A Logic for Reasoning with Inconsistency [*]

(appeared in Journal of Automated Reasoning, 1992)

Michael Kifer
Department of Computer Science
SUNY at Stony Brook
Stony Brook, NY 11794, USA
kifer@cs.sunysb.edu

Eliezer L. Lozinskii
Department of Computer Science
The Hebrew University
Jerusalem 91904, Israel

Abstract

Most known computational approaches to reasoning have problems when facing inconsistency, so they assume that a given logical system is consistent. Unfortunately, the latter is difficult to verify and very often is not true. It may happen that addition of data to a large system makes it inconsistent, and hence destroys the vast amount of meaningful information. We present a logic, called APC (annotated predicate calculus; cf. annotated logic programs of [3], that treats any set of clauses, either consistent or not, in a uniform way. In this logic, consequences of a contradiction are not nearly as damaging as in the standard predicate calculus, and meaningful information can still be extracted from an inconsistent set of formulae. APC has a resolution-based sound and complete proof procedure. We also introduce a novel notion of "epistemic entailment" and show its importance for investigating inconsistency in predicate calculus as well as its application to nonmonotonic reasoning. Most importantly, our claim that a logical theory is an adequate model of human perception of inconsistency, is actually backed by rigorous arguments.

1 Preface

Most existing logical systems provide no means for reasoning in the presence of inconsistency. For instance, consider the following set of facts: S=*flies(tweety)*, $\neg flies(tweety)$, $grade(john, A)$. Although there is certain amount of inconsistency in S, only the knowledge regarding tweety being able to fly is affected. Intuitively, this inconsistency should have no implication regarding John's grade. However, since S has no model, the standard predicate calculus would warrant any conclusion, including $grade(john, F)$. On the other hand, checking or guaranteeing consistency in large knowledge-base systems is very expensive, if at all possible. Therefore semantics of such systems cannot be based (at least directly) on standard logic. Lacking an alternative, many system designers either assume that their systems are always consistent — hardly a realistic assumption, or opt for syntactic approaches to inference, disregarding semantic issues. In the latter case, the user may never be sure of the meaning of his program.

Thus, it is desirable to devise a logic in which inconsistency would be not as destructive as in predicate calculus. The goal of the present paper is to propose just such a logic, which we call APC (*annotated* predicate calculus; cf. annotated logic programs of [3]. Furthermore, unlike

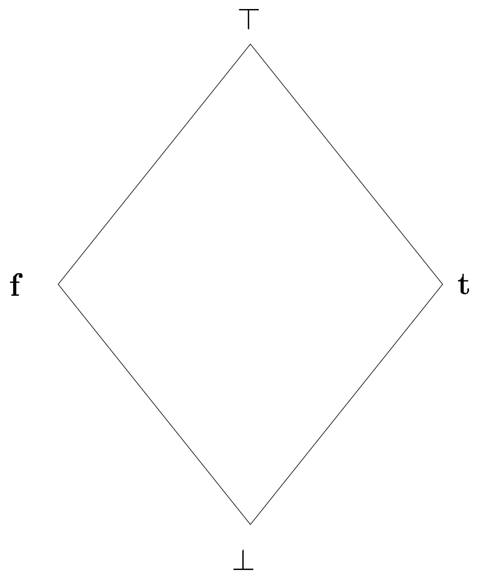
[*]A preliminary report on this research appeared in LICS'89. Work of M. Kifer was supported in part by the NSF grants DCR-8603676, IRI-8903507. Work of E.L. Lozinskii was supported in part by Israel National Council for Research and Development under the grants 2454-3-87, 2545-2-87, 2545-3-89 and by Israel Academy of Science, grant 224-88.

many other works that propose logics to deal with various reasoning phenomena, our approach is justified not only by a number of carefully selected examples but by a host of rigorous arguments presented in Section 6. In this section we provide a formalization of the intuition about the causes of inconsistency in predicate calculus (with function symbols) and then show that these precisely correspond to inconsistent beliefs in APC.

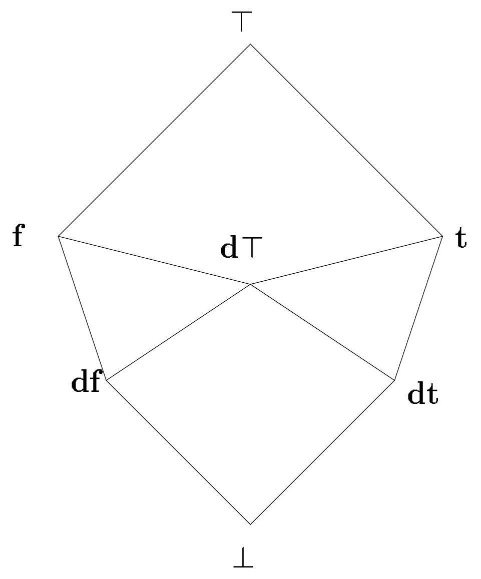
This paper is organized as follows. Section 2 introduces the syntax and semantics of annotated predicate calculus. Section 3 discusses the notions of entailment and negation and in Section 4 we give a number of examples of nonmonotonic reasoning in the presence of inconsistency. Section 5 presents a proof theory for the proposed logic. In Section 6 we argue that inconsistent beliefs in APC represent, in a well defined sense, what humans are likely to regard as causes of inconsistency in predicate calculus. Section 7 discusses various applications and extensions of APC. In Section 8, we present an alternative semantics for APC. Although it is slightly more complicated than the one given in Section 2.2, it has an advantage that the completeness theorem holds for arbitrary sets of formulas; with the semantics of Section 2.2 completeness is guaranteed only for restricted sets of clauses. Some of the proofs are delegated to Appendices A and B.

2 Introduction to APC

Several researchers have introduced and demonstrated the usefulness of lattice-valued logics. Due to these approaches, each closed formula may be assigned a truth value drawn from a lattice instead of the traditional $\{true, \; false\}$. Two popular lattices to be used in the examples are depicted in Figure 1.



(a) A 4-valued Lattice [2]



(b) A Lattice with Defaults [13]

Figure 1: Examples of Lattices

Belnap [2] was advocating the use of such logics as a general means for handling inconsistency, while Ginsberg [13] and Sandewal [29] have shown their utility in explaining Reiter's results on Default Logic [27]. Quite recently, Fitting [11] proposed a fixpoint semantics for bilattice-valued logic programs. With all their elegance, these approaches suffer from a number of drawbacks:

the lack of the notion of tautology (in the traditional sense) leads to difficulties in defining proof procedures and to the need for additional complex truth-related notions such as "formula closure" [13]. The notion of logical entailment also differs from the standard one and there is an unpleasant asymmetry in the semantics of implication (at least, in [2, 29, 11]): normally logic formulas may assume any truth value from the lattice, except for implications, which are strictly 2-valued.

Underlying APC is an upper semilattice of "degrees of belief", a *belief semilattice*, *BSL*. An upper semilattice is a partial order with a unique least upper bound for every pair of elements; the existence of greatest lower bounds is not mandatory. We assume that BSL satisfies the following conditions:

(i) It contains at least the following four distinguished elements: $\mathbf{t}$ (true), $\mathbf{f}$ (false), $\top$ (contradiction), and $\bot$ (unknown);

(ii) For every $\mathbf{s} \in BSL$, $\bot \leq \mathbf{s} \leq \top$ ($\leq$ is the semilattice ordering);

(iii) $\texttt{lub}(\mathbf{t}, \mathbf{f}) = \top$, where `lub` denotes the least upper bound.[1]

The role of BSL in APC is different from that in the lattice-valued logics. In APC, the belief semilattice is part of the object language rather than the range of truth valuations. Unlike [2, 13, 11, 29] semantic structures of APC remain 2-valued, the notion of tautology is preserved, and there is a sound and complete resolution-based proof procedure.

We distinguish between what the reasoner believes in (at the *epistemic* level),[2] and what is actually true or false in the real world (at the *ontological* level). As will be shown in Section 6, APC can faithfully interpret the standard predicate calculus at its ontological level, thus being intolerant to inconsistency, or it can interpret the calculus epistemically being able to tolerate inconsistency in full (or, it can mix the two interpretations).

Some of the ideas about treatment of inconsistency used in APC appeared in [15, 3]. However, these works are not as general as the present paper, since they are restricted to a subset of logic, and as explained later, consider only one kind of negation which we call "epistemic." In the present work, we allow both epistemic and ontological negation, and describe a sound and complete resolution-based theorem-proving procedure for the *general* logic. We also investigate the relationship of inconsistency in predicate calculus to epistemic and ontological inconsistency in APC.

Elements of our approach can be traced back even further to [33, 30]. However, [33] does not handle inconsistency and negation at all, while [30] considers only a subset of the logic. There is also an abundance of literature (e.g. [6, 28]) describing other approaches to inconsistency. We are not discussing them here since they employ quite different techniques from the ones used in this paper.

APC is close in spirit to the logics derived from Post and semi-Post algebras [26]. However, these logics do not deal with inconsistency and their negation is intuitionistic. In contrast, APC has two negations: ontologic and epistemic, both of which are classical (although it is possible to include an "epistemic intuitionistic negation", if desired). Furthermore, the Post algebra based logic is less expressive than APC and can be imitated by APC, while the logic of [26]. is too expressive to have a resolution-based theorem proving procedure.

Inconsistency can be also approached using various modal logics of belief (e.g. [8, 17]). In these logics, the sentence $B(\phi) \wedge B(\neg\phi)$ is satisfiable, as is a similar sentence in APC (APC has no modal operators, but uses different means for representing beliefs; see Section 2). However, the aforesaid logics are propositional, i.e., they do not support logical variables and quantification. Even though the validity problem in quantified modal logics is known to be recursively enumerable, no

[1]Some of these conditions are not needed either for the proof theory, or for the results of Section 6. Section 7 summarizes conditions needed for each of the results.

[2]One of the referees has pointed out that the term "epistemic" may be inappropriate here since it usually refers to knowledge rather than belief. However, having consulted with some books on philosophical logic, e.g. [14], we have found that the word "epistemic" is being used for beliefs as well as knowledge. We therefore stick to our original terminology, also because we find it more palatable than the term "doxastic" suggested by the referee.

efficient proof procedure exists (but see [12, 22, 9] for some proposals). In contrast, APC supports quantification and has a resolution-based semi-decision procedure that is comparable in efficiency to the standard predicate calculus.

In a sense, logics of belief are more powerful than APC. For instance, a statement $B(\phi \vee \psi)$ has no natural representation in APC (but there is an analogue of $B(\phi) \vee B(\psi)$). On the other hand, unlike logics of belief, literals in APC can be believed in to various degrees, so that it becomes possible to differentiate between, say, necessary and default conclusions (see Section 4 for additional remarks).

2.1 Syntax of APC

The language of APC is similar to that of predicate calculus. It contains predicates, function symbols, constants, variables, quantifiers, and logical connectives. The only syntactic difference is that atomic formulas in our logic are constructed from those of predicate calculus (e.g. l) by appending to them *annotations* that are drawn from BSL (e.g. $l : \mathbf{r}$). Informally, an annotation, $\mathbf{r}$, represents strength of reasoner's belief in the truth of the statements annotated with $\mathbf{r}$ (l in our case). For example, $p(X) : \mathbf{s}$, $q : \top$, and $\neg r(X, g(Y, Z)) : \mathbf{l}$, where $\mathbf{s}, \mathbf{l}, \top \in BSL$, are literals of APC. Similarly, $(\forall Y)\, r(Y) : \mathbf{t}$, $(\exists X) \neg q(X, Z) : \mathbf{f}$, and $(\forall X)(\exists Y)\, (p(g(X, Y), Z) : \mathbf{s} \vee q(X) : \bot)$ are formulas. We also note that $\neg p : \mathbf{s}$ means $\neg(p : \mathbf{s})$ rather than $(\neg p) : \mathbf{s}$. In fact, the latter is not a well-formed formula according to the above definition.

Occasionally we will refer to atomic formulas of APC as "annotated atoms" to contrast them with "non-annotated" atoms — the atomic formulas of PC.

2.2 Semantics of APC

As in predicate calculus, given an APC language, L, a *semantic structure*, I, is a tuple $< D, I_F, I_P >$. Here D is the domain of I; I_F associates to each k-ary function symbol f of L a mapping $I_F(f) : D^k \longrightarrow D$; and I_P associates to each m-ary predicate symbol p a function $I_P(p) : D^m \longrightarrow BSL$.

A *valuation*, v, assigns values from D to variables. This mapping can be extended to terms as usual in the first-order logic by combining v and I_F: $v(f(\ldots, s, \ldots)) = I_F(f)(\ldots, v(s), \ldots)$. So, for a term t, $v(t) \in D$.

For an atomic formula $p(t_1, \ldots, t_k) : \mathbf{s}$, we write $I \models_v p(t_1, \ldots, t_k) : \mathbf{s}$ if and only if $\mathbf{s} \leq I_P(p)(v(t_1), \ldots, v(t_k))$. This reflects the view that if the reasoner believes in $p(\bar{a})$ to the degree $\mathbf{r}$ then he also believes in it to any smaller degree. The rest follows the standard definitions of predicate calculus:

$I \models_v \phi \vee \psi$ if and only if $I \models_v \phi$ or $I \models_v \psi$;
$I \models_v \phi \wedge \psi$ if and only if $I \models_v \phi$ and $I \models_v \psi$;
$I \models_v \neg\psi$ if and only if not $I \models_v \psi$;
$I \models_v (\forall X)\psi$ if and only if $I \models_u \psi$, for *every* u that may differ from v only in its X-value;
$I \models_v (\exists X)\psi$ if and only if $I \models_u \psi$, for *some* u that may differ from v only in its X-value.

We say that ϕ is *satisfied* by I if and only if for every valuation v, $I \models_v \phi$. In this case we write $I \models \phi$.

Following the standard definitions, we say that a semantic structure I is a *model* of a set of formulas S if and only if every formula ϕ in S is satisfied in I. A set of formulas S *logically entails* a formula ϕ, denoted $S \models \phi$, if and only if every model of S is also a model of ϕ.

3 Inconsistency, Negation, and Implication in APC

3.1 Inconsistency: Ontological vs. Epistemic

Example 1: Consider a set of ground formulas $S = \{p : \mathbf{t},\ p : \mathbf{f} \vee q : \mathbf{t},\ p : \mathbf{f} \vee q : \mathbf{f},\ r : \mathbf{t}\}$. If one tried to express the same information in predicate calculus (PC), the most likely result would be $S' = \{p,\ \neg p \vee q,\ \neg p \vee \neg q,\ r\}$. □

In PC, S' is inconsistent, so the standard predicate calculus warrants every conclusion, in particular both r and $\neg r$. Intuitively, we may notice the difference in the contribution to the inconsistency made by r and the first three formulas of S'. Indeed, the latter are inconsistent regardless of the literal r, which has nothing to do with p and q.

It is our desire to turn this intuition into a result of a logically correct reasoning. As mentioned earlier, the inconsistency encountered in S' accounts for conflicting beliefs of a reasoner, and reflects his contradicting intentions or inadequate information about the real world. Separating the reality from one's beliefs allows us to tolerate this kind of inconsistency. For instance, this can be done by stating that in S', p and q are believed to be "inconsistent" while r is just "true."

To analyze the notion of inconsistency in APC, let us consider the set S of Example 1, and choose a semilattice, say, the one depicted in Figure 1(a). By the definitions in the previous section, S has several kinds of models, some of which are listed below:

m_1, where $p : \top$, $q : \top$, and $r : \mathbf{t}$ are true;
m_2, where $p : \top$, $q : \bot$, and $r : \mathbf{t}$ are true;
m_3, where $p : \top$, $q : \top$, and $r : \top$ are true;
m_4, where $p : \mathbf{t}$, $q : \top$, and $r : \mathbf{t}$ are true.

