5-1990

Strong Completeness Results for Paraconsistent Logic Programming

Howard A. Blair
Syracuse University, School of Computer and Information Science, blair@top.cis.syr.edu

V. S. Subrahmanian

Follow this and additional works at: https://surface.syr.edu/eecs_techreports

Part of the Computer Sciences Commons

Recommended Citation
https://surface.syr.edu/eecs_techreports/51

This Report is brought to you for free and open access by the College of Engineering and Computer Science at SURFACE. It has been accepted for inclusion in Electrical Engineering and Computer Science - Technical Reports by an authorized administrator of SURFACE. For more information, please contact surface@syr.edu.
Strong Completeness Results for Paraconsistent Logic Programming

Howard A. Blair and V.S. Subrahmanian

May 1990
Strong Completeness Results for Paraconsistent Logic Programming *

Howard A. Blair† and V.S. Subrahmanian‡

Abstract

In [6], we introduced a means of allowing logic programs to contain negations in both the head and the body of a clause. Such programs were called generally Horn programs (GHPs, for short). The model-theoretic semantics of GHPs were defined in terms of four-valued Belnap lattices [5]. For a class of programs called well-behaved programs, an SLD-resolution like proof procedure was introduced. This procedure was proven (under certain restrictions) to be sound (for existential queries) and complete (for ground queries). In this paper, we remove the restriction that programs be well-behaved and extend our soundness and completeness results to apply to arbitrary existential queries and to arbitrary GHPs. This is the strongest possible completeness result for GHPs. The results reported here apply to the design of very large knowledge bases and in processing queries to knowledge bases that possibly contain erroneous information.

1 Introduction

In [6,7], we introduced a multi-valued logic for logic programming with clauses containing negated atoms both in the head, and the body of clauses. Since such programs may be inconsistent (in the classical two valued sense), any such proposal must be able to deal with inconsistency in a formal and coherent manner. This was done in [6,7] via the device of a four valued Belnap lattice.

From a pragmatic viewpoint, the proposal in [6,7] is important because it provides a theoretical framework for developing very large knowledge bases. For example, a very large expert system ES is typically designed by n knowledge engineers each of whom designs a program P_i (after interacting with various “domain” experts). The normal

---

*This work was supported by U.S. Air Force Contract F30602-85-C-0008.
†School of Computer & Information Science, 4-116 Center for Science and Technology, Syracuse University, Syracuse, NY 13244, U.S.A. Electronic Mail: blair@logiclab.cis.syr.edu
‡Department of Computer Science, A. V. Williams Building, University of Maryland, College Park, Maryland 20742, U.S.A. Electronic Mail: vs@mimsy.umd.edu Address correspondence to V.S. Subrahmanian.
procedure then, would be to take \( ES \) to be \( \bigcup_{i=1}^n P_i \). But experts often disagree (this is a very common occurrence in daily life), and hence the expert system \( ES \) may be inconsistent. The fact that \( ES \) is inconsistent may not emerge until \( ES \) has already been in use for some time; thus it is necessary to develop formal methods for reasoning effectively in the presence of inconsistency. This point has been made independently by Perlis[36]. The work reported in [6,7] presents a first step towards a solution to this problem.

Subsequent to the work reported in [6], Fitting showed [14] that the logic set up by us in [6] may be viewed as a natural generalization of Kripke’s Theory of Truth[26]. This provides an epistemological justification for the logic introduced in [6]. Fitting’s work [15,16] yields an elegant characterization of the model theory and epistemological aspects of many-valued logic programming, but provides no proof theory. Our aim here is to rectify the existing situation by showing how to process queries to GHPs. This task is essentially that of mechanically showing that existentially quantified conjunctions of atoms (i.e. “queries”) are logical consequences of GHPs. The soundness and completeness results are developed with respect to the notion of logical consequence derived from the Belnap-style model-theoretic semantics we introduce.

Two proof procedures were given in [6]. The first was based on constructing AND/OR trees, and it was shown to be sound and complete only for covered\(^1\) GHPs and queries whose associated AND/OR trees were finite. The other procedure was an SLD-resolution-like proof procedure called SLDnh-resolution which was proven to be sound for existential queries to a class of GHPs called well-behaved GHPs. Completeness was proved for ground queries to well-behaved GHPs.

The principal result of this paper is a modified version of SLDnh-resolution that is sound and complete for existential queries to arbitrary GHPs. This removes all restrictions imposed on the query processing algorithms developed in [6]. In the context of existential query processing, this is the strongest completeness result possible.

The organization of this paper is as follows: in Section 2, we quickly derive the main theorems and definitions of [6] in the context of programs called pseudo-GHPs. This extends similar theorems for GHPs in a straightforward way, but, this additional machinery needs to be set up for deriving our strong soundness and completeness theorems. The reader interested in full proofs of these theorems will find them in [6]. In Section 3, we define the closure of a generalized Horn program, and develop a proof procedure called SLDgh-resolution. In Section 4, we show that SLDgh-resolution is sound and complete for processing existential queries to arbitrary GHPs. Section 5 contains a few illustrative examples.

A quick word on our philosophical views about automatic theorem proving and non-classical logic programming is in order. Suppose \( L \) is a logic equipped with a suitable model theory, and \( S \) is a sentence that is a logical consequence of a theory \( T \)

---

\(^1\)A gh-clause is covered if all variable symbols that occur in its body also occur in its head. A GHP is covered if all gh-clauses occurring in it are covered. The definition of gh-clause and GHP appear later on in the paper.
in $L$ (i.e. $S$ is satisfied by every model of $T$). We are firmly of the belief that a “proof” that $S$ is a logical consequence of $T$ must be carried out within the logic $L$ itself. This, in our opinion, is fundamental to the notion of logic, i.e. a logic must be robust enough to do proofs within it. In keeping with this philosophy, we tend to view methods like those of Morgan [32,33] who translates formulas in one logic to formulas in another logic, with deep misgivings. Our view also appears to be shared by McRobbie and his co-workers [43,31].

2 (Pseudo) Generalized Horn Programs

We assume that the reader is familiar with the usual definitions of term, atom, literal, etc. Let $T = \{\bot, t, f, \top\}$ be the set of truth values of our logic. A partial ordering $\preceq$ is defined on $T$ as for each $x \in T$, $x \preceq x$ and $\bot \preceq x \preceq \top$. (See Fig. 1.)

Intuitively, $\bot$ is a truth value representing “unknown” as in Kleene’s three valued logic [25]. Similarly $\top$ is a truth value representing “both true and false” or “inconsistent”. This set of truth values is essentially the same as that of Belnap [5] and Visser [46].

