Theoretical Computer Science 68 (1989) 135–154
North-Holland

PARACONSISTENT LOGIC PROGRAMMING

Howard A. BLAIR and V.S. SUBRAHMANIAN

Logic Programming Theory Group, School of Computer & Information Science, 313 Link Hall, Syracuse University, Syracuse, NY 13244-1240, U.S.A.

Abstract. This paper makes two contributions. First, we give a semantics for sets of clauses of the syntactic form $L_0 \Leftarrow L_1 \& \cdots \& L_n$ where each L_i is a literal. We call such clauses generally Horn clauses. Any such endeavour has to give a coherent, formal treatment of inconsistency (in the sense of two-valued logic). Thus, as a second contribution, we give a *robust* semantics for generally Horn programs that allows us to "make sense" of sets of generally Horn clauses that are inconsistent (in the two-valued logic sense). This applies to the design of very large knowledge bases where inconsistent information is often present.

1. Motivation

Over the past few years, several researchers have made attempts to allow programming with sets of non-Horn clauses [20, 22]. However, there has been relative lack of success in giving model-theoretic and fixpoint semantics to such extensions. The main reason for this lack of success has been due to the fact that sets of non-Horn clauses may be inconsistent. Thus, certain programs may mean "nothing" simply because they have no models.

However, if logic programming is to be a *pragmatic* tool for the development of knowledge bases, it must have some means for dealing with inconsistent knowledge. Take for example an expert system developed by a team of logic programmers. Each programmer might have acquired information from various domain experts. It is very common for experts to disagree (often strongly). Thus, the knowledge base so developed might contain inconsistent information.

Pioneering work has been done on reasoning in the presence of inconsistent information by da Costa [10–13] and Belnap [7]. Logics of this kind are often called *paraconsistent logics.* Recently, in an interesting paper, Fitting [14] has given a declarative semantics for reasoning in the presence of inconsistent information. Under Fitting's semantics, the sentence $A \& \neg A$ would not be falsified by the interpretation that assigns the truth value $\perp$ to A (where $\perp$ is the truth value *unknown* of Kleene [17, 19]). Thus, $A \vee \neg A \Leftrightarrow A \& \neg A$ is true in any Kleene interpretation of the kind discussed by Fitting in which the truth value of A is $\perp$, i.e. *unknown.* We feel that the sentence $A \& \neg A$ should be assigned the truth value *inconsistent* (with respect to the intuition of two valued logic) rather than *unknown.* This distinction is made in detail in [7].

Perlis [21, p. 180] has argued that methods must be found for reasoning in the presence of inconsistent information. Paraconsistent [4, 10–13] logics provide what seems to be the *only* way known thus far to declaratively characterize arbitrary sets of clauses (that may or may not be inconsistent). The first paraconsistent logic programming languages were developed in [1, 9, 23, 24].

As in [24], we shall use the device of *annotated atoms* (*literals*) instead of using the negation symbol. It will be clear from Theorem 3.7 below that there is no loss of generality in doing so. In Section 2, we shall introduce the class of generally Horn programs and then investigate the semantics of programs in this class in Sections 3 and 4. A decidable subset of the class of generally Horn programs is introduced in Section 5 and it is shown that this class possesses some interesting model theoretic properties. The operational semantics of generally Horn programs is discussed in Sections 6 and 7.

2. Generally Horn logic programs: syntax

We assume the reader is familiar with the usual syntactic notions of terms, atoms and literals. Any undefined expressions used in this paper may be found in Lloyd [18].

The set $\mathcal{T} = \{\bot, \mathbf{t}, \mathbf{f}, \top\}$ is the set of *truth values* of our four-valued logic. The truth values $\bot, \mathbf{t}, \mathbf{f}, \top$ correspond, respectively, to the truth values $*$, T, F, TF of Visser [27] and **None**, **T**, **F**, **Both** of Belnap [7] and stand, respectively, for "undefined", true (in the intuitive sense of two-valued logic), false (also in the two-valued sense), and "over-defined" (which may also be thought of as inconsistent in the intuition of two-valued logic). We define an ordering $\preccurlyeq$ on $\mathcal{T}$ as follows.

Definition 2.1. $(\forall x, y \in \mathcal{T})$, $x \preccurlyeq y \Leftrightarrow x = y \vee x = \bot \vee y = \top$.

The ordering $\preccurlyeq$ is represented by the Hasse diagram given in Fig. 1. We use the notation $x \succcurlyeq y$ iff $y \preccurlyeq x$ where $x, y \in \mathcal{T}$. Also, as usual, $\prec, \succ$ are the irreflexive restrictions of $\preccurlyeq$ and $\succcurlyeq$, respectively. The set $\mathcal{T}$ is a complete lattice under this ordering.

Definition 2.2. If A is a literal, then $A : \mu$ is called an *annotated literal*, where $\mu \in \mathcal{T}$. μ is called the *annotation* of A. If μ is one of $\mathbf{t}, \mathbf{f}$, then $A : \mu$ is called a *well-annotated* literal, and μ is called a w-*annotation*.

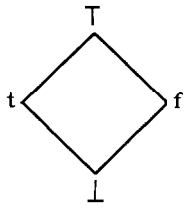

Fig. 1.

Intuitively, the annotated atom $A:\mathbf{t}$ may be read as a rough approximation to "A is *known* to be true". Similarly, the annotated atom $A:\mathbf{f}$ may be regarded as saying "A is *known* to be false". This is only meant as a rough aid to intuition, and we are not committing ourselves to any epistemic consequences of this point of view.

Definition 2.3

(1) Any annotated atom is a *formula.* (Atoms are the usual atomic formulas of first-order logic).
(2) If $A:\mu$ is an annotated atom, then $\neg A:\mu$ is a formula.
(3) If F_1, F_2 are formulas, then $F_1 \,\&\, F_2$, $F_1 \vee F_2$, $F_1 \Leftarrow F_2$, $F_1 \Leftrightarrow F_2$ are formulas.
(4) If F is a formula and x any variable symbol, then $(\forall x)F$ and $(\exists x)F$ are formulas.

Definition 2.4. If $A_0, \ldots, A_n$ are literals, and if $\mu_0, \ldots, \mu_n$ are w-annotations, then

$$A_0:\mu_0 \Leftarrow A_1:\mu_1 \,\&\, \cdots \,\&\, A_n:\mu_n$$

is called a *generalized Horn clause* (or gh-clause for short). $A_0:\mu_0$ is called the *head* of the above gh-clause, while $A_1:\mu_1 \,\&\, \cdots \,\&\, A_n:\mu_n$ is called the *body* of the above gh-clause.

The notion of a substitution is similar to that in [18]. Applying a substitution θ to an annotated literal $A:\mu$ results in the annotated literal $A\theta:\mu$. The notion of applying a substitution is extended in the obvious way to a conjunction of annotated literals, and to gh-clauses.

Definition 2.5. If $A:\mu$ and $B:\rho$ are annotated literals, then $A:\mu$ and $B:\rho$ are said to be *unifiable* (with mgu θ) iff A and B are unifiable (with mgu θ). Note that μ need not equal ρ as we are not defining the result of the unification yet.

Definition 2.6. A *generalized Horn program* (GHP) is a finite set of gh-clauses.

3. Generally Horn logic programs: semantics

In this paper, we will consider only "Herbrand-like" interpretations [18], i.e. the universe of individuals in the interpretation consists of all and only the ground terms of the language being interpreted. Unless explicitly specified otherwise, throughout the rest of this paper, the set of individuals in models and interpretations will be a Herbrand universe. We will consider an interpretation I as a function $I: B_L \to \mathcal{T}$ where B_L is the Herbrand base under consideration.