Actually, $p : \top$ means that the reasoner holds an inconsistent belief regarding p, so here we are dealing with *epistemic inconsistency*, or *e-inconsistency* for short. A model M of S is *e-inconsistent* if $M \models p : \top$ for some p. A set S of formulas is *e-inconsistent* if $S \models p : \top$, for some non-annotated atom p. A set of formulas is *ontologically inconsistent* (abbr., *o-inconsistent*) if it has no models in APC.

Ontologic inconsistency is similar to inconsistency in the ordinary predicate calculus: o-inconsistent sets of formulas have no meaning, since they entail everything. In contrast, e-inconsistent formulas do have models in APC; these models can be perceived as possible states of the reasoner's beliefs. However, if S is e-inconsistent then $S \models p : \top$, for some p, i.e. there is at least one inconsistent belief in every possible state of beliefs corresponding to S.

There is also a third possibility, the one illustrated in Example 1. A set of formulas S may be e-consistent, yet each of its models may be e-inconsistent. In other words, each of the possible states of reasoners beliefs characterized by S contains an inconsistent belief, although this belief may be different in different states (that is why S itself is e-consistent, according to our earlier definition). Corollary 3 of Section 6 reveals the connection between this latter situation and inconsistency in PC.

An examination of APC models suggests several useful notions. First, we observe that models may differ in the number of inconsistent beliefs (i.e., e-inconsistency) they contain. For instance, m_2 and m_4 contain less e-inconsistency than m_1, and the latter is "more consistent" than m_3. In addition, we may say that m_2 and m_4 are "minimal" among the listed models, in the sense that they contain the smallest (with respect to the semilattice ordering) beliefs about p, q, and r. On the other hand, m_2 and m_4 have similar amounts of inconsistency.

Formally, we say that a semantic structure I_1 is *more* (or equally) *e-consistent* than I_2 (denoted $I_2 \leq_\top I_1$) if and only if for every atom $p(t_1, \ldots, t_k)$, whenever $I_1 \models p(t_1, \ldots, t_k) : \top$ it is also the case that $I_2 \models p(t_1, \ldots, t_k) : \top$. I is *most e-consistent* in a class of semantic structures, if no semantic structure in this class is strictly more e-consistent than I (i.e., for every J in the class, $J \leq_\top I$ implies $I \leq_\top J$).

Similarly, we say that J_1 is *smaller* than J_2 (denoted $J_1 \leq J_2$) if and only if for every atom $q(t_1, \ldots, t_k)$, whenever $J_1 \models q(t_1, \ldots, t_k) : \mathbf{s}$ then also $J_2 \models q(t_1, \ldots, t_k) : \mathbf{s}$. A semantic structure

J is *minimal* in a given class of semantic structures if and only if this class contains no semantic structure strictly smaller than J.

A minimal model may be not most e-consistent, and vice versa, as illustrated by the following example.

Example 2: Consider $S = \{p : \mathbf{t},\ p : \mathbf{f} \vee\ q : \mathbf{t}\}$ and suppose p stands for "it rains" while q for "take umbrella." S has two minimal models: m_1 in which $p : \mathbf{t}$ and $q : \mathbf{t}$ are believed, and m_2 where $p : \top$ and $q : \bot$ are true. Here, m_1 is strictly more e-consistent than m_2, and also m_1 is more appealing to our intuition (i.e., "take umbrella when it rains"). The other model, m_2, corresponds to the following way of reasoning: "Although I was told that it rains and nobody has claimed otherwise, I remain sceptic making no decision about taking an umbrella." □

Sometimes, certain semilattice elements other than $\top$ may be also viewed as a kind of inconsistency. For instance, in Figure 1(b), **dt** (or **df**) stands for "concluded true (resp., false) by default." Then the intended meaning of $\mathbf{d}\top$ would be an "inconsistent default conclusion." Ginsberg [13] suggests that $\mathbf{d}\top$ should be distinguished from a stronger inconsistency represented by $\top$. In this situation, a reasoner may tend to minimize not just the number of $q : \top$ literals, but also other kinds of epistemic inconsistencies (such as $\mathbf{d}\top$).

Formally, let $\Delta \subseteq BSL$. We write $I_2 \leq_\Delta I_1$ (I_1 is more or equally e-consistent than I_2 relatively to Δ) if whenever $I_1 \models p : \lambda$ for some $\lambda \in \Delta$, then $I_2 \models p : \lambda$. This leads to the notion of most e-consistent models with respect to Δ. Several examples of the use of this notion in conjunction with the lattice of Figure 1(b) appear in Section 4.

3.2 Negation: Ontological vs. Epistemic

The negation, "$\neg$", introduced in Section 2 is of ontological nature, since $\neg p : \alpha$ is interpreted as the opposite to believing in p to the degree of α. However, at the epistemic level, it is also possible to talk about *epistemic negation*, denoted $\sim$, where $\sim p : \mathbf{t} = p : \mathbf{f}$ and $\sim p : \mathbf{f} = p : \mathbf{t}$.

Formally, *epistemic negation*, "$\sim$", is a semilattice isomorphism $BSL \rightarrow BSL$ such that

- it is symmetric, i.e. $\sim\sim \alpha = \alpha$ for all $\alpha \in BSL$; and
- $\sim \mathbf{t} = \mathbf{f}$, $\sim \mathbf{f} = \mathbf{t}$, $\sim \bot = \bot$, and $\sim \top = \top$ (the last two equalities actually follow from the symmetry of "$\sim$" and the monotonicity of semilattice homomorphisms).

It is extended to formulae as follows:

(i) $\sim p : \alpha \ \equiv\ p : \sim \alpha$

(ii) $\sim \neg\phi \ \equiv\ \neg \sim \phi$

(iii) $\sim (\phi \vee \psi) \ \equiv\ \sim \phi \wedge \sim \psi$
$\sim (\phi \wedge \psi) \ \equiv\ \sim \phi \vee \sim \psi$

(iv) $\sim (\forall X)\,\phi \ \equiv\ (\exists X) \sim \phi$
$\sim (\exists X)\,\phi \ \equiv\ (\forall X) \sim \phi$

Epistemic negation is uniquely defined for the lattice of Figure 1(a). For the BSL in Figure 1(b), we only need to postulate $\sim \mathbf{dt} = \mathbf{df}$, $\sim \mathbf{df} = \mathbf{dt}$, and $\sim \mathbf{d}\top = \mathbf{d}\top$.

3.3 Implication and Entailment: Ontological vs. Epistemic

In PC, implication $\psi \leftarrow \phi$ is defined as $\psi \vee \neg\phi$. In APC we have a choice between the *ontological implication*: $\psi \leftarrow \phi \ \equiv \psi \vee \neg\phi$, and the *epistemic implication*: $\psi \lessdot\!\!\sim \phi \ \equiv \psi \vee \sim \phi$.

The two kinds of implication differ significantly in their properties, especially in the way they propagate inconsistent beliefs.

Example 3: Consider $S = \{q : \mathbf{t}\ ,\ p : \mathbf{t} \leftarrow q : \mathbf{t}\}$. It is easy to see that $S \models p : \mathbf{t}$. Furthermore, even if q were an inconsistent belief, $p : \mathbf{t}$ would still follow: $\{q : \top,\ p : \mathbf{t} \leftarrow q : \mathbf{t}\} \models p : \mathbf{t}$. Thus, ontological implication allows us to draw conclusions from inconsistent beliefs. This corresponds to the following way of reasoning: Let ρ be a reasoner holding an inconsistent belief q, i.e. ρ believes q is true. Then the rule $p : \mathbf{t} \leftarrow q : \mathbf{t}$ says that ρ believes in p as long as it believes in q. Therefore, since ρ thinks q is true, it concludes that p is true, even though ρ also believes that q may be false. This kind of reasoning may be appropriate in situations when a reasoner has several sources of equally credulous information and wants to explore all consequences of this information.

For comparison, let us replace $\leftarrow$ with $\mathrel{<\!\!\sim}$ in the above example: $T = \{q : \mathbf{t}\,,\; p : \mathbf{t} \mathrel{<\!\!\sim} q : \mathbf{t}\}$. Surprisingly, it is no longer true that $T \models p : \mathbf{t}$. Indeed, $p : \mathbf{t} \mathrel{<\!\!\sim} q : \mathbf{t}$ is equivalent to $q : \mathbf{f} \vee p : \mathbf{t}$, and therefore there is a model I in which $I \models q : \top$, but only $I \models p : \bot$ (in particular, $I \not\models p : \mathbf{t}$). The intuitive reading in this latter case is: If ρ does not believe that q is false then ρ believes in p. Even though ρ thinks that q is true, it may still believe that q is false and so $p : \mathbf{t}$ is not concluded. □

We thus see that ontological implication is "eager": It permits drawing conclusions from inconsistent beliefs and has a *strong* modus ponens property, i.e., if $\gamma \leq \alpha$ then $\{p : \alpha,\; q : \beta \leftarrow p : \gamma\} \models q : \beta$. In contrast, epistemic implication is "overly cautious." Not only does it refuse to draw conclusions from inconsistent beliefs, but it also guards against the possibility that one of the premises (i.e. q in Example 3) *may* somehow turn out to be inconsistent after all. This excessive cautiousness can be countered with a kind of closed world assumption applied to e-inconsistency, which suggests to prefer models with the least amount of e-inconsistency.

In other words, in dealing with epistemic implication we may want to narrow our attention to the set of most e-consistent models. Logical entailment restricted to most e-consistent models will be called *epistemic entailment*, denoted $\mathrel{|\!\!\approx}$. A notion similar to most e-consistent models and epistemic entailment was independently proposed by Priest [24].

More generally, let $\Delta \in BSL$. We write $T \mathrel{|\!\!\approx}_\Delta \phi$ if and only if whenever M is a most e-consistent model of T with respect to Δ, $M \models \phi$ holds as well. Typically, inconsistency is minimized with respect to the set

$$\tilde{\Delta} = \{\alpha \mid \alpha \in BSL,\;\; \sim \alpha = \alpha\}.$$

Epistemic entailment with respect to the set $\tilde{\Delta}$ thus defined will be used throughout the rest of this paper. For the 4-valued BSL of Figure 1(a), $\mathrel{|\!\!\approx}_{\tilde{\Delta}}$ coincides with $\mathrel{|\!\!\approx}$ (i.e. with $\mathrel{|\!\!\approx}_{\{\top\}}$). In what follows, we will omit the subscript $\tilde{\Delta}$ and write $\mathrel{|\!\!\approx}$ instead of $\mathrel{|\!\!\approx}_{\tilde{\Delta}}$.

It is easy to see that in Example 3, $T \mathrel{|\!\!\approx} p : \mathbf{t}$. While countering the overly cautious behaviour of $\mathrel{<\!\!\sim}$, epistemic entailment still does not permit drawing conclusions from inconsistent premises. That is, if $T' = \{q : \top\,,\; p : \mathbf{t} \mathrel{<\!\!\sim} q : \mathbf{t}\}$, then $T' \not\mathrel{|\!\!\approx} p : \mathbf{t}$, since the model I of T' in which $I \models q : \top$ and $I \not\models p : \mathbf{t}$ hold, is most e-consistent.

Still, epistemic implication, as defined, does not adequately capture the intuition behind the notion of implication. The problem is in the inability to define the concept of a "canonic" possible world for a set of logical formulae. For instance, consider $C = \{q : \mathbf{t} \mathrel{<\!\!\sim} p : \mathbf{f}\}$, which is equivalent to $q : \mathbf{t} \vee p : \mathbf{t}$. If "$\mathrel{<\!\!\sim}$" were thought of as an implication of some sort, the most "natural" possible world for C would be the one where both p and q are unknown (have $\bot$ as a truth value). However, such a semantic structure would not be a model for C. The minimal most e-consistent models of C are the ones where either p or q is known to be true, which does not seem to be a suitable behaviour for implication in this case.

To overcome this difficulty, we modify the definition of epistemic implication as follows:

$$\phi \mathrel{<\!\!\sim}^{new} \psi \;\stackrel{\text{def}}{=}\; \phi \leftarrow (\psi \wedge \neg \sim \psi) \;\equiv\; \phi \vee \neg\psi \vee \sim \psi \;\equiv\; \phi \mathrel{<\!\!\sim} (\psi \wedge \neg \sim \psi).$$

From now on, we will abandon the old notion of epistemic implication and will use "$\mathrel{<\!\!\sim}$" to denote "$\mathrel{<\!\!\sim}^{new}$." The reader can verify that if ϕ is a disjunction and ψ is a conjunction of literals then $\phi \mathrel{<\!\!\sim} \psi$ is a clause, as should be expected of an implication. For instance, $\phi \mathrel{<\!\!\sim} (\psi \wedge \eta)$ is equivalent to $\phi \vee \neg\psi \vee \sim \psi \vee \neg\eta \vee \sim \eta$.

It is easy to verify that, when coupled with $\mathrel{|\!\!\approx}$, the new definition of implication $\mathrel{<\!\!\sim}$ is similar to the old one in that it has modus ponens and does not propagate inconsistency. However, it does not suffer from the drawback of the old definition. For instance, the semantic structure M

Paraphernalia	Properties
$\leftarrow$, $\models$	Draws conclusions from inconsistent beliefs; Logically omniscient, strong modus ponens.
$\leftarrow$, $\mid\!\approx$	Same as above.
$<\!\!\sim$, $\models$	Overly cautious, no modus ponens, non-omniscient.
$<\!\!\sim$, $\mid\!\approx$	No conclusions drawn from inconsistent beliefs; Limited modus ponens.

Table 1: Summary of APC Entailments

that assigns $\perp$ to every atom would now be a model of $C = \{q : \mathbf{t} <\!\!\sim p : \mathbf{t}\}$, since under the new definition, C is equivalent to $q : \mathbf{t} \vee \neg p : \mathbf{t} \vee p : \mathbf{f}$, and $M \models \neg p : \mathbf{t}$.

However, unlike "$\leftarrow$", only a *limited* version of modus ponens holds for "$<\!\!\sim$." For instance, for the lattice of Figure 1(b), $\{p : \mathbf{t},\ q : \mathbf{t} <\!\!\sim p : \mathbf{dt}\} \not\mid\!\approx q : \mathbf{t}$, while $\{p : \mathbf{t},\ q : \mathbf{t} \leftarrow p : \mathbf{dt}\} \models q : \mathbf{t}$. The epistemic rule $q : \mathbf{t} <\!\!\sim p : \mathbf{dt}$ cannot be applied to $p : \mathbf{t}$ because $\mathbf{t} > \ \sim \mathbf{dt} = \mathbf{df}$, where "$>$" is the BSL-ordering. In general, when $\gamma \geq \alpha$, then $\{p : \gamma,\ q : \beta <\!\!\sim p : \alpha\} \not\mid\!\approx q : \beta$ if and only if $\gamma \geq \mathtt{lub}(\alpha, \sim \alpha)$ (notice that $\mathtt{lub}(\alpha, \sim \alpha) \in \tilde{\Delta}$ and it is the smallest inconsistent degree of belief located above α in BSL).

APC with its two kinds of negation and implication provides a rich framework for modeling various situations where inconsistency may arise. For instance, Belnap [2] takes a stand that no conclusion should be based on inconsistent beliefs, and defines implication accordingly. As we have seen, epistemic implication represents that view. On the other hand, Touretzki, Horty, and Thomason [32] argued that in some situations inconsistency may be propagated through implications, which corresponds to the ontological implication in APC.

The distinction between the ontological and epistemic versions of negation and implication sheds additional light on the logics used in [33, 30, 3, 15] from which APC was gradually evolving. It turns out that those logics use the ontological implication, but the negation is epistemic ([33] does not consider negation). They also use special kinds of lattices and [33, 30, 3] are subsumed both by [15] and APC. On the other hand, [15] cannot be directly compared to APC, since the former uses quantified variables and evaluable monotonic functions over BSL. We can summarize our observations in Table 1.

4 Nonmonotonic Reasoning with Contradictory Beliefs

As is well known, nonmonotonic logics (e.g. [20, 27]) significantly extend the power of PC to model various aspects of human reasoning. Circumscription, Default Logic and other reasoning schemes can be built on top of APC along the same lines. However, even without using circumscription or defaults, APC yields more informative results than PC does.