Definition 1 If $A$ is an atom, and $\mu$ a truth value, then $A : \mu$ is an annotated atom. If $\mu$ is one of $\{t, f\}$ (resp. $\{T, t, f\}$), then $A : \mu$ is said to be well-annotated (resp. definitely annotated).

Definition 2 Suppose $A_1, \ldots, A_n$ are atoms, and $\mu_1, \ldots, \mu_n$ are annotations in $\{t, f\}$. Then

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

is a generalized Horn clause if $\mu_0 \in \{t, f\}$ and is a pseudo generalized Horn clause if $\mu_0 \in \{T, t, f\}$. (Pseudo) Generalized Horn clauses are often called (pseudo) gh-clauses for short. All variable symbols occurring in a (pseudo) Horn clause are assumed to be implicitly universally quantified.

Thus, pseudo gh-clauses differ from gh-clauses as their heads might be of the form $A : \top$. However, annotated atoms of the form $B : \top$ may not occur in the body of either a pseudo gh-clause or an ordinary gh-clause. Note that gh-clause is a pseudo gh-clause.

Definition 3 A (pseudo) generalized Horn program is a finite set of (pseudo) gh-clauses. A (pseudo) generalized Horn program is called a (pseudo) GHP, for short.

Intuitively, the annotated atom $A : t$ ($A : f$) says that “$A$ is true” (resp. “$A$ is false”). Thus, for instance, the pseudo gh-clause

$$bird(X) : f \Leftarrow donkey(X) : t$$
says that if it is true that \( X \) is a donkey, then it is false that \( X \) is a bird.

In [6,7], we investigated the properties of GHPs – pseudo GHPs were not considered there as it was thought to be unreasonable for programmers to write sentences of the form: “If \( A_1 : \mu_1, \ldots, A_n : \mu_n \) all hold, then \( A_0 \) must be inconsistent.” This is still our point of view, and we will continue to insist that GHPs are the only programs that programmers ought to be allowed to write. We will, however, show how to associate a pseudo GHP with any given GHP, and then use the pseudo GHP to answer queries to the GHP with which it is associated. This motivates the (brief) study of the declarative semantics of pseudo GHPs. The proofs of the theorems on the declarative semantics of pseudo GHPs are omitted as they are almost identical to the proofs of the corresponding theorems for GHPs for which the reader may consult the already published [6]. As usual, we will only consider Herbrand interpretations, and throughout this paper, the words “interpretation” and “model” will be used to denote Herbrand interpretation and Herbrand model respectively. We will consider interpretations to be mappings of the Herbrand Base \( B_G \) of the (pseudo) GHP \( G \) to the set \( T \) of truth values.

**Definition 4 (Satisfaction)** Let \( I \) is an interpretation and \( F \) a formula. Then \( I \models F \) iff \( I \models (\forall)F \). Now, for any closed (i.e. containing no freely occurring variable symbols) formula \( F \):

1. \( I \models (\forall)F \) iff \( I \models F' \) for every ground instance \( F' \) of \( F \)
2. \( I \models (\exists)F \) iff \( I \models F' \) for some ground instance \( F' \) of \( F \)
3. \( I \models F_1 \land F_2 \) iff \( I \models F_1 \) and \( I \models F_2 \)
4. \( I \models F_1 \lor F_2 \) iff \( I \models F_1 \) or \( I \models F_2 \)
5. \( I \models F_1 \leftrightarrow F_2 \) iff \( I \models F_1 \) or \( I \not\models F_2 \)
6. \( I \models F_1 \leftrightarrow F_2 \) iff \( I \models F_1 \leftrightarrow F_2 \) and \( I \models F_2 \leftrightarrow F_1 \)
7. \( I \models A : \mu \) iff \( \mu \preceq I(A) \) where \( A \) is a ground atom

In cases (4)-(7) above, \( F_1 \land F_2, F_1 \lor F_2, F_1 \leftrightarrow F_2 \) and \( F_1 \leftrightarrow F_2 \) are all assumed to be closed formulas, i.e. they contain no free occurrences of any variable.

The above definition of satisfaction tells us what the models of any pseudo GHP \( G \) are. We now define an operator \( T_G \) from Herbrand interpretations to Herbrand interpretations as follows:

**Definition 5** Let \( I \) be any interpretation, \( G \) any pseudo GHP, and \( A \in B_G \). Then \( T_G(I)(A) = \cup \{ \mu \mid A : \mu \models B_1 : \mu_1 \land \ldots \land B_n : \mu_n \text{ is a ground instance of a pseudo GCL-clause in } G \text{ and } I \models B_1 : \mu_1 \land \ldots \land B_n : \mu_n \} \).
The ordering $\leq$ on truthvalues is extended, in a pointwise fashion, to interpretations, i.e. $I_1 \leq I_2$ iff for all $A \in B_G$, $I_1(A) \leq I_2(A)$.

**Theorem 1** For any pseudo GHP $G$, $T_G$ is continuous. □

**Theorem 2** $I$ is a model of the pseudo GHP $G$ iff $T_G(I) \leq I$. □

The proofs of the above theorems are very similar in spirit to the proofs of the corresponding theorems for GHPs. The interested reader may consult [6] for the proofs.

**Definition 6** The transfinite upward and downward iterations of $T_G$ are defined as:

\[
\begin{align*}
T_G \uparrow 0 &= \Delta \\
T_G \uparrow \alpha &= T_G(T_G \uparrow (\alpha - 1)) \\
T_G \uparrow \lambda &= \bigcup_{\eta<\lambda} T_G \uparrow \eta
\end{align*}
\]

\[
\begin{align*}
T_G \downarrow 0 &= \nabla \\
T_G \downarrow \alpha &= T_G(T_G \downarrow (\alpha - 1)) \\
T_G \downarrow \lambda &= \bigcap_{\eta<\lambda} T_G \downarrow \eta
\end{align*}
\]

where $\Delta$ and $\nabla$ are the interpretations that assign $\bot$ and $\top$ respectively to all $A \in B_G$, $\alpha$ is a successor ordinal, and $\lambda$ is a limit ordinal.

We use the symbol $\vdash$ to denote logical consequence, i.e. if $G$ is a pseudo GHP and $F$ a formula, then $G \vdash F$ iff for every model $M$ of $G$, it is the case that $M \models F$. The next theorem follows from the previous two theorems.