Definition 3.1. The *negation* of an annotation is defined as: $\neg(\mathbf{t}) = \mathbf{f}$, $\neg(\mathbf{f}) = \mathbf{t}$, $\neg(\bot) = \bot$, $\neg(\top) = \top$.[1]

Definition 3.2. A formula is said to be *closed* if it contains no occurrence of a free variable.

In this paper, we give a non-classical interpretation to the $\Leftarrow$ symbol, i.e. we do not treat it as classical material implication. In particular, the equivalence of $A \vee \neg B$ and $A \Leftarrow B$ does not hold. This is necessary because, in general, $A:\mathbf{t} \vee A:\mathbf{f}$ may not be a tautology. This formula asserts (in an *intuitionistic* fashion) the sentence

It is *known* that A is true OR it is *known* that A is false.

A formula is *closed* iff it contains no free occurrences of variables. We use the notation $(\forall)F$ and $(\exists)F$ to denote the universal closure and existential closure, respectively, of the formula F.

Definition 3.3 (*Satisfaction*). We write $I \models F$ to say that I satisfies F. An interpretation I

(1) satisfies the formula F iff I satisfies each of its closed instances, i.e. for each variable symbol x occurring free in F, and each variable free term t, $F(t/x)$ is satisfied by I (here $F(t/x)$ denotes the replacement of all free occurrences of x in F by t),
(2) satisfies the closed annotated atom $A:\mu$ iff $I(A) \geqslant \mu$,
(3) satisfies the closed annotated literal $\neg A:\mu$ iff it satisfies $A:\neg\mu$,
(4) satisfies the closed formula $(\exists x)F$ iff for some variable free term t, $I \models F(t/x)$,
(5) satisfies the closed formula $(\forall x)F$ iff for every variable free term t, $I \models F(t/x)$,
(6) satisfies the closed formula $F_1 \Leftarrow F_2$ iff I does not satisfy F_2 or $I \models F_1$,
(7) satisfies the closed formula $F_1 \,\&\, \cdots \,\&\, F_n$ iff $I \models F_i$ for all $i = 1, \ldots, n$,
(8) satisfies the closed formula $F_1 \vee \cdots \vee F_n$ iff $I \models F_i$ for some $1 \leqslant i \leqslant n$,
(9) satisfies the closed formula $F \Leftrightarrow G$ iff $I \models F \Leftarrow G$ and $I \models G \Leftarrow F$.

Definition 3.4. An interpretation I satisfies a generalized Horn program G if it satisfies every gh-clause $C \in G$.

The $\preccurlyeq$ ordering on truth values is extended in the obvious pointwise way to interpretations. Given a GHP G, and Herbrand interpretations I_1, I_2, we say

$$I_1 \preccurlyeq I_2 \text{ iff } (\forall A \in B_G) I_1(A) \preccurlyeq I_2(A)$$

where B_G is the Herbrand base of G. Note that the set of interpretations (over the set $\mathcal{T}$ of truthvalues) is a complete lattice under the $\preccurlyeq$ ordering.

[1] Our logic differs from that of Belnap [7] and Visser [27] with respect to our definition of negation. Their difinition makes $\neg$(**Both**) = **None** and $\neg$(**None**) = **Both**. Our definition seems to be more appropriate with regard to the *known that* intuition given to the annotated atoms.

Definition 3.5. If C is a gh-clause in G, then the result of replacing all negated literals $\neg A:\mu$ in C by $A:\neg\mu$ is called the *positive counterpart* C^{pos} of C. The GHP G^{pos} obtained by replacing each gh-clause C in the GHP G by C^{pos} is called the *positive counterpart* of G and is denoted by G^{pos}.

Lemma 3.6. *$I \models (\exists)\neg A:\mu$ iff $I \models (\exists)A:\neg\mu$. Similarly, $I \models (\forall)\neg A:\mu$ iff $I \models (\forall)A:\neg\mu$.*

Proof.

$I \models (\exists)\neg A:\mu$

iff $I \models \neg A(\vec{t}/\vec{x}):\mu$ for some tuple $\vec{t}$ of variable-free terms

iff $I \models A(\vec{t}/\vec{x}):\neg\mu$ for some tuple $\vec{t}$ of variable-free terms

iff $I \models (\exists)A:\neg\mu$

(where $\vec{t}$ is a tuple of variable free terms and $\vec{x}$ is a tuple of all the variable symbols occurring in A). The proof for the universally quantified case is similar. □

Theorem 3.7. *I is a model of G iff I is a model of G^{pos}.*

The above theorem assures us that the device of annotations is powerful enough to make the use of negated atoms unnecessary. Therefore, from here on, we will always assume that GHPs do not contain any negated literals (with negations being *implicitly* present in the form of atoms annotated with **f**).

Example 3.8. Consider the GHP given below:

$$p(a):\mathbf{t} \Leftarrow, \qquad p(a):\mathbf{f} \Leftarrow.$$

This GHP has exactly one model, viz. the interpretation that assigns the truth value $\top$ to $p(a)$. This is in keeping with our intuition which says that this program contains contradictory information (in the two valued sense).

Example 3.9. Consider the GHP G given below:

$$p(a):\mathbf{t} \Leftarrow p(b):\mathbf{f}, \qquad p(b):\mathbf{t} \Leftarrow p(a):\mathbf{f}.$$

This GHP has several models as follows.

$I_1: I_1(p(a)) = \mathbf{t}$; $I_1(p(b)) = \mathbf{t}$, $I_6: I_6(p(a)) = \mathbf{t}$; $I_6(p(b)) = \bot$,

$I_2: I_2(p(a)) = \mathbf{t}$; $I_2(p(b)) = \mathbf{f}$, $I_7: I_7(p(a)) = \top$; $I_7(p(b)) = \top$,

$I_3: I_3(p(a)) = \mathbf{f}$; $I_3(p(b)) = \mathbf{t}$, $I_8: I_8(p(a)) = \mathbf{t}$; $I_8(p(b)) = \top$,

$I_4: I_4(p(a)) = \bot$; $I_4(p(b)) = \bot$, $I_9: I_9(p(a)) = \top$; $I_9(p(b)) = \mathbf{t}$.

$I_5: I_5(p(a)) = \bot$; $I_5(p(b)) = \mathbf{t}$,

The proliferation of models may appear somewhat bewildering, but it is important to observe that the GHP G has a least model (viz. I_4) and a greatest model (viz. I_7). The least model says that from G, it is not *known* that $p(a)$ is true and it is not *known* that $p(b)$ is true. Similarly, the model, I_3 says that if it is *known* that $p(a)$ is false then it is *known* that $p(b)$ is true.

Having defined the models of GHP, we now proceed to define a certain monotone operator from interpretations to interpretations. We then relate the prefixpoints[2] of this operator with the models of Generalized Horn Programs.

4. (Pre-)fixpoint semantics

It has now become a standard practice in logic programming research to try and naturally characterize the models of a program in terms of the (possibly pre- or post-)[2] fixpoints of a certain monotone operator usually denoted by T_P. If S is a subset of a complete lattice L, then $\bigsqcup S$ and $\bigsqcap S$ denote, respectively, the least upper bound of S and the greatest lower bound of S.