Example 4: Consider the following set of clauses $\mathbf{D}$:

$$flies(X) : \mathbf{t} \leftarrow bird(X) : \mathbf{t}$$
$$\sim flies(X) : \mathbf{t} \leftarrow penguin(X) : \mathbf{t}$$
$$bird(tweety) : \mathbf{t}$$
$$bird(fred) : \mathbf{t}.$$

Let $\mathbf{D}'$ denote $\mathbf{D}$ with the denotations stripped off (and $\sim$ replaced with $\neg$), i.e. $\mathbf{D}'$ is a set of PC-clauses. Both PC and APC conclude that $flies(tweety)$ and $flies(fred)$ are true. However, adding $penguin(tweety)$ to $\mathbf{D}'$ makes it PC-inconsistent, precluding any informative conclusion. In contrast, in APC, adding $penguin(tweety) : \mathbf{t}$ to $\mathbf{D}$ leads to the inconsistency regarding $flies(tweety)$ (i.e. $flies(tweety) : \top$), but still infers $flies(fred) : \mathbf{t}$ rather than, say, $flies(fred) : \mathbf{f}$. $\Box$

Although recent advances in Logic Programming [21] provided an adequate semantics for negation in rule bodies, still negation is disallowed in rule heads. In particular, rather natural situations such as that of Example 4 do not have a straightforward representation. In the framework of APC such programs can be given a natural and computationally realizable semantics. In addition, it now becomes possible to exploit the two kinds of implications found in APC.

In Logic Programming, negation in rule bodies is normally treated via a form of closed world assumption. An appropriate semantics for this treatment was recently proposed in [1, 25, 18]. Conclusions made according to such semantics may not always be logically valid, but rather have the status of defaults. Several researchers argued that preserving the distinction between true and default facts may be useful [13, 23]. We shall see in the examples below that in APC keeping track of the defaults can be facilitated by using special semilattices, such as the one in Figure 1(b). In general, the choice of a suitable semilattice is problem-dependent and may be quite different from that of Figure 1(b).

A number of recent proposals allow negated literals in rule heads [10, 3, 15]. However, all these consider the epistemic negation only. In addition, the treatment of negation in [10, 3]. is monotonic, which precludes default reasoning. Recently Kifer and Li [15] have shown that for a special type of lattices, epistemic negation can be also treated non-monotonically. This result can be extended to the case when BSL is a bilattice with conflation. Allowing both types of negation and non-monotonicity — epistemic and ontological — results in a very rich logic programming language. Many of the above issues are illustrated in the examples that follow.

Example 5: Let us refine the Tweety/penguin example as follows:

(i) $flies(X)\ :\ \mathbf{dt}\leftarrow bird(X)\ :\ \mathbf{dt}$

(ii) $flies(X)\ :\ \mathbf{df}\leftarrow penguin(X)\ :\ \mathbf{dt}$

(iii) $flies(X)\ :\ \mathbf{df} \mathrel{<\!\!\sim} wounded(X)\ :\ \mathbf{dt}$
$flies(X)\ :\ \mathbf{df} \mathrel{<\!\!\sim} wounded(X)\ :\ \mathbf{t}$

(iv) $flies(X)\ :\ \mathbf{f} \mathrel{<\!\!\sim} penguin(X)\ :\ \mathbf{t}$.

The first rule states that whenever there is an evidence (perhaps obtained via reasoning by default) that X is a bird, it is viewed as a sufficient reason to make a default conclusion that X flies. The next rule states that the default belief "X is a penguin" suffices to further conclude that X cannot fly; Rule (ii) assigns this latter conclusion the status of a default. Clauses in (iii) tell us that when we think X is wounded (say, if we hear a shot, knowing that the hunter is quite accurate) we assume X cannot fly. Again, we assign this conclusion only a default status because we may not be sure that X cannot fly when it is wounded, even more so if we are not at all sure that X is wounded. Rule (iv) says that if X is known to be a penguin then conclude that X cannot fly.

It is not accidental that the last three clauses contain epistemic implication. The rationale here is that we do not want to propagate inconsistency through (iii) and (iv) (recall from Section 3.3 that under the epistemic entailment "$\mathrel{|\!\!\approx}$", the implication "$\mathrel{<\!\!\sim}$" differs from "$\leftarrow$" in that inconsistent conclusions are not propagated through "$\mathrel{<\!\!\sim}$"). In (iii), if our information about X being wounded is contradictory (of status $\mathbf{d}\top$ or $\top$), we do not want to conclude that X cannot fly. We would like to be more cautious here than in Rule (ii) because being wounded is not a sufficient reason for not flying (as opposed to being a penguin) — even more so if there is a counter-evidence suggesting that X is not wounded in the first place. In Rule (iv), we use epistemic implication because in the presence of inconsistency of type $\top$, i.e., when there is a possibility that X may not be a penguin ($penguin(X)\ :\ \mathbf{f}$), we do not rush to conclude that X cannot fly and prefer to rely on other rules (Clauses (i) - (iii)).

To extract the desired behaviour from our program we will consider it under the epistemic entailment "$\mathrel{|\!\!\approx}$" (i.e. $\mathrel{|\!\!\approx}_{\tilde{\Delta}}$, where $\tilde{\Delta} = \{\bot,\ \mathbf{d}\top,\ \top\}$ — see Section 3.3).
Suppose that initially it is known that:
(v) $bird(tweety)\ :\ \mathbf{t}$.
Then, by (i), we can conclude that Tweety possibly flies: $flies(tweety)\ :\ \mathbf{dt}$. However, if later we somehow arrive at a default conclusion

(vi) $penguin(tweety)\ :\ \mathbf{dt}$,
Rule (ii) would lead us to conclude $flies(tweety)\ :\ \mathbf{df}$. Together with $flies(tweety)\ :\ \mathbf{dt}$ this yields $flies(tweety)\ :\mathbf{d}\top$, as expected.

It was our intention here not to impose priorities on the default evidences "*bird*" and "*penguin*." Prioritizing evidences is not hard, however. For instance, we could use a more elaborate semilattice, where several levels of defaults are represented (e.g., $\mathbf{dt}_1$, $\mathbf{df}_1$, $\mathbf{d}\top_1$, $\mathbf{dt}_2$, $\mathbf{df}_2$, $\mathbf{d}\top_2$, etc., such that $\mathbf{dt}_i$, $\mathbf{df}_i < \mathbf{d}\top_i$ and $@_i < \#_{i+1}$, where @ and # are either $\mathbf{dt}$, $\mathbf{df}$, or $\mathbf{d}\top$), and use different-strength default truth values in Clauses (i) and (ii). Now, suppose that after making further observations we upgrade Tweety's status to
(vii) $penguin(tweety)\ :\ \mathbf{t}$.
This allows us to execute Rule (iv), yielding $flies(tweety)\ :\ \mathbf{f}$. Since $\mathbf{d}\top < \mathbf{f}$ in the lattice ordering of Figure 1(b), we now conclude $flies(tweety)\ :\ \mathbf{f}$, as intended.

Even more interesting is to see what happens when the information about Tweety being a penguin gets garbled so that we become unsure about Tweety's zoological classification. Let us assume for the sake of an example that new rumors have it that Tweety is not a penguin but an undercover CIA agent. We may get confused by all this and upgrade (vii) to
(viii) $penguin(tweety)\ :\ \top$.
Since under the semantics defined by "$\models\!\!\!\approx$" Rule (iv) does not propagate inconsistent beliefs, we no longer can derive $flies(tweety)\ :\ \mathbf{f}$. Therefore, only the rules (i) and (ii) apply and we fall back on the previous conclusion $flies(tweety)\ :\mathbf{d}\top$, i.e. we become uncertain whether Tweety flies or not.

Now, let us turn to Tweety's mate, Fred. Suppose that in addition to being a bird
(v)' $bird(fred)\ :\ \mathbf{t}$
it is rumored that Fred is wounded:
(vi)' $wounded(fred)\ :\ \mathbf{dt}$.
Rules (i) and (iii) lead us to conclude $flies(fred)\ :\ \mathbf{d}\top$. To see why Rule (iii) is enabled here, observe that it is equivalent to $flies(fred)\ :\ \mathbf{df} \vee \neg wounded(fred)\ :\ \mathbf{dt} \vee wounded(fred)\ :\ \mathbf{df}$. There are two classes of models for this clause and for (vi)'. In one, $wounded(fred)\ :\ \mathbf{d}\top$ but not necessarily $flies(fred)\ :\ \mathbf{df}$ holds. In another, $wounded(fred)\ :\ \mathbf{d}\top$ does not hold, but then necessarily $flies(fred)\ :\ \mathbf{df}$ and $wounded(fred)\ :\ \mathbf{dt}$ must be true. Clearly, since $\mathbf{d}\top \in \Delta$, only the latter class contains most e-consistent models relatively to Δ. Hence

$$\{\text{Clauses (iii)},\ wounded(fred)\ :\ \mathbf{dt}\}\ \models\!\!\!\approx\ flies(fred)\ :\ \mathbf{df}.$$

If, however, subsequent rumors bring that the earlier rumors were wrong, we may become uncertain about Fred's wounds and assert
(vii)' $wounded(fred)\ :\ \mathbf{d}\top$.
Similarly to the Tweety's case, even though this inconsistency has a default status, it disables the rules in (iii), since

$$\{\text{Clauses (iii)},\ wounded(fred)\ :\ \mathbf{d}\top\}\ \not\models\!\!\!\approx\ flies(fred)\ :\ \mathbf{df}.$$

(Given $wounded(fred)\ :\ \mathbf{d}\top$ and $flies(fred)\ :\ \mathbf{df} \vee \neg wounded(fred)\ :\ \mathbf{dt} \vee wounded(fred)\ :\ \mathbf{df}$, any model where $wounded(fred)\ :\ \mathbf{d}\top$ holds but $flies(fred)\ :\ \bot$ does not, is always $\leq_{\hat{\Delta}}$-smaller than those models where, in addition, $flies(fred)\ :\ \mathbf{df}$ is true.) Therefore, only Rule (i) remains active, yielding a default conclusion that Fred can fly: $flies(fred)\ :\ \mathbf{dt}$. This is precisely the desired effect since, as explained earlier, we do not want to draw conclusions via Rules (iii) from inconsistent beliefs about $wounded(X)$.

Notice that so far only the first of the rules in (iii) was employed. The second rule there is needed in case we further upgrade the available information to $wounded(fred)\ :\ \mathbf{t}$. The first rule in (iii) does not apply to $wounded(fred)\ :\ \mathbf{t}$ because of the limited modus ponens. (Note that if the information is upgraded to $wounded(fred)\ :\ \mathbf{f}$, none of the rules in (iii) applies.)

We could continue this story and expand into the following situation:

(ix) $cage(X)\ :\ \mathbf{dt} \leftarrow flies(X)\ :\ \mathbf{dt}$

(x) $cage(X)\ :\ \mathbf{f} \lessdot\!\sim flies(X)\ :\ \mathbf{f}$.

Rule (ix) says that if there is a slightest chance that X may fly away — put it into a cage. Moreover, X must be caged even if our information is inconsistent merely at the default level. In contrast, if we believe that X cannot fly, Rule (x) suggests not to imprison X. Epistemic entailment in Rule (x) indicates that if we become uncertain of whether X flies or not, we would not want to use Rule (x) but would rather rely on other rules.

For instance, having heard about Dumbo's vast ears, we may conclude:
(xi) $flies(dumbo)\ :\ \mathbf{dt}$,
prompting us to cage Dumbo, by default: $cage(X)\ :\ \mathbf{dt}$. Later we may find out that Dumbo is an elephant and, knowing that elephants cannot fly, decide:
(xii) $flies(dumbo)\ :\ \mathbf{f}$.
This activates Rule (x), yielding $cage(dumbo)\ :\ \mathbf{f}$. Since $\mathbf{dt} \leq \mathbf{f}$ in the lattice of Figure 1(b), $cage(dumbo)\ :\ \mathbf{f}$ prevails and Dumbo is released. However, after seeing the cartoon, our beliefs may be in disarray again, for we did see Dumbo flying after all! So, we assert:
(xiii) $flies(dumbo)\ :\ \top$.
Now, rule (x) is disabled, for it does not propagate inconsistent beliefs. Therefore, we fall back on Rule (ix), concluding $cage(X)\ :\ \mathbf{dt}$. Poor Dumbo is caged again, as we decide not to take chances. □

The next example is from [13]:
Example 6: Consider the situation described by the following set of formulae $\mathbf{D}$:

$$\begin{array}{l}
bird(Tweety)\ :\ \mathbf{t}\\
flies(X)\ :\ \mathbf{dt} \lessdot\!\sim bird(X)\ :\ \mathbf{dt}\\
flies(X)\ :\ \mathbf{dt} \lessdot\!\sim bird(X)\ :\ \mathbf{t}\\
\sim acrophobic(X)\ :\ \mathbf{dt} \lessdot\!\sim bird(X)\ :\ \mathbf{dt}\\
\sim acrophobic(X)\ :\ \mathbf{dt} \lessdot\!\sim bird(X)\ :\ \mathbf{t}.
\end{array}$$

Under the semantics determined by $\approx\!\!|$ (i.e. $\approx\!\!|_{\tilde{\Delta}}$, where $\tilde{\Delta} = \{\top,\ \mathbf{d}\top,\ \bot\}$) we obtain that $\mathbf{D}\ |\!\approx\ flies(Tweety)\ :\ \mathbf{dt}$ and $\mathbf{D}\ |\!\approx\ acrophobic(Tweety)\ :\ \mathbf{df}$ hold (the latter holds since $\sim acrophobic(Tweety)\ :\ \mathbf{dt} \equiv acrophobic(Tweety)\ :\ \mathbf{df}$), as expected.

This should be contrasted with [13] where two distinct possible worlds are sanctioned: One corresponds to our result above while in another Tweety cannot fly (i.e., $flies(Tweety)\ :\ \mathbf{f}$) and it is unknown whether he is acrophobic (i.e. $acrophobic(Tweety)\ :\ \bot$). To eliminate the unwanted (second) possible world, Ginsberg [13] needs to employ an additional notion, which he calls "boundedness." □

The next example is known as Nixon's Diamond and, again, we will contrast the APC-representation to that of [13].

and a Quaker. All Quakers are supposedly doves (but there might be exceptions) and all republicans are hawks, by default. It is also assumed that hawks and doves are incompatible political views. In APC this can be represented as follows:

(i) $quaker(nixon)\ :\ \mathbf{t}$

(ii) $republican(nixon)\ :\ \mathbf{t}$

(iii) $hawk(X)\ :\ \mathbf{dt} \lessdot\!\sim republican(X)\ :\ \mathbf{dt}$
$hawk(X)\ :\ \mathbf{dt} \lessdot\!\sim republican(X)\ :\ \mathbf{t}$

(iv) $dove(X)\ :\ \mathbf{dt} \lessdot\!\sim quaker(X)\ :\ \mathbf{dt}$
$dove(X)\ :\ \mathbf{dt} \lessdot\!\sim quaker(X)\ :\ \mathbf{t}$

(v) $\sim dove(X)\ :\ \mathbf{dt} \lessdot\!\sim hawk(X)\ :\ \mathbf{dt}$
$\sim dove(X)\ :\ \mathbf{dt} \lessdot\!\sim hawk(X)\ :\ \mathbf{t}$

(vi) $\sim hawk(X)\ :\ \mathbf{dt} \lessdot\!\sim dove(X)\ :\ \mathbf{dt}$
$\sim hawk(X)\ :\ \mathbf{dt} \lessdot\!\sim dove(X)\ :\ \mathbf{t}$.

