JOHN GRANT and V. S. SUBRAHMANIAN

APPLICATIONS OF PARACONSISTENCY IN DATA AND KNOWLEDGE BASES

ABSTRACT. The study of paraconsistent logic as a branch of mathematics and logic has been pioneered by Newton da Costa. With the growing advent of distributed and often inconsistent databases over the last ten years, there has been growing interest in paraconsistency amongst researchers in databases and knowledge bases. In this paper, we provide a brief survey of work in paraconsistent databases and knowledge bases affected by Newton da Costa's important and lasting contributions to the field.

1. INTRODUCTION

This article surveys applications of paraconsistency in data and knowledge bases. In recent years the handling of inconsistencies in knowledge bases has become a practical problem. Knowledge bases are often built by putting together information from various sources. In such a circumstance, it is quite likely that inconsistencies may arise. In fact, with the large amount of information available on many topics on the Internet, anyone gathering data from different sites on the same or a similar topic is likely to find some conflicting information.

Classical first-order logic is a very powerful tool for reasoning. However, it does not handle inconsistencies in a useful way, because every formula (of the language under consideration) can be deduced from a theory that contains an inconsistency. The term "paraconsistency" is often used to refer to various approaches that have been applied to deal with inconsistencies in a less global manner. One technique is to limit the rules of inference: this is followed by the various calculi $\mathcal{C}_n$. Another method extends the truth values from False, True to a larger lattice. A third approach stays with first-order logic but uses only (maximal) consistent subsets of an inconsistent theory. A fourth proposal extends first-order logic with various (typically) modal operators. We will deal primarily with the last 3 techniques because most of the work on paraconsistency in knowledge bases uses them.

We use the language of deductive databases, where a knowledge base consists of facts, rules, and integrity constraints. An inconsistency may be

Synthese **125:** 121–132, 2000.

caused in various ways, such as contradictory facts, a contradiction caused by some facts and additional facts derivable by rules, or by the interaction of integrity constraints with some facts and rules. We do not deal with database updates that can lead to inconsistencies; there is a substantial research literature on theory change and belief revision that deals with that issue.

Section 2 gives the notation used in this article. Section 3 considers the extension of truth values. We discuss here annotated programs as well as some special cases where it is not necessary to use annotated formulas. In Section 4 we use maximal consistent subsets of the knowledge base for our reasoning with inconsistency. The use of various operators is considered in Section 5. We also deal with preferences between formulas in this section. Section 6 is the summary.

2. NOTATION

In the research literature several definitions have been used for the concept of a knowledge base. For example, some authors consider a knowledge base to be a set of literals in a propositional language. In this paper we use for our framework the terminology of deductive databases. We assume that a first-order language L exists which contains the predicate and constant symbols needed to represent the knowledge base. Our results apply to any such language.

A deductive database is often considered to have three components: facts, rules, and integrity constraints. The rules are used to deduce additional facts. The integrity constraints represent semantic information about the knowledge base. All these components of a deductive database are expressed by means of clauses of L. A clause has the form

$$A_1, \ldots, A_n \leftarrow B_1, \ldots, B_k,$$

where the A_i and B_j may be literals (atoms or negated atoms) of L.The $A_1, \ldots, A_n$ part is called the *head* of the clause; the $B_1, \ldots, B_k$ part is called the *body* of the clause. The variables are assumed to be universally quantified and all variables that appear in the head must appear in the body. In the simplest case, for *Datalog* programs, $n = 1$, and the $A_1, B_1, \ldots, B_k$ are all atomic formulas.

For facts, $k = 0$; that is, facts have an empty body. For rules, $k \geq 1$. For integrity constraints, $n = 0$, that is, integrity constraints have an empty head. An integrity constraint signifies that the conjunction of $B_1, \ldots, B_n$ is not allowed. The case where $n > 1$ is allowed is called a *disjunctive*

database. If the B_i may contain negation, the database is called *normal*. Negation in the body is usually treated as *default* negation. A program is *stratified* if recursion through negation is not allowed.