Definition 4.1. Suppose G is a GHP. Then T_G is a mapping from the Herbrand interpretations of G to the Herbrand interpretations of G defined by

$$T_G(I)(A) = \text{lub}\{\mu \mid A:\mu \Leftarrow B_1:\mu_1 \,\&\, \cdots \,\&\, B_k:\mu_k \text{ is a ground instance of a gh-clause in } G \text{ and } I \vDash B_1:\mu_1 \,\&\, \cdots \,\&\, B_k:\mu_k\}.$$

Example 4.2. Consider the GHP G given below:

$$p(a):\mathbf{t} \Leftarrow, \qquad p(a):\mathbf{f} \Leftarrow p(a):\mathbf{t}$$

and the interpretation I that assigns $\mathbf{t}$ to $p(a)$. Then $T_G(I)(p(a)) = \text{lub}\{\mathbf{t}, \mathbf{f}\} = \top$.

Theorem 4.3. *T_G is monotonic.*

Proof. Suppose $I_1 \preccurlyeq I_2$ where I_1, I_2 are interpretations. Then if $I_1 \vDash B:\mu$ then $I_2 \vDash B:\mu$ for all $B \in B_G$. Thus,

$$\begin{aligned} &T_G(I_1)(A) \\ &\quad = \text{lub}\{\mu \mid A:\mu \Leftarrow B_1:\mu_1 \,\&\, \cdots \,\&\, B_k:\mu_k \text{ is a ground instance of a gh-clause in } G \text{ and } I_1 \vDash B_1:\mu_1 \,\&\, \cdots \,\&\, B_k:\mu_k\} \\ &\quad \preccurlyeq \text{lub}\{\mu \mid A:\mu \Leftarrow B_1:\mu_1 \,\&\, \cdots \,\&\, B_k:\mu_k \text{ is a ground instance of a gh-clause in } G \text{ and } I_2 \vDash B_1:\mu_1 \,\&\, \cdots \,\&\, B_k:\mu_k\}. \quad \square \end{aligned}$$

[2] x is a pre- (resp. post-)fixpoint of a function $f: L \to L$ where L is a lattice under the $\sqsubseteq$ ordering iff $f(x) \sqsubseteq x$ (resp. $x \sqsubseteq f(x)$).

Before proceeding to investigate the fixpoints of T_G, we show that the pre-fixpoints of T_G are exactly the models of G.

Theorem 4.4. *I is a model of the* GHP *G iff $T_G(I) \preccurlyeq I$.*

Proof.

$T_G(I)(A) \preccurlyeq I(A)$
iff (directly from the definition of T_G), $\bigsqcup \{\mu \mid A : \mu \Leftarrow Body$ is a ground instance of a gh-clause in G and $I \vDash Body\} \preccurlyeq I(A)$
iff for every ground instance of the form $A : \mu \Leftarrow Body$ of a gh-clause in G, $I \vDash Body$ implies $I(A) \succcurlyeq \mu$
iff $I \vDash A : \mu \Leftarrow Body$ for all ground instances of the form $A : \mu \Leftarrow Body$ of gh-clauses in G
iff $I \vDash G$. □

Theorem 4.4 assures us that the models of G are exactly the pre-fixpoints of T_G. The monotonicity of T_G assures us that T_G has a fixpoint, and hence a pre-fixpoint, and hence a model. In addition, as T_G is monotone, and as the set of interpretations (over $\mathcal{T}$) is a complete lattice, the least fixpoint and the least pre-fixpoint of T_G must coincide, thus giving us the proof of the following result.

Theorem 4.5. *Any* GHP *G has a least model $\mathcal{M}_G$. In addition, this least model is identical to the least fixpoint* $\mathrm{lfp}(T_G)$ *of T_G.*

Unfortunately, monotonicity by itself does not guarantee us that the least fixpoint of T_G is semi-computable (i.e. recursively enumerable). We show, nevertheless, that this is indeed the case.

Definition 4.6. We define the special interpretation Δ to be the interpretation that assigns the value $\bot$ to all members of B_G. Similarly, the interpretation ∇ assigns the value $\top$ to all members of B_G.

Definition 4.7. The *upward iteration* of T_G is defined as follows:

$$T_G{\uparrow}0 = \Delta, \qquad T_G{\uparrow}\alpha = T_G(T_G{\uparrow}(\alpha - 1)), \qquad T_G{\uparrow}\lambda = \bigsqcup \{T_G{\uparrow}\eta \mid \eta < \lambda\}$$

where α is a successor ordinal and λ is a limit ordinal. The *downward iteration* of T_G is defined as:

$$T_G{\downarrow}0 = \nabla, \qquad T_G{\downarrow}(\alpha) = T_G(T_G{\downarrow}(\alpha - 1)), \qquad T_G{\downarrow}\lambda = \bigsqcap \{T_G{\downarrow}\eta \mid \eta < \lambda\}$$

where α is a successor ordinal and λ a limit ordinal ($\bigsqcap$ denotes glb).

Remark 4.8. We observe that $T_G\uparrow 0 \preccurlyeq T_G\uparrow 1 \preccurlyeq \cdots \preccurlyeq T_G\uparrow\omega$ and that $T_G\downarrow 0 \succcurlyeq T_G\downarrow 1 \succcurlyeq \cdots \succcurlyeq T_G\downarrow\omega$.

Theorem 4.9. $T_G\uparrow\omega = \mathcal{M}_G$.

Proof ($\mathcal{M}_G \preccurlyeq T_G\uparrow\omega$). In order to prove this, it is sufficient to show that $T_G\uparrow\omega$ is a model of G. Suppose

$$A:\mu \Leftarrow B_1:\psi_1 \,\&\, \cdots \,\&\, B_k:\psi_k$$

is a ground instance $C\theta$ of some gh-clause $C \in G$, and suppose

$$T_G\uparrow\omega \vDash B_1:\psi_1 \,\&\, \cdots \,\&\, B_k:\psi_k.$$

Then, it must be true that for some finite n,

$$T_G\uparrow n \vDash B_1:\psi_1 \,\&\, \cdots \,\&\, B_k:\psi_k.$$

But then, $T_G\uparrow(n+1) = T_G(T_G\uparrow n) \vDash A:\mu$ (by definition of T_G), hence it follows that $T_G\uparrow\omega \vDash A:\mu$ (by Remark 4.8). Thus, $T_G\uparrow\omega$ is a model of G, and is therefore a pre-fixpoint of T_G.

($T_G\uparrow\omega \preccurlyeq \mathcal{M}_G$). Since T_G is monotonic, for all α, $T_G\uparrow\alpha \preccurlyeq \mathcal{M}_G$ where $\mathcal{M}_G$ is the least prefixed-point of T_G. Hence, $T_G\uparrow\omega$ is the least prefixed-point of T_G. □

A subset D of a complete lattice $(L, \sqsubseteq)$ is *directed* iff for every pair of elements a, b in P, there is an element $c \in D$ such that $a \sqsubseteq c$ and $b \sqsubseteq c$.