Let us denote the above set of clauses by $\mathbf{D}$. Using the facts (i) and (ii) about Nixon and applying Clauses (iii) — (v), we derive $hawk(nixon) : \mathbf{dt}$ and $dove(nixon) : \mathbf{d}\top$. Clauses in (vi) cannot be applied to $dove(nixon) : \mathbf{d}\top$ since $\not\leftarrow$ does not propagate inconsistency. The reader can verify that $M_1 = \{quaker(nixon) : \mathbf{t}, republican(nixon) : \mathbf{t}, hawk(nixon) : \mathbf{dt}, dove(nixon) : \mathbf{d}\top\}$ is a most e-consistent model of $\mathbf{D}$ relatively to $\tilde{\Delta}$.

On the other hand, if we apply Clauses (iii), (iv), and (vi) then $hawk(nixon) : \mathbf{d}\top$ and $dove(nixon) : \mathbf{dt}$ are derived. Clause (v) cannot be applied since we have contradictory beliefs about Nixon as a hawk. Therefore, $M_2 = \{quaker(nixon) : \mathbf{t}, republican(nixon) : \mathbf{t}, hawk(nixon) : \mathbf{d}\top, dove(nixon) : \mathbf{dt}\}$ is another most e-consistent model of $\mathbf{D}$ with respect to $\tilde{\Delta}$. □

It is seen easily that M_1 and M_2 above are the only most $\tilde{\Delta}$-consistent models of $\mathbf{D}$. In M_1, Nixon is a hawk, by default, while the default conclusions regarding Nixon being a dove contradict each other; in M_2, the conclusions $dove(nixon)$ and $hawk(nixon)$ exchange status. This is an improvement over [13] where an analogue of Example 7 has only one possible world in which both $dove(nixon)$ and $hawk(nixon)$ have the status $\mathbf{d}\top$.

5 Proof Theory

This section presents a sound and complete proof theory for the monotonic semantics of APC — the one that is determined by the logical entailment relation "$\models$". Regarding the nonmonotonic semantics represented by the relation "$\models\!\approx$", it is not hard to show that, as usual with such theories, it has no complete proof theory. Computational aspects of "$\models\!\approx$" are beyond the scope of this paper; we only mention that computation under "$\models\!\approx$" is closely related to truth-maintenance.

An appealing feature of APC is that the standard techniques, such as Skolemization, Herbrand's theorem, refutational proof procedure, and so on are still applicable. We assume that the reader is familiar with these standard notions of first order logic, and present their counterparts in APC.

The Skolemization procedure in APC is essentially identical to that of PC, and we obtain the following result whose proof is identical to that of Skolem theorem in predicate calculus [4].

Theorem 1 (cf. Skolem Theorem): Let S be a set of formulas, and ϕ — a formula. Let $\hat{S}$, $\hat{\phi}$ denote a Skolemization of S and ϕ, respectively. Then $S \cup \{\neg\phi\}$ is o-inconsistent if and only if so is $\hat{S} \cup \{\neg\hat{\phi}\}$. □

As in the classic case, Skolem Theorem allows us to replace every formula by a set of clauses such that the original formula is unsatisfiable if and only if so is the corresponding set of clauses. A *clause* is a set of APC literals that is thought of as a disjunction. In particular, this means that clauses contain no duplicate literals. So, although in the following we will often write clauses as disjunctions of literals, it is important to keep in mind that these disjunctions are duplicate-free.

Given a language, L, the *Herbrand universe* in APC is the same as that in PC. The *Herbrand base* is the set of all annotated ground atoms of L. A *Herbrand interpretation*, I, is a subset of the Herbrand base such that for every ground literal p (without the annotation) there is $\mathbf{r} \in BSL$ (depending only on p) such that $p : \mathbf{s} \in I$ if and only if $\mathbf{s} \leq \mathbf{r}$. We chose this variant of the definition of Herbrand interpretations in keeping with the logic programming tradition that defines such interpretations as sets of "true things." Usually we will describe Herbrand interpretations by specifying only the highest annotation associated with each atom, e.g. $I = \{p : \mathbf{s}, q : \mathbf{r}\}$ should be understood as a shorthand for $\{p : \mathbf{u}, q : \mathbf{v} \mid \mathbf{u} \leq \mathbf{s} \text{ and } \mathbf{v} \leq \mathbf{r}\}$.

Proof of the following lemma is essentially identical to the proof of the corresponding result in predicate calculus [4].

Lemma 1: A set of clauses (in APC) has a model if and only if it has a Herbrand model. □

In general, Herbrand's theorem does not hold for APC as shown in the next example. Therefore, APC does not have the compactness property. Theorem 2 below shows, however, that under certain restrictions this property still applies. In Section 8 we propose an alternative (and a slightly more complex) semantics for APC that does have the compactness property.

Example 8: Consider a belief semilattice obtained from the 4-valued lattice (Figure 1a) by filling in the edges of the diamond with a continuum of values. For definiteness, assume that the continuum corresponding to the edge $< \bot, \mathbf{t} >$ is linearly ordered between $\bot$ and $\mathbf{t}$ and has the form $\{\mathbf{t}_r \mid r \in [0,1]\}$, where $\mathbf{t}_0 = \bot$ and $\mathbf{t}_1 = \mathbf{t}$. Let S consist of $\neg p : \mathbf{t}$ and the set $\{p : \mathbf{t}_r \mid 0 < r < 1\}$.

Clearly S is unsatisfiable, for in every model of S $p : \mathbf{t}_1$ ($\equiv p : \mathbf{t}$) must be true. However, every finite subset of S is satisfiable, since $\mathbf{t}$ is not a limit of any finite subset of $\{\mathbf{t}_r \mid 0 < r < 1\}$. □

Nevertheless, Herbrand's theorem holds in several important special cases, which include logic programs and the situations when BSL is finite.

Theorem 2 (cf. Herbrand's theorem): Consider a set of possibly nonground clauses, S, that involves only a finite number of different belief annotations (this condition is always satisfied when BSL is finite or when S itself is finite, e.g. a logic program). Then S is o-inconsistent if and only if so is some finite subset of its ground instances.

Proof: See Appendix B. □

The notion of substitution in APC is the same as in PC. There is a slight difference in unification. A pair of atoms, $p(t_1, \ldots, t_k) : \mathbf{s}$ and $p(t'_1, \ldots, t'_k) : \mathbf{r}$, is *unifiable* if and only if there exists a unifier of $p(t_1, \ldots, t_k)$ and $p(t'_1, \ldots, t'_k)$. Most general unifiers (mgu's) are, again, defined in the standard way [4].

Binary resolution is defined as follows. Let $p(\bar{t}) : \mathbf{s} \vee \phi$ and $\neg p(\bar{t'}) : \mathbf{r} \vee \psi$ be a pair of clauses not sharing common variables, such that $\theta = \mathtt{mgu}(p(\bar{t}), p(\bar{t'}))$ and $\mathbf{s} \geq \mathbf{r}$ (note the asymmetry!). Then their *binary resolvent* is $(\phi \vee \psi)\theta$.

Next, consider a clause $\psi = p(\bar{t}_1) : \mathbf{s}_1 \vee \ldots \vee p(\bar{t}_k) : \mathbf{s}_k \vee \phi$ or $\psi = \neg p(\bar{t}_1) : \mathbf{s}_1 \vee \ldots \vee \neg p(\bar{t}_k) : \mathbf{s}_k \vee \phi$. Let $\theta = \mathtt{mgu}(p(\bar{t}_1), \ldots, p(\bar{t}_k))$. Then $\psi\theta$ is a *factor*[3] of ψ.

As in predicate calculus, binary resolution can be combined with factorization to yield a single powerful rule: Let $\alpha = p(\bar{t}_1) : \mathbf{s}_1 \vee \ldots \vee p(\bar{t}_k) : \mathbf{s}_k \vee \phi$, $\beta = \neg p(\bar{t'}_1) : \mathbf{r}_1 \vee \ldots \vee \neg p(\bar{t'}_n) : \mathbf{r}_n \vee \phi$ be a pair of clauses and $\theta = \mathtt{mgu}(p(\bar{t}_1), \ldots, p(\bar{t}_k), p(\bar{t'}_1), \ldots, p(\bar{t'}_n))$. Suppose further that for all i, j, $\mathbf{s}_i \geq \mathbf{r}_j$. Then $(\phi \vee \psi)\theta$ is a (full) resolvent of α and β. However, we prefer to use binary resolution with factorization rather than the combined resolution rule.

Notice that unlike in PC, there is an asymmetry in the definition of resolution, since the positive literal to be resolved upon must not be believed in less strongly than the negative one. More important, unlike in PC, resolution and factorization alone are not complete as derivation rules. For instance, these rules do not suffice to derive an empty clause from S=$p : \mathbf{f}$, $p : \mathbf{t}$, $\neg p : \top$, although this set is unsatisfiable. Indeed, to be a model of S, I has to satisfy $p : \mathbf{f}$ and $p : \mathbf{t}$. By the definition of Herbrand models given above, I must also satisfy $p : \mathtt{lub}(\mathbf{f}, \mathbf{t}) \equiv p : \top$. But then I falsifies $\neg p : \top$. Furthermore, for any atom p, $p : \bot$ is a tautology in APC, yet it is not derivable by the above rules. To complete the picture we introduce two more rules, called *reduction* and *elimination*.

Given a pair of clauses $p(\bar{t}) : \mathbf{s} \vee \phi$ and $p(\bar{t'}) : \mathbf{r} \vee \psi$ such that $\theta = \mathtt{mgu}(p(\bar{t}), p(\bar{t'}))$, their *reduction* is the clause $p(\bar{t})\theta : \mathtt{lub}(\mathbf{s}, \mathbf{r}) \vee \phi\theta \vee \psi\theta$. Coming back to the example in the previous paragraph, we can reduce $p : \mathbf{f}$ and $p : \mathbf{t}$, producing $p : \top$. The latter literal can then be resolved with $\neg p : \top$, yielding the empty clause. The *elimination* rule, being applied to a clause C, simply removes all literals in C of the form $\neg p(\bar{t}) : \bot$. The resulting clause is called an *eliminant* of C. Notice that the elimination rules removes literals, not clauses. So, for instance, being applied to the clause $\neg p : \bot$ (recall: clauses are sets) it will yield an empty clause and being applied to $\neg p : \bot \vee C'$ it will yield C'.

Lemma 2: Resolution, factorization, elimination, and reduction are sound derivation rules.

Proof: Trivial. □

Given a set of clauses S, its *refutation* is a derivation from S of an empty clause using resolution, factorization, elimination, and reduction inference rules.

[3]If BSL were a lattice, we could then define the factor of ψ as $p(\bar{t}_1)\theta : \mathtt{glb}_{i=1}^{k}(\mathbf{s}_i) \vee \phi\theta$ or $\neg p(\bar{t}_1)\theta : \mathtt{lub}_{i=1}^{k}(\mathbf{s}_i) \vee \phi\theta$, depending on whether positive or negative literals are being factored.

Theorem 3: Refutation is a sound and complete procedure for testing o-inconsistency of sets of clauses involving only a finite number of different belief constants. That is, if S is such a set then it is o-inconsistent if and only if there is a refutation of S.
Proof: See Appendix B. $\Box$

It follows from Theorem 3 that e-inconsistency is also semi-decidable.
Corollary 1: It is semi-decidable whether a set of APC formulas is e-inconsistent, i.e., there is an algorithm that terminates with an answer "yes" if and only if S is e-inconsistent.
Proof: We first prove a weaker result and then explain what has to be done for the general case: Consider a specific belief p. Clearly, $S \models p : \top$ if and only if $S \cup \{\neg p : \top\}$ is o-inconsistent, if and only if (in view of Theorem 3) the latter set has a refutation.

For the general case (when the belief p to be tested is not specified), we have to show that S entails *some* inconsistent belief and if it does so then to find one. The most intuitive way to prove this is to consider an annotated version of HiLog — a recently introduced higher-order extension of PC [5]. What makes HiLog so useful here is the fact that its variables may range over atomic formulas of PC. Thus, to find out whether S implies $p : \top$ for some unspecified belief p, we only need to verify that $S \cup \{\neg X : \top\}$ has a refutation, where X is a variable. It is shown in [5] that HiLog has a sound and complete resolution-based refutation procedure. The reader who is familiar with HiLog should have no problem to see that the theory developed so far for APC carries over in a straightforward way to the annotated HiLog. In particular, annotated HiLog has a sound and complete proof theory. Hence, $S \cup \{\neg X : \top\}$ can be refuted if and only if $S \models (\exists X)X : \top$, if and only if there is an atom p such that $S \models p : \top$. $\Box$
Example 9: Consider the following set of clauses:

(i)	$p : \top \vee q : \top \vee \neg r : \bot$;
(ii)	$p : \mathbf{t} \vee \neg q : \mathbf{t}$;
(iii)	$\neg p : \top$;
(iv)	$p : \mathbf{f}$.

The following refutation illustrates the basic concepts:

(v)	$p : \top \vee q : \top$	(eliminant of (i));
(vi)	$p : \top \vee p : \mathbf{t}$	(resolvent of (v) and (ii));
(vii)	$p : \mathbf{t}$	(resolvent of (iii) and (vi));
(viii)	$p : \top$	(reduction of (vii) and (iv));
(ix)	empty clause	(resolvent of (viii) and (iii)). $\Box$

The factorization rule is needed in APC in situations similar to those where it is needed in PC, e.g. to refute $\{p(X) : \mathbf{s} \vee p(Y) : \mathbf{s}, \ \neg p(V) : \mathbf{s} \vee \neg p(W) : \mathbf{s}\}$.

6 Relationship to Predicate Calculus

As mentioned earlier, the purpose of APC is to cope with inconsistency. For consistent systems it should be expected to yield essentially the same consequences as the standard predicate calculus. In this section we present two different natural embeddings of PC into APC and discuss their properties.

The first, *epistemic* embedding, Ξ_{epi}, views formulas of PC as *beliefs* and interprets negation, $\neg p$, in a rather restricted sense — as a belief in the falsehood of p. Formally, $\Xi_{epi}(p(...)) = p(...) : \mathbf{t}$ and $\Xi_{epi}(\neg p(...)) = p(...) : \mathbf{f}$ ($\equiv \sim p(...) : \mathbf{t}$). The second, *ontologic* embedding, Ξ_{ont}, interprets negation $\neg p$, in a much stronger sense — as not believing that p is true. Thus, in contrast to the epistemic embedding, Ξ_{ont} views S not as a collection of beliefs, but ontologically, as knowledge about what the reasoner believes in or does not. Formally, $\Xi_{ont}(p(...)) = p(...) : \mathbf{t}$ and $\Xi_{ont}(\neg p(...)) = \neg p(...) : \mathbf{t}$. Both embeddings extend to formulas homomorphically, by commuting with quantifiers, $\vee$, and $\wedge$.

Closely related to these embeddings is a mapping Ξ from semantic structures for PC into the set of semantic structures for APC. Let $I =< D, I_F, I_P >$ be a semantic structure of PC, where I_P interprets predicate symbols as relations. Then $\Xi(I) = < D, I_F, I_{P'} >$, i.e. it shares with I the domain D and the interpretation of function symbols, I_F; $I_{P'}$ is defined as follows: for each predicate symbol p and every tuple $a_1, ..., a_n \in D$, $I_{P'}(p)(a_1, ..., a_n) = \mathbf{t}$ if and only if $I_P(p)(a_1, ..., a_n) = true$; $I_{P'}(p)(a_1, ..., a_n) = \mathbf{f}$ if and only if $I_P(p)(a_1, ..., a_n) = false$.

As seen from the definition, Ξ yields a special kind of semantic structures for APC — those where each atom is known to be either true or false (but not both). We say that an APC semantic structure, $I = < D, I_F, I_P >$, is a *tf-structure* if for each n-ary predicate symbol, p, and every tuple $a_1, ..., a_n \in D$, either $I_P(p)(a_1, \ldots, a_n) = \mathbf{t}$ or $I_P(p)(a_1, \ldots, a_n) = \mathbf{f}$. For Herbrand interpretations this means that for every ground atom p, either $p : \mathbf{t} \in I$, or $p : \mathbf{f} \in I$, but not both.

