

$$C : \quad A : \rho \leftarrow B_1 : \psi_1 \& \cdots \& B_k : \psi_k$$

such that $T_P \uparrow B_1 : \psi_1 \& \cdots \& B_k : \psi_k$ and $\rho \geq \mu$. By the assumption that $\bot$ doesn't occur in $P$, we know that $k = 0$. So clearly, there is a refutation of $A$ form $P$ (cf. Figure 6) such that $\nu(Q_{n+1}, A) \geq \rho \geq \mu$.

**Inductive Case.** $(n+1)$ Suppose now that $T_P \uparrow (n + 1)(A) \geq \mu$. Then, as $P$ is supported, there is a single clause $C$ in $P$ having a ground instance of the form:

$$A : \rho \leftarrow B_1 : \psi_1 \& \cdots \& B_r : \psi_r$$

such that $T_P \uparrow n(B_i) \geq \psi_i$ for all $1 \leq i \leq r$ and $\rho \geq \mu$. Thus, the query $Q'_{n+1}$:

$$B_1 : \psi_1 \& \cdots \& B_r : \psi_r$$

![Figure 6](image_url)
is satisfied by $T_P \uparrow n$. Hence, by the Inductive Assumption, the ALP Query Answering Algorithm returns values $\nu(Q'_i, B_i) \geq \psi$, for all $1 \leq i \leq r$. Then Figure 4 shows an unrestricted SLD-Refutation of $A$ from $LP(P)$. It follows, by the MGU lemma for SLD-Refutations that there is an SLD-refutation of $A$ from $LP(P)$ using the same clauses but different substitutions. The result now follows immediately.

EXAMPLE 12 Suppose $\mathcal{T}$ is the set $[0,1] \times [0,1]$. $\mathcal{T}$ is a complete lattice under the ordering: $(x_1, y_1) \leq (x_2, y_2)$ iff $x_1 \leq x_2$ and $y_1 \leq y_2$. Let $P$ be the program:

$$p(X) : [0.2, 0.3] \leftarrow q(X, Y) : [0.3, 0.5]$$

$$q(a, b) : [0.7, 0.2] \leftarrow$$

$$q(b, b) : [0.4, 0.6] \leftarrow$$

Then $T_P \uparrow \omega \models p(b) : [0.2, 0.3]$ and Figure 8 shows an SLD-refutation of $p(b)$ from the pure logic program $LP(P)$, and the association of truth values with the nodes in SLD-tree.

A similar technique of associating truth values with the queries in an SLD-Refutation from $LP(P)$ is possible when $P$ is an ACLP. So suppose $\mathcal{T}$ is a complete lattice under the ordering $\leq$ and $\delta : \mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}$ is a binary function that is monotone, strict and compact. We can now develop an algorithm for associating truth values with the queries in an SLD-Refutation.

**ACLP Query Processing Algorithm**

**INPUT:** an ACLP $P$ and some variable-free atom $A \in B_P$

1. Construct $LP(P)$

2. Apply SLD-Resolution w.r.t $LP(P)$ and $A$

3. Non-deterministically select an SLD-Refutation $< Q_0, C_0, \theta_0 >$, \ldots, $< Q_n, C_n, \theta_n >$ of $Q_n \equiv A$ from $P$. 


4. Associate truth values $\nu(A_j, Q_i)$ with each atom $A_j$ occurring in $Q_i$, all $1 \leq i \leq n$ as follows:

(a) Let $A$ be the solitary atom in $Q_n$, and let $C_n \in LP(P)$ be of the form $A' \leftarrow$ such that $A' \theta_n = A \theta_n$. Then $\nu(A, Q_n) = \cup \{ \mu | \mu : A' \leftarrow \text{ is a clause in } P \}$.

(b) Suppose $i < n$. Let $A_j$ be an atom in $Q_i$. If $A_j$ is not the atom on which $Q_i$ and $C_i$ resolve, then $\nu(A_j, Q_i) = \nu(A_j, Q_{i+1})$. Otherwise, let $\mathcal{F} = \{ \mu | \mu : C \text{ is a c-clause in } P \}$ such that $C$ is identical to $C_i$. Let $\nu(A_j, Q_i) = \cup \{ \delta(r, \mu) | r = \min \{ \nu(B, \theta_1, Q_{i+1}) | B \text{ is an atom in the body of } C \text{ and } \mu \in \mathcal{F} \} \}$. 

Figure 7

Figure 8
End Algorithm

Before The Soundness and Completeness Theorems for this algorithm are proved, we need a few definitions.

DEFINITION 22 Suppose $P$ is supported. Then, for each $A$ in $B_P$, there is a smallest integer, denoted $dl(A)$ such that $S_P \uparrow \omega(A) = S_P \uparrow dl(A)(A)$.