Knowledge bases store information and answer queries posed by users. A query, like an integrity constraint, is expressed as a clause with an empty head. Such a query, rewritten here as $\leftarrow Q(x_1, \ldots, x_n)$, to emphasize that its free variables are $x_1, \ldots, x_n$, is interpreted to mean: Find all tuples $\langle a_1, \ldots, a_n \rangle$ such that $Q(a_1, \ldots, a_n)$ is true in the knowledge base. In particular, a query with no variables is a yes-no query. This brings up the question of knowledge base semantics: what is meant by saying that a formula is true.

The facts and rules provide syntactic information about the knowledge base. The semantics is given by interpretations. An interpretation I consists of a domain D and predicates and constants on D interpreting the predicate and constant symbols of L. We restrict our consideration to Herbrand interpretations, where D contains exactly one constant for each constant symbol of L. This is reasonable for knowledge bases, because all the relevant constants are usually included in the facts. We write B_P for the *Herbrand base* (variable-free atoms) of the knowledge base P. The truths of formulas in an interpretation are defined in the standard way. An interpretation in which each formula of the knowledge base is true is called a model. A model that does not include another model is called *minimal*.

For Datalog programs without integrity constraints the semantics is clear because there is a unique minimal model which is the intended model. So we can take a formula to be true if it is true in this model. Hence the true statements of the knowledge base are the true statements of the minimal model. This approach to semantics, dealing with models, is called *declarative semantics*. For the purpose of computing answers to queries, two other types of semantics are important: fixpoint semantics and procedural semantics. *Fixpoint semantics* applies an operator called T_p on Herbrand interpretations; the fixpoint of this operator is then taken as the meaning of the knowledge base. *Procedural semantics* refers to a computational method, using resolution in some form, to obtain answers to queries. An important aspect of the work on knowledge base semantics has centered on finding appropriate definitions for these semantics so that they provide identical answers to queries.

However, the clarity of the semantics is lost when disjunction or negation is allowed in the clauses, because there may not be a unique model. In such a case the meaning of the answers to a query may be ambiguous. In fact, researchers have proposed various alternative semantics such as the well-founded and stratified semantics: different semantics may give differ-

ent answers to a query. Within each approach there may be a declarative, fixpoint, and procedural semantics. The situation is greatly complicated in case the knowledge base is inconsistent, for in this case standard logic provides no models at all. The paraconsistent approaches, as we will see, work around this problem in order to obtain meaningful answers to queries in those cases.

3. EXTENDING THE TRUTH VALUES

We consider the case where the classical two truth values are extended to many values. First we explore the concept of annotated logic programs and then we consider a special case, the over-determined semantics in more detail.

3.1. *Annotated Logic Programs*

Classical logic has two truth-values: **true** and **false**. In some applications, particularly in the presence of inconsistencies, it is useful to have a larger set of truth-values. In annotated logic each literal of a clause is annotated by a truth value. We start by fixing the set of truthvalues, $\mathcal{T}$. We assume that $\mathcal{T}$ is a complete lattice under an ordering $\leq$ on $\mathcal{T}$. This means that every subset S of $\mathcal{T}$ has a least upper bound (lub), $\sqcup S$, and a greatest lower bound (glb), $\sqcap S$. Additionally, we assume the existence of a function for negation: $\neg : \mathcal{T} \rightarrow \mathcal{T}$.