**Theorem 3** Suppose $G$ is a pseudo GHP. Then $T_G \uparrow \omega = lfp(T_G)$ where $lfp(T_G)$ denotes the least fixed point of $T_G$. Moreover, for all $A \in B_G$ and all $\mu \in T$, $G \vdash A : \mu$ iff $T_G \uparrow \omega(A) \geq \mu$. □

The proofs of the above theorems are very similar in spirit to the proofs of the corresponding theorems for GHPs. The interested reader may consult [6] for the proofs.

**Definition 7** Suppose $G$ is a pseudo GHP, and $I$ a is a model of $G$. $I$ is said to be weakly supported iff for all $A \in B_G$ such that $I(A) = \mu \neq \bot$, one of the following conditions holds:

1. There is a pseudo gh-clause in $G$ having a ground instance of the form $A : \mu \leftarrow B_1 : \mu_1 \& \ldots \& B_n : \mu_n$ such that $I \models B_1 : \mu_1 \& \ldots \& B_n : \mu_n$. (This case may occur if $\mu$ is any of $\top, t, f$.

2. (This case may occur only if $\mu = \top$.) There are two pseudo gh-clauses in $G$ having ground instances of the form

\[
\begin{align*}
A : t &\leftarrow B_1 : \mu_1 \& \ldots \& B_n : \mu_n \\
A : f &\leftarrow D_1 : \rho_1 \& \ldots \& D_m : \rho_m
\end{align*}
\]

such that $I \models B_1 : \mu_1 \& \ldots \& B_n : \mu_n$ and $I \models D_1 : \rho_1 \& \ldots \& D_m : \rho_m$. 5
Theorem 4 $I$ is a weakly supported model of the pseudo GHP $G$ iff $T_G(I) = I$. \hfill \Box

The following definition applies only to non pseudo GHPs.

**Definition 8** Suppose $I$ is a model of the GHP $G$. Then $I$ is said to be *strongly supported* iff for all $A \in B_G$, $I(A) = \mu \neq \bot$ implies that there is a gh-clause in $G$ having a ground instance of the form

$$A : \mu \leftarrow B_1 : \mu_1 \land \ldots \land B_k : \mu_k$$

**Definition 9** An interpretation $I$ is said to be *consistent*\(^2\) iff for all $A \in B_G$, $I(A) \neq \top$.

**Theorem 5** Suppose $G$ is a GHP. Then $I$ is a strongly supported model of $G$ iff $I$ is a consistent fixed-point of $T_G$. \hfill \Box

As every GHP is, by definition, also a pseudo GHP, theorem 4 applies to GHPs, and hence yields a model-theoretic characterization of the fixed-points of $T_G$ is obtained. Strongly supported models were first discussed by Apt, Blair and Walker [3]. It has been argued, in the context of classical logic programming, by Apt and Blair [2] that supported models are important because they yield explanations, based on the program, for why certain atoms are true.

### 3 Closed Pseudo GHPs

One of the crucial properties of classical logic programming is that if $A$ is true in the least model $M_P$ of a logic program $P$, then there is a clause in $P$ having a ground instance of the form $A \leftarrow B_1 \land \ldots \land B_n$ such that $\{B_1, \ldots, B_n\} \subseteq M_P$, i.e. $M_P \models B_1 \land \ldots \land B_n$. Unfortunately, the analog of this, in the setting of GHPs is not true.

**Example 1** The truth value assigned $p$ in the least model of the following GHP $G$ is $\top$, but there is no (pseudo) gh-clause in $P$ having a ground instance of the form $p : \top \leftarrow B_1 \land \ldots \land B_n$ such that $\text{lfp}(T_G) \models B_1 \land \ldots \land B_n$.

$$p : t \leftarrow$$

$$p : f \leftarrow$$

Closed GHPs below have the property that the class of weakly supported models coincides with the class of strongly supported models.

\(^2\)Such interpretations were originally called nice interpretations in [6]. We now feel consistent is more appropriate.
Definition 10 A (pseudo) GHP is closed iff for every pair $C_1, C_2$ (we assume that $C_1$ and $C_2$ are standardized apart, i.e. the variables are renamed so that $C_1, C_2$ share no common variables) of (pseudo) gh-clauses in $P$ of the form

$$A^1 : \mu^1 \leftarrow B_1^1 : \rho_1^1 \ldots & B_n^1 : \rho_n^1$$

$$A^2 : \mu^2 \leftarrow B_1^2 : \psi_1^2 \ldots & B_m^2 : \psi_m^2$$

such that $A^1, A^2$ are unifiable via mgu $\theta$ it is the case that

$$(A^1 : \cup \{\mu^1, \mu^2\} \leftarrow B_1^1 : \rho_1^1 \ldots & B_n^1 : \rho_n^1 \& B_1^2 : \psi_1^2 \ldots & B_m^2 : \psi_m^2)\theta$$

is a (possibly pseudo) gh-clause of $G$. The closure of a (pseudo) GHP $G$, denoted $CL(G)$, is the closed (possibly pseudo) GHP obtained by repeatedly adding to $G$ all clauses $C$ obtained from gh-clauses $C_1, C_2$ whose heads are unifiable.

As we will see below, the (possibly, pseudo) gh-clause (3) above is a logical consequence of gh-clauses (1) and (2). Closing a program $G$ adds certain logical consequences of $G$ as explicit gh-clauses.

Example 2 Consider the GHP $G$ of the preceding example. Then

$CL(G) = G \cup \{p : \top \leftarrow\}$

Note that even though $G$ is a GHP, $CL(G)$ is a pseudo GHP.

Lemma 1 Suppose $G$ is a (possibly pseudo) GHP, $I$ a weakly supported model of $G$ and $A \in B_G$ is such that $I(A) = \mu \neq \bot$. Then there is a pseudo gh-clause in $CL(G)$ having a ground instance of the form

$$A : \mu \leftarrow B_1 : \mu_1 \& \ldots & B_n : \mu_n$$

such that $I \models B_1 : \mu_1 \& \ldots & B_n : \mu_n$

Proof. Suppose $I(A) = \mu \neq \bot$, where $I$ is weakly supported. Either condition (1) or condition (2) in the definition of weakly supported models must hold with respect to (w.r.t.) $A$. If condition 1 holds, then the lemma is trivially true; so assume condition 2 holds. Then $I(A) = \top$ and there exist pseudo gh-clauses $C_1, C_2$ that are variant (cf. Lloyd[28, p.19]) of gh-clauses in $G$ (standardized apart) where