Definition 4.10. Let $\bigsqcup S$ be the least upper bound (in L) of the subset S of L. An operator $T: L \to L$ is *weakly continuous* iff

$$T(\bigsqcup D) \sqsubseteq \bigsqcup \{T(d) \mid d \in D\}.$$

T is *continuous* iff

$$T(\bigsqcup D) = \bigsqcup \{T(d) \mid d \in D\}.$$

Lemma 4.11. *$T: L \to L$ is continuous iff T is monotonic and weakly continuous.*

Theorem 4.12. *T_G is continuous.*

Proof. Since T_G is monotone, it suffices to prove that T_G is weakly continuous. Recall that the set of interpretations of GHP G is the set of mappings (of type) $B_L \to \mathcal{T}$ and this set is a complete lattice where $I_1 \preccurlyeq I_2$ iff $I_1(A) \preccurlyeq I_2(A)$ for all $A \in B_L$. Let D be a non-empty directed subset of $B_L \to \mathcal{T}$, and let $J = \bigsqcup D$. We must show that $T_G(J) \preccurlyeq \bigsqcup \{T_G(I) \mid I \in D\}$. For this, it suffices to show that for all $A \in B_L$, there is an $I \in D$ such that $T_G(J)(A) = T_G(I)(A)$, but this follows at once since $T_G(J)(A) \in \mathcal{T}$ and $\mathcal{T}$ is finite. □

The *definite* part of an interpretation I is the set $\{A:\mu \mid I(A) = \mu \text{ and } \mu \neq \bot\}$. The continuity of T_G together with standard techniques of recursion theory yields that the definite part of each $T_G\uparrow n$, and hence the definite part of $T_G\uparrow\omega$ is recursively enumerable. Hence, the definite part of the least model of G is recursively enumerable. This assures us that if $\text{lfp}(T_G) \models A:\mu$ ($A \in B_G$ and $\mu \neq \bot$), then it is indeed possible to show that $\text{lfp}(T_G) \models A:\mu$. We will discuss SLD-resolution-like proof procedures for GHPs in Sections 6 and 7.

5. Well-behaved GHPs

Theorem 4.9 is likely to immediately raise the question: "Is $T_G\downarrow\omega$ identical to the greatest model?" Unfortunately, as in the case of logic programs (cf. [18, p. 31]), the answer to this question is negative. However, the reason for this is that $T_G\downarrow 0$ is the greatest model of G. Indeed, each $T_G\downarrow\alpha$ is a model of G, since it is a prefixed-point of T_G. Thus, we are led to ask the following.

Question 5.1. What kind of models would we like GHPs to have, and in addition, what kind of GHPs should be qualified as being acceptable or decent?

One answer is immediately forthcoming. We should like an *acceptable* GHP to be one that expresses consistent knowledge. Before proceeding to finish our answer to the above question, it is necessary to introduce a new definition.

Definition 5.2. An interpretation I of a GHP G is said to be *nice* if the following condition holds:

$$(\forall A \in B_G) I(A) \neq \top.$$

We are therefore interested in those models of G that are nice. Now it should be apparent that in view of the fact that $\top$ is a formalization of the notion of *both true and false* which is exactly the notion of inconsistency, we should like to characterize the *acceptable* GHPs to be exactly those GHPs that possess a nice model. Thus, we have now answered the preceding question in full:

(1) The models of a GHP that interest us are the nice models.

(2) The GHPs that we shall be interested in are the acceptable ones.

It is appropriate, therefore, to identify a class $\mathscr{C}$ of GHPs that are guaranteed to possess nice models. In addition, it is desirable that $\mathscr{C}$ be decidable.

Definition 5.3. A GHP G is *well-behaved* iff the gh-clauses of G satisfy the following condition:

> If G_1, G_2 are gh-clauses in G such that their (respective) heads are $A_1:\mu_1$, $A_2:\mu_2$ and if A_1, A_2 are unifiable, then $\mu_1 = \mu_2$.

It remains to prove our claim that every well-behaved GHP has a nice model. We go one step further and show that for well-behaved GHPs, $T_G\uparrow\omega$ is nice. We already know that $T_G\uparrow\omega$ is a model of G; so this is sufficient to substantiate our claim.

Lemma 5.4. *G has a nice model iff $T_G\uparrow\omega$ is nice.*

Theorem 5.5. *If G is a well-behaved* GHP, *then $T_G\uparrow\omega$ is nice.*

Proof. Suppose G is well-behaved and $T_G\uparrow\omega$ is not nice. Then there is some $A \in B_G$ such that $T_G\uparrow\omega(A) = \top$. But then, there exists some finite n such that $T_Q\uparrow n(A) = \top$ $(n > 0)$. But then there must exist two gh-clauses C_1, C_2 in G of the form

$$A_1 : \mathbf{t} \Leftarrow B_1 : \psi_1 \,\&\, \cdots \,\&\, B_m : \psi_m,$$

$$A_2 : \mathbf{f} \Leftarrow D_1 : \rho_1 \,\&\, \cdots \,\&\, D_k : \rho_k$$

such that A is an instance of A_1, A_2 and $T_G\uparrow(n-1) \models B_1 : \psi_1 \,\&\, \cdots \,\&\, B_m : \psi_m$ and $T_G\uparrow(n-1) \models D_1 : \rho_1 \,\&\, \cdots \,\&\, D_k : \rho_k$. As A is a common instance of A_1, A_2, it is true that A_1, A_2 are unifiable, thus contradicting the assumption that G is well behaved. Therefore, $T_G\uparrow\omega$ must be nice. □

Theorem 5.6. *In general, the set* $\mathrm{NICE}(G) = \{G \mid G$ *is a* GHP *having at least one nice model}* *is undecidable (and indeed is Π_1^0-complete).*

Proof. Given recursively enumerable set W, we can find an ordinary definite clause program P such that $T_P\uparrow\omega$ is recursively isomorphic to W (cf. [2]). Obtain GHP G from P by annotating all the *atoms* in P with **t**, and by adding the unit gh-clause $p(a) : \mathbf{f} \Leftarrow$. Then $T_G\uparrow\omega$ is a nice model of G iff $p(a) \notin T_P\uparrow\omega$. It follows, by standard techniques of recursion theory, that the problem of whether $T_G\uparrow\omega$ is a nice model of G, and hence by Lemma 5.4, the problem of whether G has a nice model is Π_1^0-complete. □

This is not a problem to worry about too much. We observe, for example, that deciding whether a classical logic program [18] is canonical [16] is Π_3^0-complete. Thus, we are forced to try to define recursive subsets of $\mathrm{NICE}(G)$. Theorem 5.6, together with the fact that $T_G\uparrow\omega$ is recursively enumerable, entails that the set $\mathrm{NICE}(G)$ is Π_1^0-complete.

Theorem 5.7. *If G is a well-behaved* GHP, *then $T_G\downarrow 1$ is nice; hence, $T_G\downarrow\alpha$ is nice for all $\alpha > 0$.*

Proof. Suppose G is well-behaved and $T_G\downarrow 1$ is not nice. Then there is some $A \in B_G$ such that $T_G\downarrow 1(A) = \top$. Then there must be at least two gh-clauses C_1, C_2 in G having ground instances of the following form:

$$A : \mathbf{t} \Leftarrow D_1 : \rho_1 \,\&\, \cdots \,\&\, B_k : \rho_k,$$

$$A : \mathbf{f} \Leftarrow E_1 : \psi_1 \,\&\, \cdots \,\&\, E_r : \psi_r$$

such that $T_G{\downarrow}0$ satisfies the bodies of both the above instances of C_1 and C_2. But then the heads of C_1, C_2 are unifiable, but the annotations of the heads are distinct, thus contradicting the well-behavedness of G. Therefore, $T_G{\downarrow}1$ is nice. □

Example 5.8. Consider the well-behaved GHP G given below:

$$p(a):\mathbf{t} \Leftarrow q(X):\mathbf{t}, \qquad q(s(X)):\mathbf{t} \Leftarrow q(X):\mathbf{t}.$$

In this case, $T_G{\uparrow}\omega = \Delta$, while

$$T_G{\downarrow}\omega(p(a)) = \mathbf{t}.$$

For every ground atom $A \in B_G$, $A \neq p(a)$, $T_G{\downarrow}\omega(A) = \bot$. However, $T_G{\downarrow}\omega(p(a)) = \mathbf{t}$. $T_G{\downarrow}\omega$ is not a fixed-point of T_G as $T_G{\downarrow}(\omega + 1)(p(a)) = \bot$.