In particular, **t**, **f** is such a lattice with **f** $<$ **t**. Another very useful such lattice, **FOUR**, consists of $\{\top$, **t**, **f**, $\bot\}$, where $\bot <$ **t**, **f** $< \top$. Intuitively, $\bot$ denotes "unknown" and $\top$ denotes "inconsistent". This important lattice will get special consideration in the next subsection. The set of reals, [0,1], is also a complete lattice under the usual $\leq$. Here, 0 denotes false, 1 denotes true, and the real numbers between 0 and 1 may denote a degree or probability of truth. Another lattice is [0,1] $\times$ [0,1], where $\langle \mu_1, \mu_2 \rangle$ is interpreted to mean: degree of belief μ_1, degree of disbelief μ_2, and $[\mu_1, \mu_2] \leq [\rho_1, \rho_2]$ if $\mu_1 \leq \rho_1$ and $\mu_2 \leq \rho_2$. Here, [1,1] is the truth value of an absolutely inconsistent belief. A literal L annotated by a truth value $\mu \in \mathcal{T}$ is written as $L : \mu$. The annotated literals $L_0, \ldots, L_n$ form the *annotated clause*

$$L_0 \leftarrow L_1, \ldots, L_n.$$

An *annotated logic program* (ALP) is a finite set of annotated clauses.

The semantics of ALPs is defined by interpretations. In classical logic an interpretation assigns to each atom (of the Herbrand base) the value

true or *false*; for ALP, each atom is assigned an element of $\mathcal{T}$.The $\leq$ ordering on $\mathcal{T}$ extends to an ordering on interpretations in a natural way. An interpretation I is said to satisfy the annotated atom $A : \mu$ if $I(A) \geq \mu$; similarly, I is said to satisfy the annotated literal $\neg A : \mu$ if $I(A) \geq \neg\mu$. An interpretation is extended to the connectives and quantifiers and the concept of model are defined in the usual way. It turns out that if annotated literals $\neg A : \mu$ are changed to annotated atoms $A : \neg\mu$, the status of an interpretation as a model does not change. Hence we can assume that ALPs contain only annotated atoms.

Many concepts of logic programming (and knowledge bases) can be extended to ALPs. In particular, every ALP has a least model which can be taken as the intended model. This model can also be obtained by fixpoint semantics. Additionally, the usual procedural semantics, SLD-resolution, can be extended to ALPs to answer queries.

While the original work on paraconsistent logic programs only looked at a syntactic fragment of the logic, two important efforts extended this to a full first order logic. First, da Costa, Subrahmanian and Vago (1991) extended annotated frameworks and provided a rich model theory and proof theory for them. Later, Abe, da Costa and Subrahmanian (1991) extended the framework further to describe annotated set theory as well. Kifer and Lozinskii (1989) extended the framework to include alternative forms of negation in a pioneering paper. In recent years, the idea of incorporating multiple modes of negation into a logic program has led to a host of work on paraconsistent logics neatly summed up in Damasio and Pereira (1998).

3.2. *The Over-Determined Semantics*

The lattice **FOUR** is an important intuitively simple lattice for use in annotated logic programs. In fact, it can be studied without using the machinery of annotations simply by using the values $\top$ and $\bot$ implicitly. So, in this case, the interpretation I of a relation R, $I(R)$, is represented as a pair $R = \langle R^+, R^- \rangle$, where R^+ is the set of tuples true in R and R^- is the set of tuples false in R. Then, for an n-ary relation R, the set of tuples of D^n can be divided into 4 types:

1. $R_1 = R^+ \cap R^-$: these are the tuples with truth value $\top$.
2. $R_2 = R^+ - R^-$: these are the tuples with truth value **t**.
3. $R_3 = R^- - R^+$: these are the tuples with truth value **f**.
4. $R_4 = D^n - R^+ - R^-$: these are the tuples with truth value $\bot$.

A relation is *inconsistent* if $R_1 \neq \emptyset$. As studied in Bagai and Sunderraman (1995) the usual algebraic operators on relations can be generalized to

such interpretations. Computing answers to queries can then be performed by writing the queries as relational algebra programs.