$$C_1 \equiv \quad A' : t \leftarrow E_1 : \mu_1 \& \ldots & E_k : \mu_k$$

$$C_2 \equiv \quad A'' : f \leftarrow D_1 : \psi_1 \& \ldots & D_m : \psi_m$$

such that for some ground instances $C_1 \sigma_1$ and $C_2 \sigma_2$ of $C_1, C_2$ respectively, (where $\sigma_i$ affects all and only the variable symbols of $C_i$), such that $I \models (E_1 : \mu_1 \& \ldots & E_k : \mu_k)\sigma_1$ and $I \models (D_1 : \psi_1 \& \ldots & D_m : \psi_m)\sigma_2$. Thus, $A'$ and $A''$ are unifiable (as they have a common instance $A$). Let $\theta$ be the mgu of $A'$ and $A''$. Then the clause $C$ given by

$$C \equiv \quad (A' : \top \leftarrow E_1 : \mu_1 \& \ldots & E_k : \mu_k \& D_1 : \psi_1 \& \ldots & D_m : \psi_m)\theta$$

7
is a pseudo gh-clause in \( CL(G) \). Since \( A \) is a common ground instance of \( A' \) and \( A'' \), and \( C_1 \) and \( C_2 \) may without loss of generality be presumed to be standardized apart, there is a ground instance of \( C \) with head \( A : \top \) whose body is satisfied by \( I \). \( \square \)

**Theorem 6**

1. \( I \) is a model of the pseudo GHP \( G \) iff \( I \) is a model of \( CL(G) \), i.e. \( G \) and \( CL(G) \) are logically equivalent.

2. Moreover, \( T_G = T_{CL(G)} \).

**Proof.** (1) As \( CL(G) \supseteq G \), every model of \( CL(G) \) is a model of \( G \). To show that every model of \( G \) is a model of \( CL(G) \), it suffices to show that if \( I \) is a model of \( G \), then \( I \) is a model of \( CL(G) - G \). Suppose

\[
A : \mu \leftarrow B_1 : \rho_1 \& \ldots \& B_k : \rho_k \& D_1 : \psi_1 \& \ldots \& D_m : \psi_m
\]

is a ground instance of a pseudo gh-clause in \( CL(G) - G \) and \( I \models B_1 : \rho_1 \& \ldots \& B_k : \rho_k \& D_1 : \psi_1 \& \ldots \& D_m : \psi_m \). The above pseudo gh-clause must have been derived from two pseudo gh-clauses in \( G \) having ground instance of the form

\[
A : \mu_1 \leftarrow B_1 : \rho_1 \& \ldots \& B_k : \rho_k
\]

\[
A : \mu_2 \leftarrow D_1 : \psi_1 \& \ldots \& D_m : \psi_m
\]

where \( \mu = \cup \{\mu_1, \mu_2\} \). As \( I \) is a model of \( G \), and as \( I \) satisfies the body of each of the above two ground pseudo gh-clauses, it must be the case that \( I \models A : \mu_1 \) and \( I \models A : \mu_2 \), i.e. \( I(A) \geq \mu_1 \) and \( I(A) \geq \mu_2 \), i.e. \( I(A) \geq \cup \{\mu_1, \mu_2\} = \mu \). This completes the proof.

(2) Since \( G \subseteq CL(G) \), it is immediately apparent that \( T_G \preceq T_{CL(G)} \). Suppose \( I \) is an interpretation and \( A \in B_G \) is such that \( T_{CL(G)}(I)(A) = \mu \). We need to show that \( T_G(I)(A) \geq \mu \).

**Case 1.** If \( \mu = \bot \), then \( T_{CL(G)}(I)(A) \preceq T_G(I)(A) \) trivially.

**Case 2.** If \( \mu \in \{t, f\} \), then there is a pseudo gh-clause in \( CL(G) \) which is in \( G \) such that

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

and \( I \models B_1 : \mu_1 \& \ldots \& B_k : \mu_k \).

**Case 3.** \( \mu = \top \). Then there is a pseudo gh-clause in \( CL(G) \) such that

\[
A : \top \leftarrow B_1 : \mu_1 \& \ldots \& B_k : \mu_k \& D_1 : \rho_1 \& \ldots \& D_m : \rho_m
\]

and \( I \models B_1 : \mu_1 \& \ldots \& B_k : \mu_k \& D_1 : \rho_1 \& \ldots \& D_m : \rho_m \). Then there are two pseudo gh-clauses in \( G \) having ground instances of the form

\[
A : t \leftarrow B_1 : \mu_1 \& \ldots \& B_k : \mu_k
\]

\[
A : f \leftarrow D_1 : \rho_1 \& \ldots \& D_m : \rho_m
\]
It follows that $T_G(I)(A) \geq \bigcup \{ t, f \}$, i.e. $T_G(I)(A) = \top$.

This completes the proof. \hfill \square

The above theorem establishes that a pseudo GHP and its closure have the same declarative meaning, and moreover that they are equivalent in a computational sense as $T_G = T_{CL(G)}$. The requirement, in the definition of closure of pseudo GHPs, that the pseudo gh-clauses $C_1, C_2$ be standardized apart is necessary for Lemma 1 to hold, as is evident from the following example.

**Example 3** Consider the GHP $G$ below:

\begin{align*}
p : t & \leftarrow q(X) : t \\
p : f & \leftarrow r(X) : t \\
q(a) : t & \leftarrow \\
r(b) : t & \leftarrow
\end{align*}

Now, $T_G \uparrow \omega(p) = \top$. If we construct the closure of $G$ without standardizing apart the first 2 gh-clauses above, then $CL(G)$ would be:

\[ G \cup \{ p : t \leftarrow q(X) & r(X) \} \]

Note that $T_{CL(G)} \uparrow \omega(p) = \top$, but there is no clause in $CL(G)$ of the form

\[ p : \top \leftarrow B_1 : \mu_1 & \ldots & B_n : \mu_n \]

such that the weakly supported model $T_{CL(G)} \uparrow \omega \models B_1 : \mu_1 & \ldots & B_n : \mu_n$. The only pseudo gh-clause in $CL(G)$ having $p : \top$ as the head is the pseudo gh-clause added to $G$ above, but $T_{CL(G)} \uparrow \omega$ does not satisfy any of the two instances of the body obtained by instantiating $X$ to $a, b$ respectively. This example shows that pseudo gh-clauses must be standardized apart for Lemma 1 to hold.