The above example illustrates the fact that while $T_G{\uparrow}\omega$ is a fixpoint of T_G, $T_G{\downarrow}\omega$ may not be a fixpoint of T_G (of course, it is constrained to be a pre-fixpoint). We borrow the notion of a supported model from Apt et al. (cf. [3]).

Definition 5.9. A model I of a GHP G is said to be *supported* if for every ground atom $A \in B_G$ such that $I(A) \neq \bot$, there is a gh-clause in G having a ground instance of the form

$$A:\mu \Leftarrow B_1:\psi_1 \,\&\, \cdots \,\&\, B_m:\psi_m$$

such that $I \models B_1:\psi_1 \,\&\, \cdots \,\&\, B_m:\psi_m$ and $I(A) = \mu$.

Note that all literals occurring in a GHP are well annotated. Hence, $\mu \neq \top$. Thus, every supported model must be nice.

We now investigate the relationship between the fixpoints of T_G and the supported models of G.

Lemma 5.10. *If I is a supported (and hence, nice) model of the* GHP *G, then I is a fixpoint of T_G.*

Proof. Suppose I is a supported model of G and $I(A) = \mu$ and $\mu \notin \{\bot, \top\}$. Let $\Gamma = \{A:\rho \mid$ there is a ground instance C of a gh-clause in G such that C is $A:\rho \Leftarrow F$ and $I \models F\}$. $A:\mu \in \Gamma$ since I is a supported model of G. Since $I \models G$, $\mu = I(A) \geqslant \bigsqcup\{\rho \mid A:\rho \in \Gamma\} = T_G(I)(A)$. But $T_G(I)(A) \geqslant \mu$ since $\mu \in \{\rho \mid A:\rho \in \Gamma\}$. Therefore, $\mu = I(A) = T_G(I)(A)$. □

Lemma 5.11. *If I is a nice fixpoint of T_G, then I is a supported model of G.*

Proof. Assume $I(A) \neq \bot$. As I is nice, $I(A)$ must be in $\{\mathbf{t}, \mathbf{f}\}$.

Case 1: $[I(A) = \mathbf{t}]$ Therefore, since I is a fixed point of T_G, $T_G(I)(A) = \mathbf{t}$. That is, by definition, $\bigsqcup\{\mu \mid A:\mu \Leftarrow B_1:\psi_1 \,\&\, \cdots \,\&\, B_k:\psi_k$ is a ground distance of a

gh-clause in G and $I \models B_1 : \psi_1 \,\&\, \cdots \,\&\, B_k : \psi_k\} = \mathbf{t}$. But for the lub above to be $\mathbf{t}$, there must be a ground instance of the $A : \mathbf{t} \Leftarrow B_1 : \psi_1 \,\&\, \cdots \,\&\, B_k : \psi_k$ and such that $I \models B_1 : \psi_1 \,\&\, \cdots \,\&\, B_k : \psi_k$, i.e. I is supported.

Case 2: $[I(A) = \mathbf{f}]$ Similar to Case 1. □

Theorem 5.12. *I is a supported model of the* GHP *G iff I is a nice fixpoint of G.*

Jaffar et al. (cf. [15]) have introduced a *decency* thesis in conventional logic programming. Essentially, the decency thesis claims that all decent logic programs are canonical [16], i.e. $T_P{\downarrow}\omega = \text{gfp}(T_P)$ where T_P is defined as in [18]. The term *decent* as used by Jaffar et al. is intended to refer to programs that arise "naturally" in good programming practice. Under this criterion, the logic program obtained by deleting the annotations in Example 5.9 is *not* decent. We now give the definition of a canonical GHP.

Definition 5.13. A GHP G is *canonical* iff $T_G{\downarrow}\omega$ is a fixpoint of T_G.

It follows immediately from Lemma 5.11 that if G is a well-behaved canonical program, then $T_G{\downarrow}\omega$ is a supported model of G.

Theorem 5.14. *$T_G{\downarrow}\omega$ is the greatest (nice) supported model of G if G is a canonical well-behaved* GHP.

Proof. Since G is canonical, $T_G{\downarrow}\omega$ is a fixpoint of T_G, which must be the greatest fixpoint of T_G as T_G is monotonic. Since G is well-behaved, $T_G{\downarrow}\omega$ is nice. Thus, $T_G{\downarrow}\omega$ is the greatest supported model of G. □

With this, we conclude our declarative characterization of GHPs. The fact that $T_G{\downarrow}\omega$ is a supported nice model of G (G a canonical well-behaved GHP) is useful in characterizing the use of GHPs in reasoning about beliefs [9]. A more comprehensive discussion of decent GHPs is included in [9]. In addition, we have proved that the least model of a GHP is recursively enumerable and that when G is well behaved, the least model of G coincides with the least nice model of G (Theorem 5.5). We now investigate the operational semantics of GHPs which bears some similarity to the approach of Van Emden [25] for a quantitative deduction.

6. Operational semantics for GHPs

Definition 6.1. Suppose G is a GHP, A is a (not necessarily ground) atom in the language of G, and μ an annotation. We define an and/or tree $\mathbf{T}(G, A : \mu)$ as follows:

(1) The root of $\mathbf{T}(G, A : \mu)$ is an or-node labelled $A : \mu$.

(2) If N is an or-node, then it is labelled by a single annotated literal.

(3) Each and-node is labelled by a gh-clause from G and a substitution.

(4) Descendants of an or-node are and-nodes and descendants of and-nodes are or-nodes.

(5) If N is an or-node labelled by $B:\alpha$ $(\alpha \neq \perp)$, and if $C\theta$ is an instance of a gh-clause C in G of the following form:

$$B:\beta \Leftarrow D_1:\psi_1 \,\&\, \cdots \,\&\, B_k:\psi_k$$

where $\beta \geqslant \alpha$, then there is a descendant of N labelled by C and θ. An or-node with no descendants is called an *uninformative node.*

(6) If N is an and-node labelled by the gh-clause C and the substitution θ, then for every annotated literal $B:\gamma$ in the body of C, there is a descendant or-node labelled $B\theta:\gamma$. An and-node with no descendants is called a *success node.*

Associated with every node N in the and/or tree $\mathbf{T}(G, A, \mu)$ is a truth value $\nu(N)$ called the value of that node. ν is defined as follows:

If N is a success node labelled $B:\psi$, then $\nu(N) = \psi$.

If N is an uninformative node, then $\nu(N) = \perp$.

If N is an or-node that is not uninformative and its descendants are $N_1, \ldots, N_k$, then $\nu(N) = \text{lub}\{\nu(N_1), \ldots, \nu(N_k)\}$.

If N is a non-terminal and-node labelled with the gh-clause $A:\rho \Leftarrow B_1:\psi_1 \,\&\, \cdots \,\&\, B_k:\psi_k$, and if the value $\nu(N_i)$ of each of its descendant nodes N_i labelled B_i is such that $\nu(N_i) \geqslant \psi_i$ for all $1 \leqslant i \leqslant k$, then $\nu(N) = \rho$, else $\nu(N) = \perp$.

Before proceeding to investigate the soundness and completeness of the and/or tree search procedure just described, we give an example.

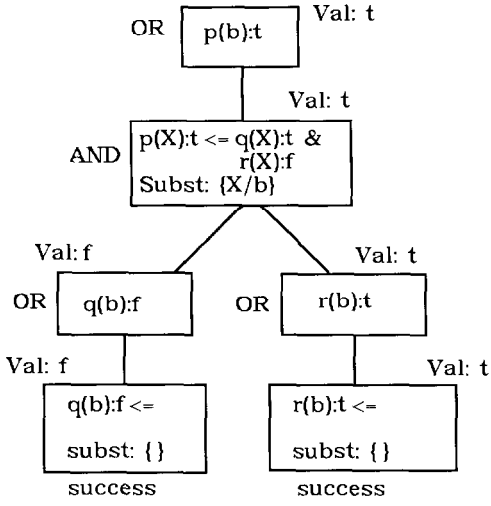

Fig. 2.