It is easy to verify that for every PC semantic structure I, $\Xi(I)$ is a tf-structure. Furthermore, for a given PC language L and a semilattice BSL, let L_{BSL} denote the corresponding language of APC. Then Ξ is a 1-1 mapping from semantic structures of L onto the set of tf-structures of L_{BSL}. The inverse mapping $\Upsilon : \{\text{tf-structures of } L_{BSL}\} \longrightarrow \{\text{semantic structures of } L\}$ is defined as follows: Let $I' = < D, I_F, I'_P >$ be a tf-structure of L_{BSL}. Then (1) $\Upsilon(I') = I$, where $I = < D, I_F, I_P >$ interprets constants and function symbols of L the same way as I'; and (2) $I_P(p(a_1, ..., a_n)) = true$ if and only if $I_{P'}(p)(a_1, ..., a_n) = \mathbf{t}$; $I_P(p(a_1, ..., a_n)) = false$ if and only if $I_{P'}(p)(a_1, ..., a_n) = \mathbf{f}$. This mapping is well defined, since I' is a tf-structure.

Theorem 4: Let S be a set of formulas in PC. Let $\mathbf{M}(S)$ denote the set of models of S, $\mathbf{M}^{ont}_{tf}(S)$ be the set of all tf-models of $\Xi_{ont}(S)$, and let $\mathbf{M}^{epi}_{tf}(S)$ be the same for $\Xi_{epi}(S)$. Then

1. $\mathbf{M}^{ont}_{tf}(S) = \mathbf{M}^{epi}_{tf}(S)$ (hereafter denoted just as $\mathbf{M}_{tf}(S)$).
2. $\Xi : \mathbf{M}(S) \rightarrow \mathbf{M}_{tf}(S)$ is a 1-1 mapping onto $\mathbf{M}_{tf}(S)$.
3. $S \models \phi$ if and only if $M \models \Xi_{epi}(\phi)$ if and only if $M \models \Xi_{ont}(\phi)$, for every $M \in \mathbf{M}_{tf}(S)$.

Proof: (1) and (2) are proved by a simple (but tedious) induction on the structure of the formulae in S. (3) is also proved by induction on the structure of ϕ. We will only show the basic step.

Suppose $\phi = p$ for some ground atomic formula p. Then $\Xi_{epi}(p) = \Xi_{ont}(p) = p : \mathbf{t}$. By the definition of Ξ, $S \models p$ if and only if $I \models p$ for every model I of S if and only if $\Xi(I) \models p : \mathbf{t}$.

If $\phi = \neg p$ then $\Xi_{epi}(\phi) = p : \mathbf{f}$ while $\Xi_{ont}(\phi) = \neg p : \mathbf{t}$. By the definition of Ξ, $S \models \neg p$ if and only if $\Xi(I) \models p : \mathbf{f}$ and $\Xi(I) \not\models p : \mathbf{t}$ if and only if $\Xi(I) \models \Xi_{epi}(\phi)$ and $\Xi(I) \models \Xi_{ont}(\phi)$ for every model I of S.

Since Ξ is a bijection between $\mathbf{M}(S)$ and $\mathbf{M}_{tf}(S)$, $\Xi_{epi}(\phi)$ and $\Xi_{ont}(\phi)$ are true in every model from $\mathbf{M}_{tf}(S)$. □

Informally, Theorem 4 says that for consistent sets of formulae S, APC does not buy us much new: both embeddings, Ξ_{epi} and Ξ_{ont}, yield essentially the same results, and the set of PC-models of S can be naturally identified with a representative subclass $\mathbf{M}_{tf}(S)$ of models of $\Xi_{ont}(S)$ and $\Xi_{epi}(S)$.

However, when S' is inconsistent, then both $\mathbf{M}(S')$ and $\mathbf{M}_{tf}(S')$ are empty, and this is one of the situations when APC becomes useful.

Theorem 5: A set of PC formulas S is inconsistent if and only if $\Xi_{ont}(S)$ is o-inconsistent.

Proof: It is easy to see that under Ξ_{ont}, each resolution step applied to a pair of clauses, $\neg p(t_1, \ldots, t_n) \vee C_1$ and $p(s_1, \ldots, s_n) \vee C_2$, yielding $C_1\sigma \vee C_2\sigma$, where $\sigma = \mathtt{mgu}(p(t_1, \ldots, t_n), p(s_1, \ldots, s_n))$, translates into a resolution step applied to $\neg p(t_1, \ldots, t_n) : \mathbf{t} \vee \Xi_{ont}(C_1)$ and $p(s_1, \ldots, s_n) : \mathbf{t} \vee \Xi_{ont}(C_2)$, yielding $\Xi_{ont}(C_1)\sigma \vee \Xi_{ont}(C_2)\sigma$ $(\equiv \Xi_{ont}(C_1\sigma \vee C_2\sigma))$.

Similarly, factorization in PC translates into factorization in APC. It is now easy to see that every derivation of an empty clause from S (in PC) translates under Ξ_{ont} into a derivation of an empty clause from $\Xi_{ont}(S)$. Hence, inconsistency of S implies o-inconsistency of $\Xi_{ont}(S)$.

Conversely, suppose $\Xi_{ont}(S)$ is o-inconsistent. Then it has no model in APC. In particular, $\mathbf{M}_{tf}(S)$ is empty. But then, by Theorem 4, the set of models of S (in PC) is also empty, hence S is inconsistent in PC. □

According to Theorem 5, Ξ_{ont} is a faithful interpretation of PC inside APC in the following sense:

Corollary 2: $S \models \phi$ in PC if and only if $\Xi_{ont}(S) \models \Xi_{ont}(\phi)$ in APC.

Proof: It is easy to see that $\Xi_{ont}(\neg\phi) = \neg\,\Xi_{ont}(\phi)$. Thus, $S \models \phi$ if and only if $S \cup \{\neg\phi\}$ is inconsistent if and only if (by Theorem 5) the set $\Xi_{ont}(S) \cup \{\Xi_{ont}(\neg\phi)\}$ is inconsistent if and only if $\Xi_{ont}(S) \cup \{\neg\Xi_{ont}(\phi)\}$ is inconsistent if and only if $\Xi_{ont}(S) \models \Xi_{ont}(\phi)$. □

In particular, Ξ_{ont} does not let one reason about inconsistency in PC, which is not surprising, since this mapping regards S as a collection of ontologically correct statements *about* reasoner's beliefs (that cannot be inconsistent), rather than as an "internal state" of reasoner's beliefs (that may be inconsistent).

As we have already mentioned, many existing logical theories are too sensitive to contradictory information. It may happen that the addition of an elementary piece of data (like a unit clause) to a large knowledge base S makes it inconsistent, and hence destroys the meaningful information contained in S. One of the goals of this work is therefore to show that APC is able to isolate that information (which humans would intuitively regard as the "cause of inconsistency") from the information that intuitively has nothing to do with the inconsistency, thereby preserving the original meaning of the "unspoiled" data in S. In the rest of this section we formally show that the epistemic interpretation of PC within APC, Ξ_{epi}, results in the desired isolation of inconsistency. Thus, our claim that APC's way of modeling inconsistency closely corresponds to human reasoning will be supported by a host of rigorous arguments, not only by a number of carefully selected examples, which is quite rare in AI literature.

It is easy to see that every set R is mapped by Ξ_{epi} into an o-consistent set R'. Thus, Ξ_{epi} permits full tolerance of inconsistency in the reasoner's beliefs. By Theorem 4, if R is inconsistent then R' has no tf-model. Thus, in this case every model of R' necessarily contains $\top$-annotated literals. Therefore, inconsistency in R is reflected by the occurrences of $\top$ in the models of R'. Moreover, an appearance of a $p : \top$ in a most e-consistent model of R' indicates that p may be a reason for inconsistency in R. We formalize this observation below.

In the ensuing discussion we will restrict ourselves to the case of formulae in Skolemized form. We do not have a characterization of the cause of inconsistency in the general case. However, this restriction seems satisfactory for many practical purposes, since Skolemization is routinely performed in resolution-based proof procedures. Once Skolemized, we can assume without loss of generality that all formulae are in clausal form. Since, by Herbrand's Theorem, for every inconsistent set S of clauses there is a finite inconsistent set of ground instances of S, we can further limit our discussion to the case of ground clauses.

Consider a (consistent or inconsistent) set, R, of ground clauses in PC. A ground atom, p, is a *suspect* with respect to R (a "suspected cause" of inconsistency in R) if there are *consistent* subsets R_1, $R_2 \subseteq R$ such that $R_1 \models p$ and $R_2 \models \neg p$. Any set of formulae, E, such that $R_1 \cup R_2 \subseteq E \subseteq R$ (in particular, R itself) is called an *indictment* against p in R. E *indicts* S, where S is a set of atoms, if and only if E is an indictment against one of the atoms in S.

The notion of a "suspect" does not suffice to characterize the cause of inconsistency. To see why, consider the following example. Let R be $\{p, \neg p, \neg q, q \leftarrow p\}$. Then R indicts both p and q. However, intuitively, p seems more of an inconsistency cause than q, since p is suspected regardless of what are the suspicions about q. In other words, removing clauses involving q does not make R consistent because p and $\neg p$ can still be derived. The following definitions formalize this intuition.

An atom p *implicates* a set of atoms S if every indictment E against p is also an indictment against S. According to this definition, nonsuspects are true paranoiacs: since no indictment exists against them, every nonsuspect implicates everybody, including self. A *gang* (of culprits of inconsistency in R) is a set, G, of atoms occurring in formulas in R, such that:

(1) Every atom in R implicates G; and

(2) No proper subset of G possesses property (1).

Since any nonsuspect can be dropped from a set of atoms without affecting property (1), it follows that every gang-member is a suspect. Thus, a gang is a minimal set of suspects that is implicated by everybody. Clearly, R has gangs if and only if it is inconsistent. Furthermore, if R is inconsistent, it may have more than one gang. The following example illustrates these notions.

Example 1: For the set of clauses p, $\neg p$, $\neg q$, $q \leftarrow p$ mentioned earlier, both p and q are suspects, as p, $\neg q$, $q \leftarrow p$ indicts both. Also, q implicates p, since every indictment against q is also an indictment against p (verified directly). Since also p (trivially) implicates $\{p\}$, the latter is a gang. On the other hand, q is not a gang, since it is not implicated by p (p, $\neg p$ is an indictment against p, but not against q).

The following set of formulae $\{p,\ q,\ r,\ \neg q \leftarrow p,\ \neg r \leftarrow q,\ \neg p \leftarrow r\}$ has three gangs: $\{p,\ q\}$, $\{q,\ r\}$, and $\{r,\ p\}$. The set $\{p,\ q,\ \neg q \leftarrow p\}$ is an indictment against q and p, the set $\{q,\ r,\ \neg r \leftarrow q\}$ is an indictment against q and r, and so on.

For yet another example, consider the set of formulae $R = \{q,\ \neg q,\ r,\ \neg r,\ p \leftarrow q,\ \neg p \leftarrow r\}$. Here the only gang is $\{q,\ r\}$. Although p is also a suspect, it implicates this gang, which can be verified by considering all indictments against p. For instance, consider the following indictment against p: q, r, $p \leftarrow q$, $\neg p \leftarrow r$. This is also an indictment against q and r. To see this, say, in case of q, consider $R_1 = \{q\}$ and $R_2 = \{r,\ p \leftarrow q,\ \neg p \leftarrow r\}$. Clearly, both R_1 and R_2 are consistent and $R_1 \models q$ while $R_2 \models \neg q$.

On the other hand, $\{p,\ r\}$ is not a gang, since an indictment $\{q,\ \neg q\}$ against q does not indict $\{p,\ r\}$. $\Box$

It is interesting to note that our treatment of the cause of inconsistency follows the standard prescription for defining logical entailment. The notion of an "indictment" corresponds to the concept of a model in logical theories. The notion of "gang-implication" then becomes identical to logical entailment once the term "model" is replaced by the term "indictment." Thus, a "gang" is a minimal set of suspects "logically entailed" in that sense by everybody. This intuition is strengthened by the theorems below; they say that it is precisely the gang members of R (the *gangsters*) who may be mapped by Ξ_{epi} into inconsistent beliefs in one of the most e-consistent models of R. In particular, by (1) and (2) of Theorem 6, S is consistent in PC if and only if each of the most e-consistent models of $\Xi_{epi}(S)$ is also an e-consistent semantic structure. In other words, Ξ_{epi} is a faithful translation of the cause of inconsistency in PC into e-inconsistency in APC.

Theorem 6: Let S be a set of ground clauses in PC. Then the following holds true.

(1) If G is a gang in S, then there is a model M of $\Xi_{epi}(S)$ such that $G = \{p \mid M \models p : \top\}$ and M is most e-consistent within the class of models of $\Xi_{epi}(S)$.

(2) Let M be a most e-consistent model of $\Xi_{epi}(S)$. Then the set $\{p \mid M \models p : \top\}$ is a gang in S.

For instance, in Example 1 we have $S = \Xi_{epi}(S')$, and p and q are gangs in S'. Thus, every model of S (cf. m_1, m_2, m_3, m_4) contains either $p : \top$ or $q : \top$. In contrast, r is not a cause of inconsistency in S' and, indeed, $r : \top$ does not belong to any most e-consistent model (namely, m_2, m_4) of S.

Proof: First observe that for every semantic structure I, the set of atoms $I_H = \{p : \mathbf{s} \mid I \models p : \mathbf{s}\}$ is a Herbrand interpretation. Furthermore, for any clause C,[4] $I \models C$ if and only if $I_H \models C$; hence I is a most e-consistent model of C if and only if so is I_H. We can thus restrict our attention to Herbrand interpretations and models.

Call any clause of the form $p_1 : \top \vee \ldots \vee p_k : \top$ — a $\top$-clause. We will need the following three claims whose proofs appear in Appendix A.

Claim 1 (Corollary A1 in Appendix A): Let S be a set of ground clauses, and $\bar{C}$ be a possibly infinite set of ground literals of the form $q : \top$. Suppose also that $\bar{C}$ has the following *intersection property*: For each model M of $\Xi_{epi}(S)$, $M \cap \bar{C} \neq \emptyset$. Then S is inconsistent (in PC), and there is a minimal (with respect to the above intersection property) finite subset C of $\bar{C}$, such that for every element $q : \top \in C$, q is a suspect of inconsistency in S.

[4]Note: a *clause*, not just any formula. Absence of existential quantifiers in the prenex normal form is essential here.

Claim 2 (Lemma A2 in Appendix A): If E is an inconsistent subset of clauses of S then there is a $\top$-clause, $C = q_1 : \top \vee \ldots \vee q_n : \top$, where $n \geq 1$, such that $\Xi_{epi}(E) \models C$ and E is an indictment against every one of the q_1, ..., q_n.
Claim 3 (Lemma A3 in Appendix A): Let G be a gang of culprits of inconsistency of S. Then

(1) For each $p \in G$ there is a $\top$-clause C of the form $p : \top \vee C'$ such that $\Xi_{epi}(S) \models C$ and C' contains no gangsters from G.

(1) If C is a $\top$-clause such that $\Xi_{epi}(S) \models C$ then C contains at least one gangster from G.

Assuming the above claims, we return to the proof of the theorem.

(1) Let M be a Herbrand model of $\Xi_{epi}(S)$ such that $p : \top \in M$ if and only if $p \in G$. Such a model exists, for if not, it would be the case that every Herbrand model of $\Xi_{epi}(S)$ contains a literal of the set $\bar{C} = \{q : \top \mid q$ is a literal of the Herbrand base that is *not* in $G\}$. But then, Claim 1 guarantees that there is a minimal finite subset C of $\bar{C}$ that intersects every model of $\Xi_{epi}(S)$. By (2) of Claim 3, C contains one of the gangsters from G, contrary to the construction of C.

The fact that M is a most e-consistent model of S holds by the following argument. Suppose there is a strictly more e-consistent model N, i.e. for some $p \in G$, $p : \top \notin N$ and $\{q \mid q : \top \in N\} \subset G$. Let $p : \top \vee C'$ be a $\top$-clause such that $\Xi_{epi}(S) \models p : \top \vee C'$ and C' does not contain any member of G. Such a clause exists by (1) of Claim 3. But then this clause is falsified by N and hence N cannot be a model of $\Xi_{epi}(S)$.
(2) Let M be a most e-consistent model of $\Xi_{epi}(S)$ and $G = \{q \mid q : \top \in M\}$. We will show that G is a gang.