A much more elaborate approach is given in Wagner (1993) where two types of negation are used: explicit negation and implicit negation. The tuples in R^- are explicitly negated for R: they are known to be false. The tuples in $D^n - R^+$ are implicitly negated for R: they are not known to be true. These concepts lead to four different notions of model. The concept of *liberal* model corresponds to the models with truth values in **FOUR**. The others are increasingly restrictive and called *credulous*, *conservative*, and *skeptical*.

There are some disadvantages in reasoning with the semantics of **FOUR**. One problem is that the Law of Excluded Middle does not hold. This is due to the existence of incomplete relations where some tuples may have the truth value $\perp$ for a relation. If we wish to use this law in our reasoning, then this semantics is not appropriate.

A way to get around this problem is to omit $\perp$ and use instead of the lattice **FOUR** its 3-valued upper semilattice. This allows for inconsistency but not incompleteness; so for example, the Law of Excluded Middle holds. This semantics was introduced in Grant (1978) and further studied in Grant and Subrahmanian (1995a). Treating it as a knowledge base, the difference in the causal form from Datalog programs is that any of the $A_1, B_1, \ldots, B_k$ may be either an atomic formula or the negation of an atomic formula. In an *OD-interpretation* I, for every n-ary relation R and n-tuple $\langle a_1, \ldots, a_n \rangle$, either $I \models_{\mathbf{od}} R(a_1, \ldots, a_n)$ or $I \models_{\mathbf{od}} \neg R(a_1, \ldots, a_n)$, or both.

An interpretation over the literals is expanded over all formulas by using the standard rules for connectives and quantifiers. In particular, $I \models_{\mathbf{od}} F \leftarrow G$ if either $I \models_{\mathbf{od}} F$ or $I \not\models_{\mathbf{od}} G$. Some important laws of logic continue to hold in the OD-semantics.

Let **A**, **B**, C be metavariables ranging over ground atoms. Then the following axiom schemes are valid in the OD-semantics:

1. (Law of Excluded Middle) $\mathbf{A} \vee \neg \mathbf{A}$.
2. $(\mathbf{A} \vee \mathbf{B}) \leftarrow (\mathbf{A} \leftarrow \neg \mathbf{B})$.
3. (Reasoning with Cases) $\mathbf{A} \leftarrow ((\mathbf{A} \leftarrow \mathbf{B}) \,\&\, (\mathbf{A} \leftarrow \mathbf{C}) \,\&\, (\mathbf{B} \vee \mathbf{C}))$.
4. (Resistance to Inconsistency) It is not the case that $(\mathbf{A} \,\&\, \neg \mathbf{A}) \models_{\mathbf{od}} \mathbf{B}$.

Grant and Subrahmanian (1995a) also provide a computation procedure similar to the T_p operator of fixpoint semantics.

4. REASONING WITH MAXIMAL CONSISTENT SUBSETS

Suppose we are given an inconsistent knowledge base, most of which is reliable information, and we would like to use standard logic for reasoning. We can take for the meaning of the knowledge base the set of its maximal consistent subsets and reason in the usual way with those subsets. First we explain the optimistic and cautious semantics. Then we show how the concept of maximal consistent subsets can be applied to combine multiple knowledge bases which contain integrity constraints.

4.1. *Optimistic and Cautious Semantics*

The optimistic and cautious semantics are explored in detail in Grant and Subrahmanian (1995b). Let P be a knowledge base that may be inconsistent and F a formula of L. We define

1. $P \vdash_\exists F$ iff there is some maximal consistent subset $P' \subseteq P$ such that $P' \models F$.
2. $P \vdash_\forall F$ iff $P' \models F$ for every maximal consistent subset $P' \subseteq P$.