### 4 Query Processing Procedure

**Definition 11** A query $Q$ is an existentially closed conjunction of annotated atoms.

In logic programming, the notion of query is usually restricted to that of existential queries. We do not claim that this is the only kind of question one may wish to ask of a database. We observe that given a formula of the form $(\forall x)F$ where the variable symbol $x$ has a free occurrence in $F$, the problem: “Is $(\forall x)F$ true in all Herbrand models of $P$?” is a $\Pi_2^n$-complete problem. There is a strong feeling, both in logic programming, and to some extent in AI also, (cf. Shoham [37]), that all the individuals that affect our domain of discourse are known to exist. This appears to justify the consideration of Herbrand models. As solutions to $\Pi_2^n$-complete problems
are not effectively implementable on a computer, we do not address such queries. As far as our semantics is concerned, given a ground implication:

\[ A_0 : \mu_0 \leftarrow (A_1 : \mu_1 \& \ldots \& A_n : \mu_n) \]

where each \( A_i : \mu_i \) is an annotated atom, we can answer such a query (w.r.t. GHP \( G \)) by simply checking, using our SLDgh-procedure described below, whether \( A_0 : \mu_0 \) is a logical consequence of the expanded GHP

\[ G \cup \{A_1 : \mu_1, \ldots, A_n : \mu_n\}. \]

In order to consider disjunctive queries, we need to use two mechanisms that are known to be highly expensive, viz. factoring and ancestry resolution. As GHPs are intended to yield a viable programming language, one needs to make a trade-off between efficiency and ease and implementation on one hand, and expressive power on the other. Therefore we do not consider disjunctive queries in this paper.

**Definition 12** Suppose \( Q \equiv (\exists)(A_1 : \mu_1 \& \ldots \& A_n : \mu_n) \) is a query and \( C \equiv A : \rho \leftarrow B_1 : \psi_1 \& \ldots \& B_m : \psi_m \)

is a pseudo gh-clause such that \( C \) and \( Q \) are standardized apart and such that for some \( A_i, 1 \leq i \leq n \) and \( A \) are unifiable via mgu \( \theta \) and \( \rho \supseteq \mu_i \). Then the SLDgh-resolvent of \( Q \) and \( C \) on \( A_i \) is the query

\[ (\exists)(A_1 : \mu_1 \& \ldots \& A_{i-1} : \mu_{i-1} \& B_1 : \psi_1 \& \ldots \& B_m : \psi_m \& A_{i+1} : \mu_{i+1} \& \ldots \& A_n : \mu_n) \theta \]

\( A_i : \mu_i \) is called the annotated literal resolved upon.

**Example 4** Consider the query \( Q \equiv (\exists X, Y)(p(X) : t \& q(X, Y) : f) \), and the GHP \( G \) below:

\[
\begin{align*}
(C1) & \quad p(a) : t \leftarrow r(Z) : t \\
(C2) & \quad q(a, b) : f \leftarrow \\
(C3) & \quad q(a, c) : t \leftarrow
\end{align*}
\]

The SLDgh-resolvent of \( Q \) and \( C_1 \) on \( p(X) : t \) is

\[ (\exists Y, Z)(r(Z) : t \& q(a, Y) : f) \]

The SLDgh-resolvent of \( Q \) and \( C_2 \) on \( q(X, Y) : f \) is

\[ p(a) : t \]

There is no SLDgh-resolvent of \( Q \) and \( C_3 \) since \( t \not\in f \).

We now show, in the next section that SLDgh-resolution with a slight twist, is sound and complete for existential queries to GHPs and pseudo GHPs.
5 Soundness and Completeness Issues

The main aim of this section is to prove the soundness and completeness of an SLDgh-resolution based strategy for processing queries to GHPs.

Definition 13 An SLDgh-deduction from the initial query $Q_1 \equiv A_1 : \mu_1 \& \ldots \& A_n : \mu_n$ and the GHP (not pseudo GHP !) $G$ is a sequence

$\langle Q_1, C_1, \theta_1 \rangle, \ldots \langle Q_r, C_r, \theta_r \rangle, \ldots$

where

1. each $C_i$ is a pseudo gh-clause from $CL(G)$ whose variable symbols are renamed so that $C_i$ contains no variable symbols in common with any $C_j, j < i$ and $Q_j, j \leq i$

2. $Q_{i+1}$ is the SLDgh-resolvent of $Q_i$ and $C_i$ and $\theta_i$ is the mgu used in the SLDgh-resolution

Definition 14 An SLDgh-refutation of the initial query $Q_1$ is a finite sequence

$\langle Q_1, C_1, \theta_1 \rangle, \ldots \langle Q_r, C_r, \theta_r \rangle$

such that the SLDgh-resolvent of $Q_r$ and $C_r$ is the empty query.

It is important to note that the (pseudo) gh-clauses used in an SLDgh-deduction of a query $Q_1$ from a GHP $G$ come from the closure of $G$ but not necessarily from $G$ itself. (This is the "twist" alluded to earlier on in the paper.)

Theorem 7 (Soundness) If there exists an SLDgh-refutation of the initial query $Q_1$:

$A_1 : \rho_1 \& \ldots \& A_m : \rho_m$

from the GHP $G$, then

$T_G \uparrow \omega \models \exists(Q_1)$

Proof. Suppose there exists an SLDgh-refutation of the initial query $Q_1$ from $G$ of the following form:

$\langle Q_1, C_1, \theta_1 \rangle, \ldots, \langle Q_n, C_n, \theta_n \rangle$

We will show by induction on $n$ that $T_Q \uparrow \omega \models Q_1$.

Base Case: $[n = 1]$ Then the gh-resolvent of $Q_1$ and $C_1$ is the empty query, i.e. $Q_1$ is a unit conjunction (i.e. $A_1 : \rho_1$) and $C_1$ is a unit pseudo gh-clause, i.e. $C_1$ is of the form $A' : \beta \leftarrow$
where $\beta \geq \rho_1$ and $A_1\theta_1 = A'\theta_1$. As $T_G \uparrow \omega$ is a model of the GHP $G$ and hence of $CL(G)$, it must be a model of $C_1$, and so it must be true that for every ground instance $A_2$ of $A'$, $T_G \uparrow \omega(A_2) \geq \beta$. In particular, for every $A_3$ that is a ground instance of $A_1\theta_1$, $T_G \uparrow \omega(A_3) \geq \beta \geq \rho_1$, hence $T_G \uparrow \omega \models Q_1$.