Example 6.2. Consider the GHP G given below:

$$p(a):\mathbf{t} \Leftarrow, \qquad p(X):\mathbf{t} \Leftarrow q(X):\mathbf{f} \,\&\, r(X):\mathbf{t},$$

$$r(a):\mathbf{t} \Leftarrow, \qquad r(b):\mathbf{t} \Leftarrow,$$

$$q(a):\mathbf{f} \Leftarrow, \qquad q(b):\mathbf{f} \Leftarrow.$$

Consider the problem of checking whether $p(b):\mathbf{t}$ is satisfied by $T_G\uparrow\omega$. The and/or tree associated with this problem is shown in Fig. 2. We see that the truth value of $p(b)$ in $T_G\uparrow\omega$ is $\mathbf{t}$.

Definition 6.3. A GHP G is said to be *covered* if every variable symbol that occurs in the body of a gh-clause $C \in G$ occurs in the head of C.

Theorem 6.4. *If G is a covered* GHP, $A \in B_G$ *and if* $\mathcal{T}(G, A:\mu)$ *is finite with root* R, *then* $\nu(R) \preccurlyeq T_G\uparrow\omega(A)$.

Proof. Similar to the proof of Theorem 3.1′ of Van Emden [25]. Observe that the proof in [25] applies only to covered rule sets. □

The reason for introducing *covered* GHPs is due to the following GHP Q and and/or tree.

Example 6.5

$$p:\mathbf{t} \Leftarrow q(X):\mathbf{t} \,\&\, r(X):\mathbf{t}, \qquad r(a):\mathbf{t} \Leftarrow, \qquad q(b):\mathbf{t} \Leftarrow.$$

Corresponding to this program Q and query $p:\mathbf{t}$, we have the AND/OR tree shown in Fig. 3. However, $T_Q\uparrow\omega(p) = \bot$, though the truth value associated with the root

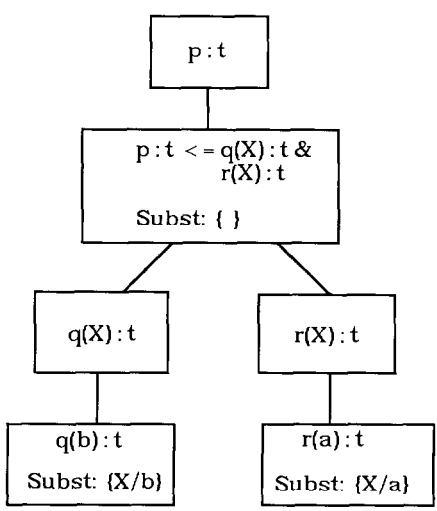

Fig. 3.

of this tree is **t**. For covered programs, the atoms occurring in OR-nodes of and/or trees are ground. This restriction to analogously defined covered rule sets is also necessary for the theorems of [25].

Theorem 6.6. *If G is a* GHP, $A \in B_G$, *and if $\mathcal{T}(G, A:\mu)$ is finite with root R, then $\nu(R) \geqslant T_G{\uparrow}\omega(A)$.*

Proof. Let R be the root of $\mathcal{T}(G, A:\mu)$. It suffices to prove by induction that

$$(\forall n \in \mathbb{N})\nu(R) \geqslant T_Q{\uparrow}n(A)$$

(where $\mathbb{N}$ denotes the set of non-negative integers).

Base case ($n = 0$): Trivially true as $T_G{\uparrow}0(A) = \bot$.

Inductive case: Let $\alpha = T_G{\uparrow}(n+1)(A)$.

Case 1 ($\alpha = \top$): Then there are two ground instances $C_1\theta_1$, $C_2\theta_2$ of gh-clauses $C_1, C_2 \in G$ of the form

$$A:\mathbf{t} \Leftarrow B_1:\psi_1 \,\&\, \cdots \,\&\, B_r:\psi_r,$$

$$A:\mathbf{f} \Leftarrow D_1:\rho_1 \,\&\, \cdots \,\&\, D_s:\rho_s$$

and $T_G{\uparrow}n$ satisfies the bodies of $C_1\theta_1$ and $C_2\theta_2$, respectively. Each B_i and D_j is ground, and therefore, by the induction hypothesis, $\nu(B_i') \geqslant \psi_i$ and $\nu(D_j') \geqslant \rho_j$ where B_i' and D_j' are the atoms in the roots of the and/or trees $\mathcal{T}(G, B_i:\psi_i)$ and $\mathcal{T}(G, D_j:\rho_j)$, respectively. Therefore, the descendant nodes N_{C_1}, N_{C_2} of R that are labelled with C_1, C_2, respectively, are such that $\nu(N_{C_1}) \geqslant \mathbf{t}$, $\nu(N_{C_2}) \geqslant \mathbf{f}$, respectively, whence, $\nu(R) \geqslant \text{lub}\{\mathbf{t}, \mathbf{f}\} = \top$. This completes ths case.

Case 2 ($\alpha = \bot$): Trivial.

Case 3 ($\alpha = \mathbf{t}$ or $\mathbf{f}$): In this case, the analysis is almost identical to that of the case when $\alpha = \top$ except that we consider only one gh-clause C_1 instead of two. □

Corollary 6.7 (Soundness). *If R is a covered* GHP, $A \in B_G$ *and $\mathcal{T}(G, A:\mu)$ is finite, then $\nu(R) = T_G{\uparrow}\omega(A)$ where R is the root of $\mathcal{T}(G, A:\mu)$.*

As in the case of [25], the restriction to finite and/or trees seems to be necessary. The following question then arises:

Question 6.8. Can an SLD-resolution like proof procedure be given for GHPs?

The answer to this question seems to be "NO". This is because a certain amount of breadth-oriented search seems necessary in order to compute $T_G{\uparrow}\omega(A)$ for $A \in B_G$. However, in the case of *well-behaved* GHPs, it does seem to be possible to define an SLD-resolution like proof procedure.

7. SLDnh-resolution for well-behaved GHPs

Recall that a literal is well annotated if the annotation is either **t** or **f**. A *query* is the existential closure of a conjunction of w-annotated literals. From here on, when

discussing queries, we will often write them as a conjunction of w-annotated literals and assume all variables in it to be implicitly existentially quantified.

Definition 7.1. An nh-*resolvent* with respect to $A_i : \rho_i$ of the query Q given by $A_1 : \rho_1 \,\&\, \cdots \,\&\, A_k : \rho_k$ and the gh-clause C of the form $D : \beta \Leftarrow B_1 : \psi_1 \,\&\, \cdots \,\&\, B_r : \psi_r$ is the query

$$(A_1 : \rho_1 \,\&\, \cdots \,\&\, A_{i-1} : \rho_{i-1} \,\&\, B_1 : \psi_1 \,\&\, \cdots \,\&\, B_r : \psi_r \,\&\, A_{i+1} : \rho_{i+1} \,\&\, \cdots \,\&\, A_k : \rho_k)\theta$$

where $\beta \geqslant \rho_i$ and θ is the most general unifier of D and A_i. (Without loss of generality, we assume that C and Q contain no variable symbols in common, i.e. we assume that they have been standardized apart, cf. Lloyd [18].) $A_i : \rho_i$ and $D : \rho$ are called "the literals nh-resolved upon" ("nh" stands for "non-Horn"). Note that an nh-resolvent is always a query.