Consider an arbitrary element $q_0 \in G$ and a set $S' \subseteq S$ consisting entirely of clauses that contain no atoms from $G - \{q_0\}$. Let $C = \{p : \top \mid p$ is ground, and either $p \notin G$ or $p = q_0\}$. Notice that, by construction, $q_0 : \top \in C$. Furthermore, C has the following *intersection property*: C intersects every model N of $\Xi_{epi}(S')$. Indeed, let N be a Herbrand model of $\Xi_{epi}(S')$ such that $N \cap C = \emptyset$. Then $N \cup \{p : \top \mid p \in G, p \neq q_0\}$ is a model of $\Xi_{epi}(S)$ that is strictly more e-consistent than M. Also, every minimal subset of C that has the above intersection property contains $q_0 : \top$, since M is a model of S such that $M \cap C = \{q_0 : \top\}$.

Now, consider such a minimal subset C' of C, which exists by Claim 1. Since $q_0 : \top \in C'$, q_0 is a suspect, by Claim 1. As q_0 was chosen arbitrarily, every member of G is a suspect in S.

To show that every suspect implicates G, consider some suspect p along with an indictment E against p in S. By Claim 2, there is a clause $C = q_1 : \top \vee \ldots \vee q_n : \top$ such that $\Xi_{epi}(S) \models C$ and E is an indictment against q_1, ..., q_n. Since M is a model of $\Xi_{epi}(S)$, C is satisfied by M; hence one of the q_is must be in G. Thus, E is an indictment against one of the members of G. Since E is an arbitrarily chosen indictment against p, it follows that p implicates G.

Finally, G is a gang: If it were not, there would be a gang $G' \subset G$. By (1) of Theorem 6, there would then be a model N of S such that $G' = \{p \mid N \models p : \top\}$, which is strictly more e-consistent than M — a contradiction. □
Corollary 3: Let S be a set of ground clauses in PC. It is inconsistent if and only if every model $\Xi_{epi}(S)$ is epistemically inconsistent (i.e. implies at least one inconsistent belief — see Section 3.1).
Proof: S is inconsistent if and only if it has a gang. Theorem 6 then says that every most e-consistent model of S (hence every model) is e-inconsistent. □

Our next result shows that Ξ_{epi}, in a well defined sense, "salvages" part of the information that still can be viewed as undamaged by the inconsistency. First, we need a number of definitions.

Let S be, as before, a (possibly infinite) inconsistent set of ground clauses. Consider a literal, l, of the form p or $\neg p$, where p is a ground atomic formula. We say that l is *spoiled* by the inconsistency in S, if p belongs to every gang; it is *supported* if there is a consistent subset S' of S that entails l: $S' \models l$. In this case we say that S' is an *evidence* for l. A literal l is *recoverable* if either

(i) l is supported and non-spoiled, and for every gang G in S, either $p \in G$ (here p is the atom such that either $l = p$ or $l = \neg p$), or there is an evidence S' for l, such that S' mentions no gangster from G;[5] or

(ii) l is a negation of a literal that satisfies (i), i.e., negation of a recoverable literal is also recoverable.

Finally, a supported literal is *damaged* (by the inconsistency in S) if it is neither spoiled nor recoverable.

We will see that the spoiled literals cannot be assigned any meaning other than that of an inconsistent belief. Recoverable literals, as their name suggests, can be viewed as the "robust" part of S: They still can be assigned a consistent truth value, even though their supporting evidences may rely on gangster's testimonies. Intuitively, the robustness of recoverable literals comes from the fact that their supporting evidences do not depend on any one gang, which makes those evidences sufficiently credulous.

On the other hand, a damaged literal can be viewed as a literal that may have had a meaning before S became inconsistent (since it still has supporting evidences), but that meaning cannot be reliably recovered, because all these evidences depend on the same gang (which cannot be trusted). For instance, if $S = \{q, \neg q, p \leftarrow q, \neg p, r \leftarrow q, \neg r \leftarrow q\}$, the set $\{q\}$ is the only gang and hence q is spoiled. It can be also verified that p and $\neg p$ are recoverable, while r, $\neg r$ are damaged. The ability to recover the meaning of $\neg p$ can be intuitively explained by the fact that inconsistency in q precludes deriving p, which leaves us with $\neg p$. Informally, $\neg p$ can be viewed as a true statement "despite the inconsistency." Its negation, p, can then be thought of as a false statement. Observe also that deleting $\neg q$ from S would make $\{p, r\}$ a gang and so q will no longer be a spoiled atom. Instead, q and $\neg q$ will become recoverable, while the other literals will retain their old status. The next theorem justifies our definitions.

Theorem 7: Let S be an arbitrary inconsistent set of clauses and l be a literal of the form p or $\neg p$, where p is a ground atom. Then

(1) l is spoiled if and only if $\Xi_{epi}(S) \mid\!\approx p : \top$.

(2) l is recoverable if and only if $\Xi_{epi}(S) \not\mid\!\approx p : \top$ and either $\Xi_{epi}(S) \mid\!\approx \Xi_{epi}(l)$ or $\Xi_{epi}(S) \mid\!\approx \sim \Xi_{epi}(l)$.

Proof: (1) If l is spoiled, it belongs to every gang and, by (2) of Theorem 6, $p : \top$ holds true in every most e-consistent model of S. Hence $\Xi_{epi}(S) \mid\!\approx p : \top$. Conversely, if $\Xi_{epi}(S) \mid\!\approx p : \top$ then $p : \top$ belongs to every most e-consistent model of S and, by (1) of Theorem 6, p belongs to every gang.

(2) "Only if": Suppose l is recoverable. If $\Xi_{epi}(S) \mid\!\approx p : \top$ then, by (1), l is spoiled contrary to the assumption that it is recoverable. Therefore, $\Xi_{epi}(S) \not\mid\!\approx p : \top$

Next, since $\Xi_{epi}(\neg l) = \sim \Xi_{epi}(l)$, it suffices to establish the case when l is recoverable due to condition (i) in the definition of recoverability. So, suppose l is such a literal, and consider a most consistent model M of $\Xi_{epi}(S)$. Let $G = \{q : \top \mid M \models q : \top\}$. By (2) of Theorem 6, G is a gang. If $p \in G$ then $M \models p : \top$ hence $M \models \Xi_{epi}(l)$.

So, suppose that $p \notin G$. Since l is recoverable, there is a consistent evidence $S' \subseteq S$ that does not mention atoms from G and $S' \models l$. Since both S' and l are ground, the factorization rule does not need to be applied, and there exists a derivation (in PC) by binary resolution[6] of l from S'.

Under Ξ_{epi}, this derivation translates into a derivation by reduction of a clause of the form $\Xi_{epi}(l) \vee q_1 : \top \vee \ldots \vee q_n : \top$, where the q_is are some of the atoms mentioned in S'. Because of soundness of the reduction rule, $\Xi_{epi}(S) \models \Xi_{epi}(l) \vee q_1 : \top \vee \ldots \vee q_n : \top$. Since S' does not mention members of G, none of the q_is belongs to G. Therefore, by the construction of G, each of the $q_i : \top$ is false in M. It remains that $M \models \Xi_{epi}(l)$. Since M is arbitrary, $\Xi_{epi}(S) \mid\!\approx \Xi_{epi}(l)$.

[5] Since l is non-spoiled, there is at least one such evidence.

[6] Note: we are talking here about derivation *from* S', not refutation *of* S'. That is, here we are dealing with a set of clauses c_1, ..., c_k, such that $c_k = l$ and each c_{i+1} either belongs to S' or is derived by a resolution step from $\{c_1, \ldots, c_i\}$, $i = 1, \ldots, k-1$.

"If": In the other direction, suppose that $\Xi_{epi}(S) \mid\!\approx \Xi_{epi}(l)$ and $\Xi_{epi}(S) \not\mid\!\approx p : \top$. Then l is non-spoiled, and we can find a gang, G, such that $p \notin G$. Consider a most consistent model, M, such that $G = \{q \mid M \models q : \top\}$; it exists by (1) of Theorem 6.

Let S' be a subset of these clauses in S that do not mention any member of G. Then S' is nonempty. Indeed, if it were empty, then every clause in S would have contained either q or $\neg q$ as a disjunct, for some $q \in G$. But then every clause in $\Xi_{epi}(S)$ would have contained $q : \mathbf{t}$ or $q : \mathbf{f}$. Since $M \models q : \top$, the set $G \cup \{r : \bot \mid r \notin G\}$ would then be a most e-consistent model of $\Xi_{epi}(S)$ that falsifies $p : \mathbf{t}$ and $p : \mathbf{f}$, contrary to the assumption that $\Xi_{epi}(S)$ epistemically entails $\Xi_{epi}(l)$.

Furthermore, S' is consistent. To see that, consider a Herbrand interpretation M' that coincides with M on the atoms in the complement to G and assigns the annotation $\bot$ to the atoms in G. Since M is a model of $\Xi_{epi}(S)$ and M' differs from M only on the atoms from G, M' is a model of $\Xi_{epi}(S')$. It is an e-consistent model, since M associates the $\top$-annotations only with the elements of G. Now, M' can be made into a tf-model of $\Xi_{epi}(S')$ by adding, say, $r : \mathbf{t}$ to M' for every atom r for which $M \not\models r : \mathbf{t}$ and $M \not\models r : \mathbf{f}$ both hold. This can be done because the ontological negation, "$\neg$", does not appear in the clauses of $\Xi_{epi}(S')$. Now, since $\Xi_{epi}(S')$ has a tf-model, S' has a model in PC, by (2) of Theorem 4, and, therefore, is consistent.

It remains to show that $S' \models l$. Consider a tf-model, I, of $\Xi_{epi}(S')$; its existence is guaranteed by Theorem 4, since S has just been shown to be consistent. By the construction of S', $N = I \cup \{q : \top \mid q \in G\}$ is a model of $\Xi_{epi}(S)$. It is a most e-consistent model, since if there were a model of $\Xi_{epi}(S)$ that is strictly more e-consistent than N, then M would not have been a most e-consistent model either — a contradiction. Now, since N is a most e-consistent model of $\Xi_{epi}(S)$, it follows that $N \models \Xi_{epi}(l)$, as $\Xi_{epi}(S) \mid\!\approx \Xi_{epi}(l)$ by the assumption. Since $p \notin G$, it also follows that $I \models \Xi_{epi}(l)$. Because of the arbitrariness of I, we conclude that $\Xi_{epi}(l)$ is true in every tf-model of $\Xi_{epi}(S')$. Then, by (3) of Theorem 4, $S' \models l$. □

Finding the recoverable part of the information is at least as hard as the problem of logical entailment itself. This is because if S is consistent, recoverability reduces to logical entailment. In general, detecting whether a literal is recoverable requires a proof procedure for epistemic entailment. At present, we have such procedure only for a special case when S is a logic program, which will be reported elsewhere. On the other hand, spoiled literals are easy to detect:
Corollary 4: A ground literal l is spoiled in S if and only if both $l \in S$ and $\neg l \in S$.
Proof: Without loss of generality, assume that $l = p$ for some ground atom p. If p is spoiled, (1) of Theorem 7 yields $\Xi_{epi}(S) \mid\!\approx p : \top$. But then also $\Xi_{epi}(S) \models p : \top$ holds true, since if an inconsistent belief is held in every most e-consistent model of S then it is held in every model of S. Hence, since we are dealing only with ground literals and clauses and because $\Xi_{epi}(S)$ mentions no ontological negation, there must be a derivation by reduction of $p : \top$ from $\Xi_{epi}(S)$. Indeed, resolution, factorization, elimination, and reduction are complete, factorization is inapplicable due to groundedness, while elimination and resolution do not apply because the ontological negation does not occur in $\Xi_{epi}(S)$. But since reduction does not decrease the size of a clause, $p : \top$ can be derived by reduction from $\Xi_{epi}(S)$ only if both $p : \mathbf{t}$ and $p : \mathbf{f}$ are in $\Xi_{epi}(S)$, which is equivalent to saying that $p, \neg p \in S$.

In the other direction, if $p, \neg p \in S$ then $p : \mathbf{t}, p : \mathbf{f} \in \Xi_{epi}(S)$. Hence, $M \models p : \top$ for every model of $\Xi_{epi}(S)$. But then, (1) of Theorem 6 implies that p belongs to every gang. □

7 Generalizations

By a simple inspection of proofs, the reader can find out that the condition $\texttt{lub}(\mathbf{t}, \mathbf{f}) = \top$ in BSL is used only in Section 6; it is irrelevant to the proof theory of APC.

The least element of BSL, $\bot$, is used in Section 6 only in the proof of Theorem 7. However, the arguments in this proof go through also if we assumed the existence of *some* lower bound of $\mathbf{t}$ and $\mathbf{f}$, which is much weaker than requiring BSL to have the least element. The annotation, $\bot$, is also used in the proof of Proposition B1 in Appendix B (completeness of ground deduction).

In knowledge representation, it is often convenient to associate different BSLs with different predicates. For instance, for a predicate *grade*, the four valued lattice of Figure 1(a) seems quite adequate, since a fact such as $grade(john,\ cs100,\ F)$ can be easily verified and it is unlikely that someone would want to speculate about the "likelihood" of the fact that John's transcript contains "F" for the course *cs100.* In contrast, it is more difficult to talk categorically about facts of the type $causes(cause,\ life\_style,\ desease)$, and finer lattices* may be more appropriate here.[7] It is easy to verify that degrees of belief annotating different predicate symbols do not interact with each other and, therefore, we could associate different BSLs with different predicate names, essentially without changing anything in the definitions and proofs.

Finally, we note that even the very requirement that BSL must form an upper semilattice can be dropped. Instead of postulating that every finite set of elements must have a lub, we could require the existence of a finite number of *minimal* upper bounds (mub's). For the proof theory of Section 5 to be sound and complete we would only have to change the reduction rule to:

> From $p(\bar{t})\ :\ \mathbf{s} \vee \phi$ and $p(\bar{t'})\ :\ \mathbf{s}' \vee \psi$ infer $p(\bar{t})\theta\ :\ \mathbf{r}_1 \vee \ldots \vee p(\bar{t})\theta\ :\ \mathbf{r}_n \vee \phi\theta \vee \psi\theta$, where θ is the mgu of $p(\bar{t})$ and $p(\bar{t'})$, and $\mathbf{r}_1$, ..., $\mathbf{r}_n$ are all the mub's of $\mathbf{s}$ and $\mathbf{s}'$.

However, such generalization comes at a price: multiple mub's lead to added complexity of proofs. For instance, from a pair of facts, $p\ :\ \mathbf{s}$ and $p\ :\ \mathbf{s}'$, by reduction one can infer a disjunction, $p\ :\ \mathbf{r}_1 \vee \ldots \vee p\ :\ \mathbf{r}_n$. Therefore, the class of Horn clauses is not closed under the inference rules, and no efficient proof strategy, such as SLD-resolution, is likely to exist. In contrast, an analogue of SLD resolution is known to exist for Horn clauses in APC [31] Thus, our restriction on BSL to be a semilattice was motivated by the desire to keep the proof theory simple.

8 An Alternative Semantics

It was noted in Section 5 that Herbrand's theorem does not hold for APC in its full generality, and the restrictions in Theorems 2 and 3 are necessary. In this section we provide an alternative semantics for which the analogues of these theorems hold without the corresponding restrictions; this semantics leads to essentially the same proof theory and entailment relation $\models$ for sets of clauses that involve only a finite number of annotations (i.e. that satisfy the restrictions of Theorems 2 and 3).