**Induction Case:** Suppose

$$(Q_1, C_1, \theta_1), \ldots, (Q_n, C_n, \theta_n), (Q_{n+1}, C_{n+1}, \theta_{n+1})$$

is an SLDgh-refutation of $Q_1$. Then

$$(Q_2, C_2, \theta_2), \ldots, (Q_{n+1}, C_{n+1}, \theta_{n+1})$$

is an SLDgh-refutation of $Q_2$. Therefore, by the induction hypothesis, $T_G \uparrow \omega \models Q_1$.

But $Q_2$ is the gh-resolvent of $Q_1$ and $C_1$. So $Q_2$ has the form:

$$(A_1 : \rho_1 \& \ldots \& A_{i-1} : \rho_{i-1} \& E_1 : \psi_1 \& \ldots \& E_r : \psi_r, \& A_{i+1} : \rho_{i+1} \& \ldots \& A_m : \rho_m)\theta_1$$

where $C_1$ is the pseudo gh-clause

$$H : \delta \Leftarrow E_1 : \psi_1 \& \ldots \& E_r : \psi_r$$

such that $H\theta_1 = A_i\theta_1$ and $\delta \geq \rho_i$ (by definition of gh-resolution). Because $T_G \uparrow \omega \models Q_2$ it follows that

$$T_G \uparrow \omega \models (E_1 : \psi_1 \& \ldots \& E_r : \psi_r)\theta_1$$

(as this is a sub-conjunct of $Q_2$). As $T_G \uparrow \omega$ is a model of $G$ and hence of $CL(G)$, it must satisfy every pseudo gh-clause of $CL(G)$ (and every renamed version of any pseudo gh-clause in $CL(G)$). In particular, $T_G \uparrow \omega$ must satisfy $C_1\theta_1$. $T_G \uparrow \omega$ satisfies the body of $C_1\theta_1$, so it must satisfy $H\theta_1 : \delta$. But $H\theta_1 = A_i\theta_1$; so $T_G \uparrow \omega \models A_i\theta_1 : \delta$. Therefore, since $\delta \geq \rho_i$, $T_G \uparrow \omega \models A_i\theta_1 : \rho_i$. Thus,

$$T_G \uparrow \omega \models (\exists)((A_1 : \rho_1 \& \ldots \& A_m : \rho_m)\theta_1)$$

\[\square\]

**Definition 15** An unrestricted SLDgh-refutation of the query $Q_1$ from the pseudo GHP $G$ is an SLDgh-refutation of $Q_1$ from $G$ except that the requirement that the $\theta_i$'s be most general unifiers is dropped – the $\theta_i$'s need only be unifiers.

**Lemma 2 (Mgu Lemma)** Let $G$ be a GHP and $Q_1$ a query. Suppose $Q_1$ has an unrestricted SLDgh-refutation from $G$. Then $Q_1$ has an SLDgh-refutation from $G$ of the same length. If $\theta_1, \ldots, \theta_n$ are the substitutions from the unrestricted refutation, and $\theta'_1, \ldots, \theta'_n$ are the substitutions from the SLDgh-refutation, then there is a substitution $\gamma$ such that $\theta_1 \cdots \theta_n = \theta'_1 \cdots \theta'_n\gamma$.

**Proof.** By induction on the length of the unrestricted refutation. \[\square\]
Lemma 3 (Lifting Lemma) Suppose $G$ is a GHP, $Q_1$ a query and $\theta$ a substitution. Suppose there is an SLDgh-refutation of $Q_1\theta$. Then there is an SLDgh-refutation of $Q_1$ from $G$ of the same length. Moreover, if $\theta_1, \ldots, \theta_n$ are the mgu’s from the SLDgh-refutation of $Q_1\theta$ and $\theta'_1, \ldots, \theta'_n$ are the mgu’s from the SLDgh-refutation of $Q_1$, then there is a substitution $\gamma$ such that $\theta_1 \cdots \theta_n = \theta'_1 \cdots \theta'_n \gamma$.

Theorem 8 (Completeness) Suppose $G$ is any GHP. Then if $Q_1 \equiv (\exists)(A_1 : \rho_1 \ldots \& A_m : \rho_m)$ is a query that is satisfied by $T_G \uparrow \omega$, then there is an SLDgh-refutation of $Q_1$ from $G$.

Proof. If $T_G \uparrow \omega \models (\exists)(A_1 : \rho_1 \ldots \& A_m : \rho_m)$, then $T_G \uparrow \omega \models (A_1 : \rho_1 \ldots \& A_m : \rho_m)\theta$ where $\theta$ is a substitution that makes $Q_1\theta = (A_1 : \rho_1 \ldots \& A_m : \rho_m)$ ground. Thus, there is an integer $n$ such that $T_G \uparrow n \models (A_1 : \rho_1 \ldots \& A_m : \rho_m)\theta$. The proof proceeds by induction on $n$.

Base Case. $(n = 1)$ Then for each $A_i \theta : \rho_i$ there is a unit gh-clause in $CL(G)$ having a ground instance $C_i$ such that $C_i \equiv A_i \theta : \mu'_i$ (where $\mu'_i \supseteq \rho_i$). $CL(G)$. Figure 2 then shows an unrestricted SLDgh-refutation path for the query $Q_1\theta$. By the Mgu Lemma, there then exists an SLDgh-refutation of $Q_1\theta$ from $G$. By the Lifting Lemma, there is an SLDgh-refutation of $Q_1$ from $G$.

Inductive Case. $(n + 1)$ Suppose $T_G \uparrow (n + 1) \models Q_1\theta$. Then, for each $A_i \theta : \rho_i$, either

1. $T_G \uparrow n \models A_i \theta : \rho_i$ in which case, by the inductive hypothesis, there is an SLDgh-refutation of $A_i \theta : \rho_i$ from $G$, or

2. Case (1) does not occur, in which case, there is a gh-clause in $CL(G)$ having a ground instance of the form

$$A_i \theta : \mu \leftarrow B_1 : \psi_1 \ldots \& B_k : \psi_k$$

such that $\mu \supseteq \rho_i$ and $T_G \uparrow n \models B_1 : \psi_1 \ldots \& B_k : \psi_k$. By the induction hypothesis, the ground query $B_1 : \psi_1 \ldots \& B_k : \psi_k$ has an SLDgh-refutation $R$ from $G$; hence the following diagram (Fig. 3) shows an unrestricted SLDgh-refutation of $A_i \theta : \rho_i$ from $G$. By the Mgu Lemma, there is an SLDgh-refutation of $A_i \theta : \rho_i$ from $G$.