Definition 7.2. An SLDnh-*deduction* from the *initial* query Q_0 and the GHP G is a sequence

$$\langle Q_0, C_1, \theta_1\rangle, \ldots, \langle Q_i, C_{i+1}, \theta_{i+1}\rangle, \ldots$$

where Q_{r+1} is the nh-resolvent of Q_r and C_{r+1} and C_{r+1} is a renamed version of some gh-clause in G such that C_{r+1} has no variable symbols in common with any of $Q_0, \ldots, Q_r$, $C_1, \ldots, C_r$, and θ_{r+1} is the most general unifier of the literals nh-resolved upon.

Definition 7.3. An SLDnh-*refutation* of the initial query Q_0 is a finite SLDnh-deduction

$$\langle Q_0, C_1, \theta_1\rangle, \ldots, \langle Q_n, C_{n+1}, \theta_{n+1}\rangle$$

such that the result of resolving Q_n and Q_{n+1} is the empty gh-clause. Associated with every SLDnh-refutation is a path called the SLDnh-*refutation path.* Figure 4 gives the SLDnh-refutation path associated with the above SLDnh-refutation.

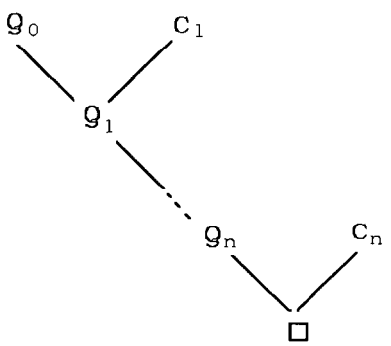

Fig. 4.

We now address the soundness of SLDnh-resolution.

Theorem 7.4 (Soundness). *If there exists an* SLDnh-*refutation of the initial query* Q_0

$$A_1 : \rho_1 \,\&\, \cdots \,\&\, A_m : \rho_m$$

from the GHP G, *then*

$$T_G \uparrow \omega \models \exists(Q_0).$$

Proof. Suppose there exists an SLDnh-refutation of the initial query Q_0 from G of the following form:

$$\langle Q_0, C_1, \theta_1 \rangle, \ldots, \langle Q_n, C_{n+1}, \theta_{n+1} \rangle.$$

We will show by induction on n that $T_Q \uparrow \omega \models \exists(Q_0)$.

Base case ($n = 0$): Then the nh-resolvent of Q_0 and C_1 is the empty query, i.e. Q_0 is a unit conjunction (i.e. $A_1 : \rho_1$) and C_1 is a unit gh-clause, i.e. C_1 is of the form $A' : \beta \Leftarrow$ where $\beta \geqslant \rho_1$ and $A_1\theta_1 = A'\theta_1$. As $T_G \uparrow \omega$ is a model of the GHP G, it must be a model of C_1, and so it must be true that for every ground instance A_2 of A', $T_G \uparrow \omega(A_2) \geqslant \beta$. In particular, for every A_3 that is a ground instance of $A'\theta_1$, $T_G \uparrow \omega(A_3) \geqslant \beta \geqslant \rho_1$, whence $T_G \uparrow \omega \models \exists Q_0$.

Induction case: Suppose

$$\langle Q_0, C_1, \theta_1 \rangle, \ldots, \langle Q_n, C_{n+1}, \theta_{n+1} \rangle, \langle Q_{n+1}, C_{n+2}, \theta_{n+2} \rangle$$

is an SLDnh-refutation of Q_0. Then

$$\langle Q_1, C_2, \theta_2 \rangle, \ldots, \langle Q_n, C_{n+1}, \theta_{n+1} \rangle$$

is an SLDnh-refutation of Q_1. Therefore, by the induction hypothesis, $T_G \uparrow \omega \models \exists(Q_1)$. But Q_1 is the nh-resolvent of Q_0 and C_1. So Q_1 is of the form

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

where C_1 is the gh-clause

$$H : \delta \Leftarrow E_1 : \psi_1 \,\&\, \cdots \,\&\, E_r : \psi_r$$

such that $H\theta_1 = A_i\theta_1$ and $\delta \geqslant \rho_i$ (by definition of nh-resolution). As $T_G \uparrow \omega \models (\exists) Q_1$ it follows that

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

(as this is a sub-conjunct of Q_1). As $T_G \uparrow \omega$ is a model of G (Theorem 4.9), it must satisfy every gh-clause of G (and every renamed version of any gh-clause in G). In particular, $T_G \uparrow \omega$ must satisfy $C_1\theta_1$. As $T_G \uparrow \omega$ satisfies the body of $C_1\theta_1$, 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, as $\delta \geqslant \rho_i$, $T_G \uparrow \omega \models A_i\theta_1 : \rho_i$. Thus,

$$T_G \uparrow \omega \models (\exists)(A_1 : \rho_1 \,\&\, \cdots \,\&\, A_m : \rho_m)\theta_1. \qquad \square$$

Notice that the soundness theorem above does not restrict the GHP to covered GHPs. However, this restriction is needed for our completeness result.

Theorem 7.5 (Completeness). *Suppose G is a covered, well-behaved* GHP. *Then if $Q_0 = A_1 : \rho_1 \,\&\, \cdots \,\&\, A_m : \rho_m$ is a ground query that is satisfied by $T_G \uparrow \omega$, then there is an* SLDnh-*refutation of Q_0 from G.*

Proof. Suppose G is covered and well behaved. Let $Q_0 = (A_1 : \rho_1 \,\&\, \cdots \,\&\, A_m : \rho_m)$ be a ground query that is satisfied by $T_G \uparrow \omega$. Then there must be some $n > 0$ (as $\rho_i \neq \perp$ for all i) such that $T_G \uparrow n \models Q_0$. We will prove by induction on n that an SLDnh-refutation of Q_0 from G exists.

Base case: Suppose $T_G \uparrow 1 \models Q_0$. Then for each $1 \leq i \leq m$, there is a gh-clause C_i in G having a ground instance $C_i\theta_i$ of the form $A : \beta_i \Leftarrow$ where $\beta_i \geq \rho_i$. As Q_0 is ground, none of the substitutions θ_i affect it, and so

$$\langle A_1 : \rho_1 \,\&\, \cdots \,\&\, A_m : \rho_m, C_1, \theta_1 \rangle,$$

$$\langle A_2 : \rho_2 \,\&\, \cdots \,\&\, A_m : \rho_m, C_2, \theta_2 \rangle$$

$$\vdots$$

$$\langle A_m : \rho_m, C_m, \theta_m \rangle$$

is an SLDnh-refutation of Q_0.

Inductive case: Suppose $T_G \uparrow (r+1) \models A_1 : \rho_1 \,\&\, \cdots \,\&\, A_m : \rho_m$. Then, for each A_i, there is a gh-clause C_i in G having a ground instance $C_i\theta_i$ of the form

$$A_i : \beta_i \Leftarrow B_1^i \,\&\, \cdots \,\&\, B_{\kappa_i}^i$$

where $\beta_i \geq \rho_i$, and $T_G \uparrow r \models B_1^i \,\&\, \cdots \,\&\, B_{\kappa_i}^i$. Now, as each A_i is ground, and as G is covered, $B_1^i \,\&\, \cdots \,\&\, B_{\kappa_i}^i$ is ground and so, by the induction hypothesis, there exists an SLDnh-refutation $\mathbf{R_i}$ of $B_1^i \,\&\, \cdots \,\&\, B_{\kappa_i}^i$. Therefore,

$$\langle A_i : \rho_i, C_i, \theta_i \rangle, \mathbf{R_i}$$

is a SLDnh-refutation of A_i. Denote the SLDnh-refutation tree of each $A_i : \rho_i$ by Γ_i. Then Fig. 5 gives an SLDnh-refutation tree for Q_0. □

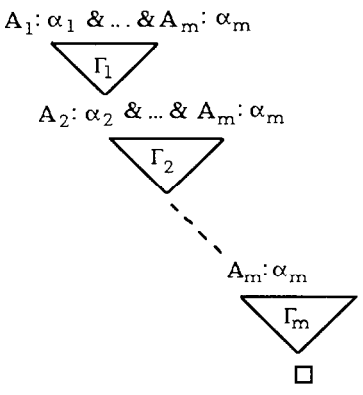

Fig. 5.