The "traditional" semantics of Section 2.2 was used in [3, 16] in the context of logic programs. However, it turned out that adhering to this semantics makes the operator T_P (which is commonly used in logic programming) to be discontinuous. This lead Blair, Kifer, and Subrahmanian to restrict consideration to special subclasses of logic programs. With the new semantics, T_P can be shown to be continuous and the problem disappears. Furthermore, Corollary 5 below ensures that the results of [3, 16] carry over to the new semantics.

Semantic structures in the new semantics are similar to the old ones, i.e., they are triples $< D,\ I_F,\ I_P >$, where D and I_F have the same meaning, as before. However, I_P now interprets each predicate symbol, p, by a function $I_P(p)\ :\ D^m \longrightarrow I(BSL)$, where $I(BSL)$ is a set of all *ideals* of BSL. A subset $\mathbf{r}$ of BSL is an *ideal* if it is:

1. *Downward closed* , i.e. if $d \in \mathbf{r}$ and $e \leq d$ then $e \in \mathbf{r}$; and
2. Closed with respect to finite lub's, i.e. if $d_1,\ d_2 \in \mathbf{r}$ then $\mathtt{lub}(d_1, d_2) \in \mathbf{r}$.

The only change in the definition of $\models_v$ with respect to Section 2.2 concerns atomic formulas: $I \models_v p(t_1, ..., t_k)\ :\ \mathbf{s}$ if and only if $\mathbf{s} \in I_P(p)(v(t_1), \ldots, v(t_k))$.

To see the connection with the old notion of semantic structure, notice that each element of BSL, $\mathbf{r}$, determines a *principal* ideal of BSL, $ideal_{\mathbf{r}}$, in the usual way: $ideal_{\mathbf{r}} = \{\mathbf{s} \mid \mathbf{s} \leq \mathbf{r}\}$. Thus, each semantic structure in the old sense can be viewed as a structure in the new sense, but not vice versa. For instance, a mapping I_P that maps p into the set $\{\mathbf{t}_r \mid 0 \leq r < 1\}$ in Example 8 of

[7]E.g., a lattice of pairs $[a,\ b]$, where $0 \leq a,\ b \leq 1$; $[a,\ b] \leq [c,\ d]$ if and only if $a \leq c$ and $b \leq d$ (see, for instance, [15].

Section 5 is legal according to the new definition of semantic structures but not according to the old one: In the old definition, $I_P(p)$ would map D^m into a proper subset of $I(BSL)$ consisting of all principal ideals.

As mentioned, the reason for developing the new semantics is our desire to find a theory for which both Herbrand's theorem (Theorem 2) and the completeness theorem (Theorem 3) hold without restrictions. The main result of this section is that, for the new semantics, a refutation procedure consisting of resolution, factorization and reduction rules is sound and complete, even for sets of clauses involving infinite number of annotations. Interestingly, the elimination rule is no longer sound with respect to the new semantics: $\neg p : \bot$ is satisfied by every semantic structures that assigns to p the empty ideal, $\{\}$.

Theorem 3': Under the new semantics, a refutation procedure that involves resolution, factorization, and reduction alone is sound and complete for testing o-inconsistency of arbitrary sets of clauses. $\Box$

An easy inspection of the proof of Theorem 3 in Appendix B shows that modifications are necessary in the proofs of Herbrand's theorem (Theorem 2) and Proposition B1. In Proposition B1 the only change is that there is no need to apply the elimination rule. All the rest goes without change. Herbrand's theorem for the new semantics is re-proved below.

It should be noted that the restriction in Theorem 3 stems from the corresponding restriction in Theorem 2. So, once we show that with respect to the new semantics Herbrand's theorem holds without restrictions, so will the completeness theorem.

Theorem 2' (Herbrand's theorem for the new semantics): Let S be an arbitrary set of clauses. Then it is o-inconsistent if and only if so is some finite subset of ground instances of S.

Proof: Instead of modifying the old proof (which is not so easy to do), we will use a technique borrowed from [7] where it was used for proving the compactness theorem for propositional calculus.

First note that, by Lemma 1 (whose proof requires no modification), S is unsatisfiable if and only if so is the set S' of all ground instances of S. Therefore, without loss of generality we can assume that S is ground. To prove the theorem we will show that if every finite subset of S is satisfiable then so is S.

Notice that each atomic formula in the Herbrand base of APC can be viewed as a proposition. Therefore, the above statement becomes identical to the compactness theorem of predicate calculus. Without repeating every detail of the proof in [7], we will sketch its most salient points and indicate where minor modifications are needed.

Let us call a ground set of formulas of APC *finitely satisfiable* whenever every finite subset of this set is satisfiable. We thus have to show that every finitely satisfiable set is satisfiable. The proof consists of two steps. First, given a finitely satisfiable set S, a *maximal* finitely satisfiable superset S^* of S is constructed by the same process as in [7]. In the second step, we show that S^* is a Herbrand interpretation of S (in the new sense). The proofs require only minor modifications with respect to [7], such as showing that if $p : \mathbf{r} \in S^*$ and $\mathbf{s} \leq \mathbf{r}$ then $p : \mathbf{s} \in S^*$, and that if $p : \mathbf{r}, p : \mathbf{s} \in S^*$ then $p : \mathtt{lub}(\mathbf{r}, \mathbf{s}) \in S$.

It is a simple yet useful exercise for the motivated reader to find out where the second step of this proof fails for the old notion of Herbrand interpretations, which made it necessary to place restrictions on S (e.g. why S^* in Example 8 of Section 5 is not an Herbrand interpretation of S in the old sense, but *is* an interpretation in the new sense). $\Box$

Corollary 5: Let S be a set of APC formulas involving only a finite number of annotations and ϕ be a formula. Suppose also that neither S nor ϕ has literals of the form $\neg p : \bot$. Then $S \models_{new} \phi$ if and only if $S \models_{old} \phi$. In other words, the old and the new semantics give the same results on the subset of formulas for which Theorem 3 holds, provided that the elimination rule does not need to be applied.

Proof: Theorems 3 and 3' deal with exactly the same proof procedure, except that the elimination rule is not used in Theorem 3'. Both proof procedures are sound and the one in Theorem 3' is also complete. The proof procedure used in Theorem 3 is also complete under the restrictions listed in the Corollary. The claim therefore follows form these facts and since, by assumption, the elimination rule does not need to be applied. □

9 Conclusions

We have presented a logic capable of handling inconsistent beliefs. Several extensions to APC are possible. First, without any additional apparatus one can associate different semilattices to different predicates. This may be useful when granularity of degrees of beliefs differs from predicate to predicate.

Second, it is possible to relax the restriction that in a literal, $p : \mathbf{s}$, the annotation "s" is a semilattice constant. Instead, a number of monotonic semilattice functions and variables could be allowed, which would bring in the terms of the form $p : f(X, Y, a)$. Variables over semilattices can be quantified, which adds more power to the language, for instance, allowing rules like

$$\forall X \, \forall Y \, (\mathit{flies}(X) \; :\!\sim Y \leftarrow \mathit{penguin}(X) \; : Y \,).$$

Unfortunately, we are not aware of any complete refutation procedure for such an extended logic. The difficulty here is that one cannot assume semilattice functions to be uninterpreted (as in the theory of Logic Programming), since for the most part lattices are finite. In [15] complete proof procedure was developed for a restricted logic in which only Horn-like rules are permitted and body literals may have only lattice variables and constants as annotations. The same ideas can be adapted for a subset of APC and some results in this direction are reported in [16].

Third, the framework of APC can be used to reason about elementary beliefs of multiple agents by attaching multiple annotations to literals. For instance, the literal $p \, :_i \mathbf{t} \, :_j \mathbf{f}$ can be viewed as a statement that the reasoner j does not believe that the reasoner i believes in p. We leave this issue for the future research.

In a broader perspective, APC can be regarded as a technique for extending the scope of many "conventional" logics (including modal logics) to dealing with inconsistency. In this sense, the present work is an example of applying that technique to the standard predicate calculus. We also conjecture that the concepts of *gangs* and *recoverable literals* developed in Section 6 can be made into useful heuristics for driving belief revision processes in truth maintenance systems.

Acknowledgements: We thank V.S. Subrahmanian for the pointers to several papers on paraconsistent logics and for many helpful suggestions. Joseph Tashjian found several inaccuracies in the earlier draft of this paper, and Anita Wasilewska pointed out a connection with Post algebras. We are especially grateful to one of the referees for the very detailed report. The help of other referees is also appreciated.

Appendix A: Gang-related Matters

Lemma A1: Let S be a possibly infinite set of ground clauses and $C = p_1 : \top \vee \ldots \vee p_n : \top$ be a ground clause such that $\Xi_{epi}(S) \models C$ and no proper subclause of C (i.e. a disjunction in which some of the literals $p_i : \top$ are omitted) is implied by $\Xi_{epi}(S)$. Then S is inconsistent (in PC) and every literal p_i, $i = 1, \ldots, n$, is a suspect of inconsistency.

Proof: Consider a set of clauses, $\hat{S}$, that is a minimal inconsistent (in PC) subset of S and such that the clauses in $\hat{S}$ mention only the literals p_1, ..., p_n. If $\hat{S}$ did not exist then the subset S' of clauses that mentions only p_1, ..., p_n would be consistent. Let N be a tf-model of $\Xi_{epi}(S')$, which exists by Theorem 4. Being a tf-model, N necessarily falsifies the $\top$-clause C. Then the interpretation, $M = N \cup \{q : \top \mid q \text{ is not one of the } p_i\}$ is a model of $\Xi_{epi}(S)$ but is not a model of C, contrary to the assumption that $\Xi_{epi}(S) \models C$. Thus $\hat{S}$ does exist. It is also finite, since it involves only a finite number of literals. Since $S \supseteq \hat{S}$, S is also inconsistent.

Next, we show that $\Xi_{epi}(\hat{S}) \models C$ and no proper subclause of C is implied by $\Xi_{epi}(\hat{S})$. Since $\hat{S}$ is inconsistent, there exists a PC-derivation of an empty clause from $\hat{S}$ that uses the resolution rule alone. Under Ξ_{epi} this translates into a derivation by reduction (in APC) from $\Xi_{epi}(\hat{S})$ of a $\top$-subclause of C (it is a subclause of C because $\hat{S}$ involves only the literals mentioned in C). Thus, $\Xi_{epi}(\hat{S}) \models C$. No proper subclause of C is implied by $\Xi_{epi}(\hat{S})$, since otherwise this subclause would have been implied by $\Xi_{epi}(S)$, contrary to the minimality of C.

To complete the proof, we will show that each one of the literals p_1, ..., p_n is a suspect. Consider, say, p_1 and suppose it is not a suspect. Then either for every consistent subset S_0 of $\hat{S}$ it is the case that $S_0 \not\models p_1$, or for each such subset $S_0 \not\models \neg p_1$ holds. Let us assume the former, for definiteness. Consider a partition of $\hat{S}$ into a pair of sets, $\hat{S}_{\neg p_1}$ and $\hat{S}_{rest}$. The former set contains all clauses of $\hat{S}$ that contain the literal $\neg p_1$, while the latter contains the rest of the clauses in $\hat{S}$.

The set $\hat{S}_{\neg p_1}$ is nonempty. For if it were, then p_1 cannot be deleted (by resolution) from any of the clauses in $\hat{S}_{rest}$, hence it is not necessary for any clause containing p_1 to participate in a derivation of the empty clause from $\hat{S}$. Under Ξ_{epi} this derivation transforms into a derivation by reduction of a proper subclause of C from $\Xi_{epi}(\hat{S})$ (since no clause in this derivation mentions p_1) — contrary to the minimality of C.

Since $\hat{S}_{\neg p_1}$ is nonempty, $\hat{S}_{rest}$ is a proper subset of $\hat{S}$, and thus it is consistent in PC and $\hat{S}_{rest} \not\models p_1$, by the earlier assumption about subsets of $\hat{S}$. In particular, $\hat{S}_{rest}$ has a model M in which p_1 is false. By Theorem 4, Ξ_{epi} maps M into a tf-model $\hat{M}$ of $\Xi_{epi}(\hat{S}_{rest})$ such that $p_1 : \mathbf{f} \in \hat{M}$. But $\hat{M}$ is also a model for $\Xi_{epi}(\hat{S}_{\neg p_1})$ since all clauses in $\Xi_{epi}(\hat{S}_{\neg p_1})$ are of the form $p_1 : \mathbf{f} \vee \ldots$. Thus, $\hat{M}$ is a tf-model of $\Xi_{epi}(\hat{S})$. On the other hand, $\hat{M}$ falsifies C, since each $p_i : \top$, $i = 1, \ldots, n$, is false in $\hat{M}$ (because $\hat{M}$ is a tf-model). This contradiction shows that p_1 is a suspect. □

Corollary A1: Let S be as before, and $\bar{C}$ be a possibly infinite set of ground literals of the form $q : \top$. Suppose also that $\bar{C}$ has the following intersection property: For each model M of $\Xi_{epi}(S)$, $M \cap \bar{C} \neq \emptyset$. Then S is inconsistent (in PC), and there is a minimal (with respect to the above intersection property) finite subset C of $\bar{C}$ such that for every element $q : \top \in C$, q is a suspect of inconsistency in S.

Proof: If S were consistent, then it would have had a model. The Ξ_{epi}-image of that model would have been a tf-model of $\Xi_{epi}(S)$ with an empty intersection with $\bar{C}$ — a contradiction. As in the proof of Lemma A1, consider $\hat{S}$ — a subset of S that consist of clauses that are built only of those non-annotated atoms that are mentioned in $\bar{C}$. As in Lemma A1, this set can be shown nonempty and inconsistent (the assumption that $M \cap \bar{C} \neq \emptyset$ for every model M is used here, but the minimality of $\bar{C}$ with respect to this intersection property is not required).

Any derivation by resolution of an empty clause from $\hat{S}$ corresponds under Ξ_{epi} to a derivation by reduction of a finite disjunction $\hat{C} = p_1 : \top \vee \ldots \vee p_n : \top$, where the p_is are some of the literals mentioned in $\hat{S}$, i.e. $\Xi_{epi}(S) \models \Xi_{epi}(\hat{S}) \models \hat{C}$. By the construction of $\hat{S}$, each p_i, $i = 1, \ldots, n$, is also mentioned in $\bar{C}$. Since $\hat{C}$ is a finite disjunction, it has a minimal subdisjunction, $C = p_{i_1} : \top \vee \ldots \vee p_{i_k} : \top$, such that $\Xi_{epi}(\hat{S}) \models C$. The problem now reduces to that of Lemma A1. □

Lemma A2: If E is inconsistent then there is a clause $C = q_1 : \top \vee \ldots \vee q_n : \top$, where $n \geq 1$, such that $\Xi_{epi}(E) \models C$ and E is an indictment against every one of the q_1, ..., q_n.

Proof: There is a derivation by resolution of an empty clause from E. As observed earlier, this derivation gets transformed into a derivation by reduction of a nonempty $\top$-clause C' from $\Xi_{epi}(E)$. By the soundness of reduction, $\Xi_{epi}(E) \models C'$. Consider a minimal subclause of C', $C = q_1 : \top \vee \ldots \vee q_n : \top$, such that $\Xi_{epi}(E) \models C$. By Lemma A1, every literal, q_i, in C is a suspect of inconsistency of E. Thus, E is an indictment against every one of them. □

Lemma A3: Let G be a gang of culprits of inconsistency of S. Then:

1. For each $p \in G$ there is a $\top$-clause C of the form $p : \top \vee C'$ such that $\Xi_{epi}(S) \models C$ and C' contains no gangster from G.
2. If C is a $\top$-clause such that $\Xi_{epi}(S) \models C$ then C contains at least one gangster from G.

Proof: (1) Suppose, to the contrary, that each $\top$-clause implied by $\Xi_{epi}(S)$ that contains $p : \top$ also contains $q : \top$ for some $q \in G$, $q \neq p$. We will show that then p implicates a smaller set,

$G' = G - \{p\}$, contrary to the assumption that G is a gang and $p \in G$.