Thus, for each $1 \leq i \leq m$, there is an SLDgh-refutation (from $G$) $R_i$ of $A_i \theta : \rho_i$. These can be combined into an SLDgh-refutation of $Q_1\theta$ from $G$ (see Fig. 4). By the Lifting Lemma, it follows that there exists an SLDgh-refutation of $Q_1$ from $G$.

6 Applications of GHPs

In this section, we present two examples that demonstrate the application of GHPs to the development of very large knowledge bases. The first develops a small expert system for real estate investment, while the second is a “murder mystery”.
Example 5 (Land Investment) The following gh-clauses define a simple GHP for real estate investment.

\[
\text{buy}(X) : \text{t} \leftarrow \text{tourist_resort}(X) : \text{t} \quad (1)
\]

\[
\text{buy}(X) : \text{t} \leftarrow \text{has_oil}(X) : \text{t} \quad (2)
\]

\[
\text{buy}(X) : \text{f} \leftarrow \text{possible_problem}(X) : \text{t} \quad (3)
\]

\[
\text{possible_problem}(X) : \text{t} \leftarrow \text{near_nuclear_plant}(X) : \text{t} \quad (4)
\]

\[
\text{possible_problem}(X) : \text{t} \leftarrow \text{disputed_land}(X) : \text{t} \quad (5)
\]

\[
\text{tourist_resort}(\text{waikiki}) : \text{t} \leftarrow \quad (6)
\]

\[
\text{has_oil}(\text{el_haciendos}) : \text{t} \leftarrow \quad (7)
\]

\[
\text{near_nuclear_plant}(\text{el_haciendos}) : \text{t} \leftarrow \quad (8)
\]

The first two gh-clauses say that it is worth buying \( X \) if \( X \) either is a tourist resort, or has oil. The third gh-clause advocates caution—"don't buy land at a place where problems may crop up". gh-clauses (4) and (5) describe two possible problems, viz. being near a nuclear plant, and being disputed land. The last three clauses tell us that \textit{waikiki} is a tourist resort, and that \textit{el_haciendos} has oil, but is near a nuclear plant.

This GHP contains inconsistent information because, gh-clauses (2) and (7) yield \textit{buy}(\textit{el_haciendos}) : \text{t}, while (3),(4),(8) yield \textit{buy}(\textit{el_haciendos}) : \text{f}. Nonetheless, not every sentence is derivable from this GHP, e.g. \textit{buy}(\textit{waikiki}) : \text{f} is not a logical consequence of this GHP, i.e. \( T_G \upharpoonright \omega \not\models \textit{buy}(\textit{waikiki}) : \text{f} \). The purpose of this example is to illustrate two facts:

1. Inconsistencies can easily arise while building expert systems. Each of the rules contained in this GHP could arise, quite legitimately, while designing an expert system for land acquisition.

2. The framework of this paper is such that inconsistent information about \( A \) does not affect \( B \), unless \( B \) is defined in terms of \( A \). Thus, our framework, in some sense, localizes the inconsistency, and permits us to continue reasoning safely with \( B \) despite the inconsistent information about \( A \) being present.

Example 6 (Murder Mystery) This is a simple GHP for identifying suspects in a murder mystery. As anyone who has read murder mysteries knows, inconsistencies abound as people lie to protect themselves, and all that is known is a tangled web of lies. The scenario here is that Al was killed in his room at 10:30 AM. John, Ann, and Bill made the following statements:

1. John admits to being at Al's room until 10:00 AM. He has no alibi for 10:30 AM.

2. Ann claims to have been at home with Bill from 9 AM to 11 AM.
3. Bill was with Ann

The physical evidence at the murder site (Al’s room) was that:

1. Al was stabbed

2. two sets of fingerprints were found on the murder weapon; John’s, with Ann’s superimposed on his.

3. The neighbours were asleep and didn’t see anything

The information here could be summed up as follows:

\[
\begin{align*}
\text{suspect}(X) : t & \iff \text{alibi}(X) : T \\
\text{suspect}(X) : t & \iff \text{opportunity}(X) : t \& \text{motive}(X) : t \\
\text{motive}(\text{ann}) : t & \iff \\
\text{motive}(\text{john}) : t & \iff \\
\text{opportunity}(\text{john}) : t & \iff \\
\text{at(\text{al} \text{ room}, \text{john}, 10)} : t & \iff \\
\text{at(\text{al} \text{ room}, X, T)} : t & \iff \text{at(\text{al} \text{ room}, Y, T1)} : t \& \\
& \quad \text{superimposed fingerprint}(X, Y) : t \& \\
& \quad (T > T1) \\
\text{at(\text{ann} \text{ house}, \text{ann}, T)} : t & \iff 9 < T < 11 \\
\text{at(\text{ann} \text{ house}, \text{bill}, T)} : t & \iff 9 < T < 11 \\
\text{alibi}(X) : T & \iff \text{lying}(X) : t \\
\text{superimposed fingerprint}(\text{ann}, \text{john}) : t & \iff \\
\text{lying}(X) : t & \iff \text{at(Place1, X, T)} : t \& \\
& \quad \text{at(Place2, X, T)} : t \\
& \quad \text{Place1} \neq \text{Place2}
\end{align*}
\]

In addition, we assume that clauses defining the < relation are available. The above program can be used to identify two suspects. John is a suspect because he had both the motive, and the opportunity (no alibi for 10:30 AM). Ann becomes a suspect because she is lying (as her fingerprints were superimposed on those of John, who admitted being in the room at 10:00 AM; hence Ann must have been in Al’s room some time after 10:00 AM). Note that as in the case of classical logic programming [28], the ordering of gh-clauses and the arrangement of literal in the body of a clause make no difference as far as the model theoretic semantics, fixed point semantics and existence of refutations are concerned.

The purpose of the above example is to show that our model theory is robust enough to allow reasoning in the face of inconsistency (in classical logic). Note that the above GHP possesses models (in the non-classical logic). Of course, a great deal
of work remains to be done in developing a formal methodology for the construction of very large logic programs (which may possibly contain erroneous information). The actual design of such programs is a different issue from that of reasoning about such programs.

It has been suggested that the Murder mystery program can be written in Prolog itself. This scheme, to our mind is philosophically ill-founded. We have heard of no system of (first-order) logic in which the truth values can be treated as first-class objects. Any such attempt must fall within the realm of metalogic – and the semantics of metalogic programming is not at all well-understood (cf. [20], [39]). Moreover, converting GHPs by adding truth values as arguments to predicate symbols, is essentially a translational mechanism like those of Morgan [32]. When such translational mechanisms are proposed, formal theoretical relationships between the initial formalism and its translation need to be carefully studied.