Associated with any ACLP $P$ is a dependency graphy on the Herbrand Base $B_P$ of $P$.

DEFINITION 23 Let $P$ be a pure ACLP, and $A, A' \in B_P$. A is said to refer to $A'$ iff there is a rule in $P$ having a ground instance whose head is $A$ and that contains $A'$ in the body. The relation depends on is the transitive closure of the refers to relation.

DEFINITION 24 Suppose $P$ is an ACLP, $A \in B_P$ and $\alpha \neq \bot$. $P$ is said to be stably descending iff for all $A' \in B_P$ such that $A'$ depends on $A$, the downward limit of $A$ is less than the downward limit of $A'$.

THEOREM 8 (Soundness) Suppose $P$ is a supported ACLP, $\delta$ a strict, monotone combination function, and $A$ an atom such that $\nu(Q_0, A) = \mu \neq \bot$. Then $S_P \uparrow \omega(A) \geq \mu$.

(Completeness) Suppose $P$ is a stably descending supported ACLP such that $T_P \uparrow \omega(A) = \mu \neq \bot$ and $\delta$ is a strict, compact monotone combination function. Then there is a selection of a refutation $R$ in step 3 of the ACLP Query Processing Algorithm such that $\nu(Q_0, A) = \mu$.

PROOF. (Soundness) As $\mu \neq \bot$, we know that there is at least one SLD-Refutation of $A$ from $LP(P)$. Hence, we proceed by induction on the length of the refutation selected in Step 3.
Base Case. \((n = 1)\) In this case, as \(\delta\) is strict, there is a unit clause in \(LP(P)\) of the form \(A' \leftarrow\) such that \(A'\theta = A\) for some substitution \(\nu\). \(\nu(Q_0, A)\) by Step 4(b) of the ACLP Query Processing Algorithm, is equal to \(\sqcup\{\mu \mid \mu : A \rightarrow \text{is a ground instance of a clause in } P\}\). Clearly, as \(S_P \uparrow \omega\) is a model of \(P, S_P \uparrow \omega \geq \sqcup\{\mu \mid \mu : A \rightarrow \text{is a ground instance of a clause in } P\}\).

Inductive Case. \((n + 1)\) Let the SLD-Refutation of \(Q_n\) from \(LP(P)\) be:
\[
\langle Q_0, C_0, \theta_0 \rangle, \ldots, \langle Q_{n+1}, C_{n+1}, \theta_{n+1} \rangle.
\]
Then
\[
\langle Q_1, C_1, \theta_1 \rangle, \ldots, \langle Q_{n+1}, C_{n+1}, \theta_{n+1} \rangle.
\]
is an SLD-Refutation of \(Q_1\) from \(LP(P)\) and let \(S_P \uparrow \omega(Q_1\Theta) \geq \nu(Q_1\Theta)\). (Here \(\Theta = \theta_1 \cdots \theta_n, \theta_{n+1}\).) Let \(\nu(Q_1\Theta) = \kappa\). Then, by the definition of the ACLP Query Processing Algorithm, the truth value \(\mu\) associated with \(Q_1 \equiv A\) is \(\sqcup\{\delta(\kappa, \mu_i) \mid \mu_i : A' \leftarrow Body\) is a clause in \(P\) such that \(A'\theta_0 = A\) and \(Body\theta_0 = Q_1\Theta\}\). This hub is over a finite and non-empty set. As \(S_P \uparrow \omega\) is a model of \(P\), it is immediately apparent that \(S_P \uparrow \omega(A) \geq \sqcup\{\delta(\kappa, \mu_i) \mid \mu_i : A' \leftarrow Body\) is a clause in \(P\) such that \(A'\theta_0 = A\) and \(Body\theta_0 = Q_1\Theta\}\) is \(\mu\).

This completes the soundness proof.

(Completeness) As \(P\) is supported, there is a least integer \(k\) such that
\[
\mu = S_P \uparrow k(A) = S_P \uparrow (k + 1)(A) = \cdots
\]
We proceed by induction on \(k\).