Since $\Xi_{epi}(S) \models C$ implies $\Xi_{epi}(S) \models p : \top \vee C$, the contraposition of (1) just assumed means that *every* $\top$-clause implied by $\Xi_{epi}(S)$ contains some member of G'. Consider an indictment E against p in S. Since E is inconsistent, by Lemma A2 it follows that there is a clause $C' = q_1 : \top \vee \ldots \vee q_n : \top$ such that $\Xi_{epi}(E) \models C'$ and E is an indictment against each one of the q_1, ..., q_n. Since $E \subseteq S$, it follows that $\Xi_{epi}(S) \models \Xi_{epi}(E)$, hence $\Xi_{epi}(S) \models C'$. By the above, one of the q_i must be a member of G'. Since E is an arbitrary indictment against p, p implicates G'.
(2) Suppose to the contrary that there is a $\top$-clause, C, such that $\Xi_{epi}(S) \models C$ but none of the members of C belongs to the gang G. Let S' be a subset of S consisting of clauses that are built only of those non-annotated atoms that occur in C. As in the earlier proofs, S' is a nonempty inconsistent set and $\Xi_{epi}(S') \models C$. By Lemma A2, S' is an indictment against at least one of the literals mentioned in C. On the other hand, it does not indict any of the gangsters in G, since none of these gangsters is mentioned in the clauses of S'. Thus, G is not implicated by every atom in S, contrary to the assumption that G is a gang. □

Appendix B: Proof of Completeness

We are following the proof of [4] with the modifications from [19]. First we have to adapt the definition of semantic trees to the model theory of APC.

Apart from the Herbrand base of APC, it will be also convenient to consider the set of ground atoms of PC that is obtained from the atoms in the Herbrand base by stripping off the annotations. We will call the resulting set a *pure* Herbrand base. Given a pure Herbrand base B over some language L and a belief semilattice BSL, a *semantic tree* T in APC is a possibly infinite tree such that:

(i) For every non-leaf node, ν, its outgoing edges stand in a 1-1 correspondence with the elements of $\{p_\nu : \alpha \mid \alpha \in BSL\}$, where p_ν is a pure ground atomic formula from B corresponding to ν. Each edge is labeled by the corresponding atom $p_\nu : \alpha$.

(ii) For each $p \in B$ and every branch of T, p labels exactly one edge on the branch (i.e., it appears only once with some annotation).

Note that in APC, semantic trees may have infinite branching factor since belief semilattices may be infinite. Thus, unlike in PC, semantic trees may be infinite even if the corresponding pure Herbrand base is finite.

Each branch of the tree is a truth assignment to the atoms of B and each such assignment determines a Herbrand interpretation of APC in which an atom $q : \alpha$ is true if and only if some $q : \lambda$, where $\alpha \leq \lambda$, appears on the branch. Conversely, each Herbrand interpretation of B corresponds to some branch. It is also clear that for every $\nu \in T$, the path from the root of T to ν determines a partial interpretation $T(\nu)$.

Given a set of ground clauses, S, a pure Herbrand base of S, B_S, and a corresponding semantic tree T_S, a *failure* node for S is a node $\nu \in T_S$ such that the partial interpretation $T_S(\nu)$ falsifies some clause in S and no node on the path from ν to root is a failure node for S. Thus, if ν is a failure node then every interpretation corresponding to the branches of T_S containing ν falsifies some clause from S.

If S is unsatisfiable and T_S is a semantic tree for S then each branch of T_S has a failure node. A *failure tree* for an unsatisfiable set S is obtained from a semantic tree for S by trimming off all branches beneath the failure nodes.

Theorem 2 (cf. Herbrand's theorem): Consider a set of possibly nonground clauses S that involves only a finite number of different belief annotations (this condition is always satisfied when BSL is finite, or when S itself is finite, e.g. a logic program). Then S is o-inconsistent if and only if so is some finite subset of its ground instances.

Proof: The "if-direction" is trivial. "Only if": Let S' be a set of ground instances of S and $T_{S'}$ be a corresponding semantic tree. Since S is unsatisfiable, so is S' and we can construct a failure tree $FT_{S'}$ out of $T_{S'}$. In the classic logic we could have stopped here since by Ko:nig's lemma it would follow that $FT_{S'}$ is finite and thus falsifies a finite number of clauses of S'. However, this argument is not immediately applicable to APC since the branching factor here may be infinite. To overcome this problem we need to do some extra work.

If the belief semilattice is finite then the branching factor in each node is also finite and we could use Ko:nig's lemma to conclude that $FT_{S'}$ is finite, as in predicate calculus. If BSL is infinite then consider some finite subsemilattice BSL_S of BSL containing all the annotations mentioned in S (BSL_S exists because the number of annotations in S is finite). Since S' is unsatisfiable with respect to BSL, it is unsatisfiable with respect to BSL_S. But then, by the above, some finite subset of S' is unsatisfiable with respect to BSL_S. Its unsatisfiability with respect to BSL follows from Lemma B1 below. □

Lemma B1: Let S be a set of ground clauses and BSL_S a *finite* subsemilattice of BSL containing all annotations in S[8] Then S is unsatisfiable with respect to BSL if and only if it is unsatisfiable with respect to BSL_S.

Proof: The "only if" direction is trivial. In the other direction, we need to show that unsatisfiability with respect to BSL_S implies unsatisfiability with respect to BSL. Suppose to the contrary that S is satisfied with respect to BSL. Then there is a Herbrand model M of S with respect to BSL. This implies that for each clause C in S there is a literal of C that is true in M. We will show that there is a model M_S with respect to BSL_S, such that every clause of S is true in M_S if and only if it is true in M.

We construct M_S as follows. Consider a non-annotated atomic formula p mentioned in a clause in S and let $p : \alpha_1$, ..., $p : \alpha_n$, $\neg p : \beta_1$, ..., $\neg p : \beta_m$ be all the literals that occur in S, share p as their non-annotated part, and are true in M. Let α denote lub of α_i, $i = 1, \ldots, n$. Since the above n positive literals are true in M, $p : \alpha$ should be true in M. However, since the m negative literals are also true, α must be strictly smaller or incomparable to β_j, $j = 1, \ldots, m$. Make $p : \alpha$ a true literal in M_S with α being the highest degree of belief in p. Clearly, all of the aforementioned literals are then true in M_S. Repeating this procedure for every atomic formula in S we obtain M_S that possesses the required property. □

Example 8 of Section 5 shows that the requirements of Herbrand's theorem are necessary. In general, Lemma B1 and Theorem 2 do not hold if BSL_S is infinite. Next, we establish completeness of resolution for the ground case.

Proposition B1: Suppose S is a possibly infinite unsatisfiable set of ground clauses such that there is a finite subsemilattice BSL_S that includes all annotations occurring in clauses in S. Then there is a refutation of S.

Proof: By Herbrand's Theorem we can assume that S is finite. Furthermore, we can always close S under all possible applications of resolution, factorization and reduction while preserving finiteness. So, we assume that S is closed in this sense. Since S is finite, it has a finite failure tree FT with respect to BSL_S. Call a node of FT an *inference node* if all its children are leaves of FT (i.e. failure nodes).

Any failure tree contains an inference node unless the tree consists of a single node. In the latter case, since a single-node failure tree can falsify only the empty clause, S must contain such a clause and we are done. So assume that FT consists of more than one node.

Let ν be an inference node, μ_1, ..., μ_n be all its children, and $p_\nu : \alpha_1$, ..., $p_\nu : \alpha_n$ be the labels on the edges $< \nu, \mu_1 >$, ..., $< \nu, \mu_n >$. We will arrive at a contradiction by showing that ν cannot be an inference node. Hence FT cannot have more than one node, so the empty clause must be in S.

Since the μ_is are failure nodes, let C_1, ..., C_n be the clauses of S falsified by interpretations $FT(\mu_1)$, ..., $FT(\mu_n)$, respectively. We assume that the elimination rule has been applied to remove

[8]Notice that since BSL_S is a subsemilattice of BSL, it also contains all the lub's of the annotations mentioned in S.

all literals of the form $\neg q : \bot$.

Since ν is not a failure node, each C_i must be of the form $C'_i \cup \{l_i\}$, where l_i is of the form $p_\nu : \gamma_i$ or $\neg p_\nu : \gamma_i$ and

(i) each C'_i is falsified by $FT(\nu)$,

(ii) $p_\nu : \alpha_i \not\models l_i$ or, equivalently, l_i is not satisfied in $FT(\mu_i)$.

Assume for definiteness that $l_i = p_\nu : \beta_i$, when $1 \leq i \leq s-1$ $(1 < s \leq n)$, and $l_i = \neg p_\nu : \beta_j$, when $s \leq i \leq n$. Notice that there has to be at least one positive and one negative literal among the l_i's, for if all of them were positive, take an edge labeled with $p_\nu : \top$, say $< \nu, \mu_{i_0} >$, and verify that $FT(\mu_{i_0})$ satisfies all the l_is and hence C_{i_0}, contrary to the assumption. Similarly, there is an edge $< \nu, \mu_{k_0} >$ labeled with $p_\nu : \bot$ and, if all the l_i were negative, $FT(\mu_{j_0})$ would have satisfied C_{j_0}.

Let C be the result of reducing $C_1, \ldots, C_{s-1}$ on p_ν. Denoting $\mathtt{lub}_{i=1}^{s-1}\beta_i$ by β, we can represent C as follows: $C = \cup_{i=1}^{s-1} C'_i \cup \{p_\nu : \beta\}$. Clearly, C is falsified by each of the $FT(\mu_i)$, where $i = 1, \ldots, s-1$. By the definition of semantic trees, there is an arc labeled with $p_\nu : \beta$, say $< \nu, \mu_k >$, where $k < s$, Since this arc satisfies all the l_is, $i < s$, it has to falsify some $l_{k_0} = \neg p_\nu : \beta_{k_0}$, $k_0 \geq s$.

Hence, $\beta \geq \beta_{k_0}$ and C can be resolved with C_{k_0} yielding $\bar{C}$ — a clause in which p_ν does not occur. But then $FT(\nu)$ falsifies $\bar{C}$. Hence, ν is a failure node for S, not an inference node — a contradiction. □

Lemma B2 (Lifting Lemma): Suppose C'_1 and C'_2 are instances of clauses C_1 and C_2, respectively. Let C'_3 be a resolvent (or reductant) of C'_1 and C'_2. Then there is a clause C_3 that is obtained by factorization and resolution (resp., reduction) from C_1 and C_2 such that C'_3 is an instance of C_3. Similarly, if C' is an instance of C and $\bar{C}'$ is a factor (or an eliminant) of C' then there is a factor (resp., eliminant) $\bar{C}$ of C such that $\bar{C}'$ is its instance.

Proof: The proof is identical to that of the lifting lemma for predicate calculus, except for the simple modification needed to account for the reduction rule and the annotations. This modification is straightforward, since annotations are constants drawn from BSL and variables are not allowed to range over them. □

Theorem 3: Refutation is a sound and complete procedure for testing o-inconsistency of sets of clauses that involve only a finite number of different annotations. That is, if S is such a set then it is o-inconsistent if and only if there is a refutation of S.

Proof: The proof proceeds as in predicate calculus [4]. Let S be a possibly infinite unsatisfiable collection of clauses with a finite number of different annotations and let S' be the set of ground instances of clauses in S, which is also unsatisfiable. By Proposition B1, there is a refutation of S' and by Lemma B2 this refutation can be lifted to S (note: the lift of the empty clause is also empty). As in predicate calculus, the need in factorization becomes apparent only at the point when one needs to lift ground refutations to the nonground ones. □

References

[1] K.R. Apt, H. Blair, and A. Walker. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann, Los Altos, CA, 1988.

[2] N. Belnap. A useful four-valued logic. In M. Dunn and G. Epstein, editors, *Modern Uses of Multi-Valued Logic*, pages 8–37. Reidel Publ. Co., 1977.

[3] H.A. Blair and V.S. Subrahmanian. Paraconsistent logic programming. *Theoretical Computer Science*, 68:135–154, 1989.

[4] C.L. Chang and R.C.T. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, 1973.

[5] W. Chen, M. Kifer, and D.S. Warren. HiLog: A foundation for higher-order logic programming. *Journal of Logic Programming*, 15(3):187–230, February 1993.

[6] N.C.A. da Costa. On the theory of inconsistent formal systems. *Notre Dame J. of Formal Logic*, 15(4):497–510, October 1974.

[7] H.B. Enderton. *A Mathematical Introduction to Logic.* Academic Press, 1972.

[8] R. Fagin and J.Y. Halpern. Belief, awareness, and limited reasoning. *Artificial Intelligence*, 34:39–76, 1988.

[9] M. Fitting. First-order modal tableaux. *Journal of Automated Reasoning*, 1988.

[10] M. Fitting. Negation as refutation. manuscript, 1988.

[11] M. Fitting. Bilattices and the semantics of logic programs. *Journal of Logic Programming*, 11(2):91–116, August 1991.

[12] C. Geissler and K. Konolige. A resolution method for quantified modal logics of knowledge and belief. In J.Y. Halpern, editor, *Theoretical Aspects of Reasoning about Knowledge*, pages 309–324, 1986.

[13] M.L. Ginsberg. Multivalued logics: A uniform approach to reasoning in Artificial Intelligence. *Computational Intelligence*, 4:265–316, 1988.

[14] S. Haack. *Philosophy of Logic.* Cambridge University Press, 1978.

[15] M. Kifer and A. Li. On the semantics of rule based expert systems with uncertainty. In *Int'l Conference on Database Theory*, volume 326 of *Lecture Notes in Computer Science*, pages 186–202. Springer-Verlag, September 1988.

[16] M. Kifer and V.S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *Journal of Logic Programming*, 12(4):335–368, April 1992.

[17] H.J. Levesque. A logic of implicit and explicit belief. In *National Conference on Artificial Intelligence*, pages 198–202, 1984.

[18] V. Lifschitz. On the declarative semantics of logic programs with negation. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 177–192. Morgan-Kaufmann, Los Altos, CA, 1988.

[19] D. Maier and D.S. Warren. *Computing with Logic: Logic Programming with Prolog.* Benjamin-Cummings, Menlo Park, CA, 1988.

[20] J. McCarthy. Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*, 28:89–116, 1986.

[21] J. Minker, editor. *Foundations of Deductive Databases and Logic Programming.* Morgan-Kaufmann, Los Altos, CA, 1988.

[22] R.C. Moore. A formal theory of knowledge and action. In J.R. Hobbs and R.C. Moore, editors, *Formal Theories of the Common-Sense World*, pages 319–354. 1986.

[23] D. Perlis. Circumscription as introspection. Technical report, Univ. of Maryland, 1987.

[24] G. Priest. Reasoning about truth. *Artificial Intelligence*, 39:231–244, 1989.

[25] T.C. Przymusinski. On the declarative semantics of deductive databases and logic programs. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 193–216. Morgan Kaufmann, Los Altos, CA, 1988.

[26] H. Rasiowa and G. Epstein. Approximate reasoning and scott's information systems. In *2-nd Int. Symposium on Methodologies for Intelligent Systems*, pages 33–42, 1987.

[27] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[28] N. Rescher and R. Brandom. *The Logic of Inconsistency.* Billing & Sons, Ltd., Oxford, UK, 1980.

[29] E. Snadewall. A functional approach to non-monotonic logic. In *Int'l Joint Conference on Artificial Intelligence*, pages 100–106, 1985.

[30] V.S. Subrahmanian. On the semantics of quantitative logic programs. In *IEEE Symposium on Logic Programming*, pages 173–182, 1987.

[31] V.S. Subrahmanian. Mechanical proof procedures for many-valued lattice-based logic programming. *Contemporary Mathematics*, 1989.

[32] D.S. Touretzky, J.F. Horty, and R.H. Thomason. A clash of intuitions: The current state of nonmonotonic multiple inheritance systems. In *Int'l Joint Conference on Artificial Intelligence*, pages 476–482, San Francisco, CA, 1987. Morgan Kaufmann.

[33] M.H. van Emden. Quantitative deduction and its fixpoint theory. *Journal of Logic Programming*, 1(4):37–53, 1986.