7 Conclusions

Mechanical reasoning in inconsistent formal systems is of vital importance in expert systems design. This, of course, has been known for some time to both computer scientists (cf. Perlis[36]) and logicians (notably Newton da Costa [9,10,11,12] and Michael Dunn [13]).

To our knowledge, the first attempt at addressing the problem of inconsistencies in logic programming theory, was that of Fitting [14]; a concurrent effort was due to Blair and Subrahmanian[6,7], as well Subrahmanian [1,38]. Before proceeding to say any further, we emphasize, as we have done in the previous paragraph, that we are not the first to study the problem of mechanical reasoning in inconsistent formal systems. This has been done by many individuals in different settings – e.g. Girard [19] and Avron [4] study a system of relevance logic called linear logic that addresses issues in concurrent programming. Likewise Patel-Schneider [34,35] addresses the tautological entailment issue in the context of four-valued terminological logics. The inheritance net community [45,44] has also studied a relevant-like semantics for inheritance nets. But it is only recently that a declarative semantics has been proposed for inheritance nets by Kifer and his co-workers [23,24]. Both these works use the setting of annotated programs introduced in [1,38,6]. Martins and Shapiro [29] develop a theory of non-monotonic belief revisions based on relevance logic. Their system lacks a simple model theory. Their implementation, however, is impressive. In logic programming, we are also concerned with designing a viable programming language, i.e. one that can be easily implemented. (Note that an extension of the results presented here has been implemented in an experimental system called QUANTLOG, cf. Subrahmanian and Umrigar [39]. The truth values of QUANTLOG consist of the reals in the unit interval [0, 1] together with an additional value T. These truth values are ordered as shown in Figure 1 below. The special case of the QUANTLOG system obtained by allowing annotations to consist only of 0.5, 0, 1, T yields the same logic as the four-valued logic
A declarative semantics for GHPs was originally presented in [6]. Two proof procedures were presented in [6].

1. The first proof procedure was based on an AND/OR tree searching technique, and was proven to be sound for ground queries to a class of GHPs called well-covered.

2. The second proof procedure was an SLD-resolution like technique which was proven to be sound for all existential queries, but completeness was proved only for ground queries to well-behaved GHPs.

Thus, there was scope for strengthening the completeness results obtained in [6]. In fact, this problem is cited as an open problem at the end of [6]. In this paper, we have developed a procedure called SLDgh-resolution that is sound and complete for processing existential queries to arbitrary GHPs. Thus, none of the restrictions (viz. ground queries, well-coveredness, well-behavedness) are needed in our soundness and completeness results. This is the strongest possible completeness result one can obtain for the fragment of annotated logics that we are concerned with.

The framework described in this paper does not allow clauses to have disjunctive heads. Thus, for instance, if we wish to say that “Clyde is either a frog or a squirrel, but I am not sure which”, we find that we cannot express this as a GHP. In

\[ A (\text{pseudo}) \text{ gh-clause is well-covered iff there are no variables in the body of the (pseudo) gh-clause that do not occur in the head of that (pseudo) gh-clause. A (pseudo) GHP is well-covered iff each (pseudo) gh-clause in it is well-covered.} \]

\[ A \text{ GHP } G \text{ is well-behaved iff whenever } A_1 : \mu_1 \text{ and } A_2 : \mu_2 \text{ are the heads of any two gh-clauses in } G \text{ such that } A_1, A_2 \text{ are unifiable, it is the case that } \mu_1 = \mu_2. \]
logic programming too, disjunctions are not allowed to appear in the heads of clauses. (Note that the clause $p \leftarrow \neg q$ in logic programming is not operationally equivalent to $(p \lor q)$.) This is because resolution systems that are non-Horn require two expensive tests – factoring and ancestry resolution, for completeness. One of the aims of logic programming is to provide a viable (easily, and reasonably efficiently implementable) programming language. One of the trade-offs that have traditionally been made in programming language development (not just in Prolog) has been to trade-off expressive power against efficiency gains. In the framework described in this paper, we have avoided ancestry resolution and factoring in order to gain expressive power. This is why disjunctions are not allowed to appear in the head of a clause in GHPs. In addition, many fixed-point results collapse for disjunctive programs. A fixed-point semantics and a proof theory for GHP extended by allowing disjunctions to appear in clause heads is described in a recent work of Subrahmanian [41]. Unfortunately, in this extended framework, the relationship between models (resp. supported models) of a GHP $G$ and the pre-fixed-points (resp. fixed-points) of the $T_G$ operator collapses. Moreover, ancestry resolution and factoring are required, thus making query processing more inefficient. As a side remark, we emphasize that pure definite clause logic programs (not GHPs) are computationally expressive enough in the sense that all the r.e. sets are computable as the success set (cf. Lloyd [28] for a formal definition of success set) of a logic program. Thus for computational purposes, Horn clauses seem fine. (Non-r.e. sets cannot be computed anyway). Thus, there is an additional trade-off to be made between computational power and expressive power.

### 7.1 Discussion

The development of a fixed-point semantics for pure logic programs was one of the central early developments in classical logic programming. These results are clearly and elegantly described in Lloyd’s book [28]. What then, are we to expect of non-classical logic programs? What semantical properties should these programs have? Is it reasonable to expect that the models (over a fixed pre-structure, say) be characterizable in terms of the pre-fixed-points (or fixed-points) of a monotone operator that maps structures to structures?

We have no definite answer to the last two questions. Our inclination, for purely aesthetic reasons, is to say yes. Our own research has been, thus far, carried out with this goal in mind. After all, talking of pre-fixed-points make sense only when the space of structures is partially ordered. What if no such natural partial ordering exists?

The other issue surrounding non-classical logics is that of proof theory. Do complete proof procedures exist? In several cases (e.g. various systems of modal and temporal logic), the answer is negative. If a program in logic $L$ is a finite set of sentences having

---

5Fixed-point semantics for programming languages have been studied extensively by Scott et al [18]. In logic, the best known work linking fixed-point theory and logical consequence is due to Tarski [42]. Studies of the fixed-point theory associated with relevance and certain kinds of four valued logic has been conducted by Woodruff[30].

---
a certain syntactic form, then an effort must be made to ensure that the set of queries that are logical consequences of such programs is recursively enumerable. Only then is the notion of program in logic $L$ a computationally useful one.

Acknowledgements. We thank Newton da Costa, Mel Fitting, Mike Kifer and Wiktor Marek for many useful discussions.

References