Base Case. \((k = 1)\) Then, as \(\delta\) is strict, there is a unit clause \(C \equiv \kappa : A' \leftarrow\)
in \(P\) such that \(A = A'\theta\) for some substitution \(\theta\) and \(\kappa \geq \mu\). Thus
\[
\langle A, C, \theta \rangle
\]
is an SLD-refutation of \(A\), and the truth value associated with \(A\) by the ACLP Query Processing Algorithm is \(\kappa\).
Inductive Case. \((k + 1)\) As \(P\) is supported, there is a clause \(C\) in \(P\) having a ground instance of the form

\[
\kappa: A \leftarrow B_1 \& \cdots \& B_m
\]

such that

\[
\mu = \delta(S_P \uparrow (k - 1)(B_1 \& \cdots \& B_m), k).
\]

\[
p(a) \quad p(X) \leftarrow q(X) \& r(X,Y)
\]

\[
q(a) \& r(a,Y) \quad q(a)
\]

\[
r(a,Y) \quad r(a,b)
\]

\[
\]

Figure 9

By the induction hypothesis, \(Q \equiv (B_1 \& \cdots \& B_m)\) has a refutation \(\mathcal{R}\) from \(P\) such that the truth value \(\xi\) associated with \(Q\) is greater than or equal to \(S_P \uparrow (k - 1)(B_1 \& \cdots \& B_m)\) \([\text{which is equal to } S_P \uparrow \omega(B_1 \& \cdots \& B_m) \text{ by the stable descent property}]\). Clearly,

\[
< A, C, \emptyset >, \mathcal{R}
\]

is a refutation of \(A\) with all the desired properties.
EXAMPLE 13 (Van Emden's Quantitative Rule Sets) As explained earlier, in Van Emden's version of quantitative logic programming (cf. [16]), the combination function \( \delta \) is taken to be multiplication. Thus, if we take the program \( P \) to be:

\[
\begin{align*}
0.3: & \quad p(X) \leftarrow q(X) \land r(X, Y) \\
0.7: & \quad q(a) \leftarrow \\
0.5: & \quad q(b) \leftarrow \\
0.6: & \quad r(a, b) \leftarrow \\
0.8: & \quad r(b, b) \leftarrow \\
0.4: & \quad r(a, a) \leftarrow 
\end{align*}
\]

Then \( T_P \cdot \omega(p(a)) = 0.18 \). There are two SLD-refutations of the query \( p(a) \) from \( LP(P) \). One of these is shown in Figure 9.

Of these, the ACLP Query Processing Algorithm associates 0.18 with \( p(a) \) in Figure 9 while when the ACLP Query Processing algorithm selects a different SLD-refutations tree in Step 3 (see Figure 9), it associates 0.12.

The results on ACLPs presented in this paper are much more powerful than existing methods (i.e. Van Emden's [16]) for the following reasons:

1. Firstly, our procedure applies to ACLPs based on arbitrary complete lattices - Van Emden's works only for the truth value set [0,1].

2. Secondly, we allow any combination function (as long as it is monotone, commutative and associative). This is an improvement on Van Emden's procedure which allows only multiplication.

3. Thirdly, our soundness and completeness theorems are much stronger than Van Emden's. Van Emden's theorems work under two conditions, viz. that the AND/OR tree associated with the query is finite, and that programs be covered. Neither of these restrictions is needed in our results.
4. It is generally accepted in the logic programming community that SLD-Resolution procedures are more efficient than AND/OR tree search procedures. Thus, we believe our procedure is more efficient than Van Emden’s breadth first search procedure. But we emphasize that we do not have any complexity theoretic results to back up this belief.

4 CONCLUSIONS

To conclude, we sum up the principal results contained in this paper.

1. We have demonstrated a modification of SLD-resolution which is sound and complete w.r.t. ALPs and ACLPs. All many-valued logic programming languages proposed thus far are either ALPs or ACLPs, hence the modified version of SLD-Resolutions is very general.

2. The Completeness Theorem for ACLPs is much stronger than the only existing completeness results for ACLPs (Van Emden [16], Subrahmanian [14]) in many ways.

(a) Firstly, the theorem is more general than Van Emden’s even for the specific logic of Van Emden’s because there is no restriction requiring AND/OR trees to be finite as required by Van Emden.

(b) Secondly, it applies to many different logics based on complete lattices, and not just to logics having the unit interval \([0,1]\) as the set of truth values. This comment also applies to Subrahmanian’s framework [14].

(c) And thirdly, we allow a great deal of flexibility in selecting the function \(\delta\). We only require that \(\delta\) be monotone, commutative and associative. Van Emden [16], on the other hand, allows only multiplication.

3. From the point of view of efficiency, the algorithms have approximately the same degree of efficiency as SLD-Resolution.

4. The completeness theorems in [6] were restricted to a family of ALPs called covered ALPs. No such restriction applies here.
Acknowledgements. I am grateful to Howard Blair and Melvin Fitting for useful discussions. This work was supported jointly by U.S. Air Force Contract F30602-85-C-0008 and a Syracuse University Senate Research Grant.

References


vs@logiclab.cs.syr.edu
School of Computer & Information Science
313 Link Hall
Syracuse University
Syracuse, N.Y. 13244-1240, U.S.A.

Revised: Jan. 28, 1989