Example 7.6. Consider the following covered, well-behaved GHP:

$$p(X):\mathbf{t} \Leftarrow q(X):\mathbf{f} \,\&\, r(X):\mathbf{t},$$

$$q(a):\mathbf{t} \Leftarrow, \quad q(b):\mathbf{f} \Leftarrow, \quad q(c):\mathbf{f} \Leftarrow,$$
$$r(a):\mathbf{t} \Leftarrow, \quad r(a):\mathbf{f} \Leftarrow, \quad r(b):\mathbf{f} \Leftarrow, \quad r(c):\mathbf{t} \Leftarrow,$$

and the queries $p(b):\mathbf{t}$ and $p(c):\mathbf{t}$. The latter succeeds, but the former does not. This is despite the fact that the above GHP expresses inconsistency via the fifth and sixth clauses. This is a GHP that expresses inconsistency via the annotations, but is not inconsistent in the classical sense (i.e. it has models, and does not entail everything).

8. Summary and future work

We have developed a fixpoint semantics for arbitrary sets of clauses. Based upon the intuitive connection between arbitrary sets of clauses and positive GHPs, we have given a means of associating a fixed point semantics with sets of arbitrary clauses. It is clear that a GHP that expresses inconsistency via the annotations need not be classically inconsistent (in terms of having no models). Example 7.6 amply demonstrates this fact. Thus, GHPs provide a theoretical framework for mechanical reasoning in systems that are inconsistent.

Some open problems remaining are:

Problem 1. Can we find a recursive class of GHPs that (strictly) includes the class of well-behaved GHPs, while still guaranteeing existence of a nice model?

Problem 2. How much can the covering condition used in the completeness result of SLDnh-resolution be weakened? In view of the very strong conditions imposed on completeness results for SLDNF-resolution for general logic programs [5, 6] it should not be surprising that we have such a restriction in Theorem 7.5.

Acknowledgment

We thank R. Anand for typesetting the diagrams.

References

[1] R. Anand and V.S. Subrahmanian, FLOG: A logic programming system based on a six-valued logic, in: *Proc. AAAI Intl. Symp. on Knowledge Engg.*, Madrid, Spain (1987).

[2] H. Andreka and I. Nemeti, The generalized completeness of Horn predicate logic as a programming language, *Acta Cybernet.* **4** (1978) 3–10.

[3] K.R. Apt, H.A. Blair and A. Walker, Towards a theory of declarative knowledge, in: J. Minker, ed., *Foundations of Logic Programming and Deductive Databases* (Morgan Kauffman, Los Altos, CA, 1988).

[4] A.I. Arruda, A survery of paraconsistent logic, in: A.I. Arruda, R. Chuaqui and N.C.A. da Costa, eds., *Mathematical Logic in Latin America* (Reidel, Dordrecht, 1979) 1-41.

[5] R. Barbuti and M. Martelli, Completeness of the SLDNF-resolution for a class of logic programs, in: E. Shapiro, ed. *Proc. Intl. Symp. on Logic Prog.*, Lecture Notes in Computer Science **225** (Springer, Berlin, 1986), 600-614.

[6] R. Barbuti and M. Martelli, Completeness of SLDNF-resolution for structured programs, submitted for publication, 1986.

[7] N.D. Belnap, A useful four-valued logic, in: G. Eppstein and J.M. Dunn, eds., *Modern Uses of Many-valued Logic* (Reidel, Dordrecht, 1977) 8-37.

[8] H.A. Blair, Decidability in the Herbrand Base, in: J. Minker, ed., *Proc. Workshop on Foundations of Logic Programming and Deductive Databases*, College Park, MD, 1986.

[9] H.A. Blair and V.S. Subrahmanian, Foundations of generally Horn logic programming, in preparation.

[10] NC.A. da Costa, On the theory of inconsistent formal systems, *Notre Dame J. Formal Logic* **15** (1974) 497-510.

[11] N.C.A. da Costa and E.H. Alves, A semantical analysis of the calculi $\mathscr{C}_n$, *Notre Dame J. Formal Logic* **18** (1977) 621-630.

[12] N.C.A. da Costa and E.H. Alves, Relations between paraconsistent logic and many-valued logic, *Bull. Section of Logic* **10** (1981) 185-191.

[13] N.C.A. da Costa and D. Marconi, An overview of paraconsistent logic in the 80's, in: *Logica Nova* (Akademie Verlag, Berlin) to appear.

[14] M.C. Fitting, A Kripke-Kleene semantics for logic programming, *J. Logic Program.* **4** (1985) 295-312.

[15] J. Jaffar, J.-L. Lassez and M. Maher, Issues and trends in the semantics of logic programming, in: E. Shapiro, ed., *Proc. 3rd Intl. Conf. on Logic Programming*, Lecture Notes in Computer Science **225** (Springer, Berlin, 1986).

[16] J. Jaffar and P.J. Stuckey, Canonical logic programs, *J. Logic Program.* **3** (1986) 143-155.

[17] S.C. Kleene, *Introduction to Metamathematics* (Van Nostrand Reinhold, New York, 1957).

[18] J. W. Lloyd, *Foundations of Logic Programming* (Springer, Berlin, 1984).

[19] J.-L. Lassez and M. Maher, Optimal fixed points of logic programs, *Theoret. Comput. Sci.* **39** (1985) 115-125.

[20] L. Naish, Negation and quantifiers in NU-Prolog, in: E. Shapiro, ed., *Proc. 3rd Intl. Conf. on Logic Programming*, Lecture Notes in Computer Science **225** (Springer, Berlin, 1986) 624-634.

[21] D. Perlis, On the consistency of commonsense reasoning, *Comput. Intelligence* **2** (1986) 180-190.

[22] D. Poole and R. Goebel, Gracefully adding negation and disjunction in Prolog, in: E. Shapiro, ed., *Proc. 3rd Intl. Conf. on Logic Programming*, Lecture Notes in Computer Science **225** (Springer, Berlin, 1986) 635-641.

[23] V.S. Subrahmanian, On the semantics of quantitative logic programs, in: *Proc. 4th IEEE Symp. on Logic Prog.* (Computer Society Press, Washington, DC, 1987) 173-182.

[24] V.S. Subrahmanian, Towards a theory of evidential reasoning in logic programming, *Logic Colloquium '87 (European Summer Meeting of the Association of Symbolic Logic)*, Granada, Spain, 1987.

[25] M.H. Van Emden, Quantitative deduction and its fixpoint theory, *J. Logic Program.* **4** (1986) 37-53.

[26] M.H. Van Emden and R.A. Kowalski, The semantics of predicate logic as a programming language, *J. ACM* **23** (1976) 733-742.

[27] A. Visser, Four valued semantics and the liar, *J. Philos. Logic* **13** (1984) 181-212.