The semantics using $\vdash_\exists$ is called the *optimistic* semantics. According to this semantics, a statement that is true in *any* maximal consistent subset is taken to be true. The optimistic semantics has some interesting properties. For example,

$$\text{if } P \vdash_\exists F_1 \& F_2, \text{ then } P \vdash_\exists F_1 \text{ and } P \vdash_\exists F_2.$$

This is so because for any maximal consistent subset P' such that $P' \models F_1 \& F_2$, we must have $P' \models F_1$ and $P' \models F_2$. However, the converse is not true: it may be that $P \vdash_\exists F$ and $P \vdash_\exists \neg F$, because F and $\neg F$ are true in different maximal consistent subsets, but no maximal consistent subset can have $F \& \neg F$ true.

The semantics using $\vdash_\forall$ is called the *cautious* semantics. According to this semantics a statement is taken to be true only if it is true in *all* maximal consistent subsets. With the cautious semantics we get $P \vdash_\forall F_1 \& F_2$ iff $P \vdash_\forall F_1$ and $P \vdash_\forall F_2$. Also, with the cautious semantics , there cannot be a formula F such that $P \vdash_\forall F$ and $P \vdash_\forall \neg F$. In general, $P \vdash_\forall F$ implies $P \vdash_\exists F$ and if P is consistent, the concepts $\models$, $\vdash_\exists$, and $\vdash_\forall$ coincide. A fixpoint operator is defined for optimistic semantics and a Kripke style semantics using the modal operators $\Diamond$ (possible) and $\Box$ (necessary) is developed for both the optimistic and cautious semantics.

4.2. *Knowledge Bases with Integrity Constraints*

Up to this point we have dealt only with the facts and rules of the knowledge base. Now we consider integrity constraints. When knowledge bases are combined, it is possible that the union of the facts and rules is consistent, yet the resulting knowledge base is inconsistent because of integrity constraints. We review here some of the results of Baral et al. (1991, 1992) where this problem is investigated by the use of maximal consistent subsets.

We start by introducing some terminology. We assume that all the clauses of the knowledge base P are ground, i.e., have no variables and that P is stratified. For clauses with variables we just make all possible substitutions by constants to obtain all ground clauses. The semantics of a knowledge base is given by its minimal models. In the definition of the *perfect model semantics*, a relation $\leq$ is defined between the atoms of B_P as follows:

1. $C < B$ if there is a clause in P with $\neg B$ in the body and C in the head.
2. $C \leq B$ if there is a clause in P with B in the body and C in the head.
3. $C \leq B$ and $B \leq C$ if there is a clause in P with both B and C in the head.

Transitivity is then used to extend $\leq$ to a partial order.

For minimal models M and N we say that N is *preferable* to M if $\forall A \in N - M \exists B \in M - N$ such that $A < B$. The intuition here is that for the clause $C \leftarrow \neg B$ we prefer the minimal model C to the minimal model B. Using this concept we define the set of preferred minimal models, $PER(P)$ by eliminating those minimal models that have a preferred counterpart. We take $PER(P)$ for the semantics of P.

We say that a knowledge base P is *consistent* with a set of integrity constraints IC if IC is true in every model of $PER(P)$. We also define a partial order $\leq$ between knowledge bases: $P_i \leq P_j$ if $\forall M \in PER(P_i) \exists N \in PER(P_j)$ such that $M \subseteq N$. Among a set $P_1, \ldots, P_n$, P_i is called *maximal*if there is no P_j with $i \neq j$, such that $P_i \leq P_j$. Also, P is called *correct* with respect to $P_1, \ldots, P_n$ if $P \leq P_1 \cup \ldots \cup P_n$. The goal is to combine a set of knowledge bases $P_1, \ldots, P_n$ to P in such a way that P is maximal among all consistent and correct combinations of $P_1, \ldots, P_n$. In Baral et al. (1991) algorithms are given that yield such a maximal combination for various cases of stratified logic programs.

The case considered in Baral et al. (1992) allows negated atoms in the head of a clause. In such a case, even without considering integrity constraints, the union of a set of consistent knowledge bases need not be consistent; for instance, one may contain $A \leftarrow$ and the other $\neg A \leftarrow$. As

explained in the previous subsection, we reason by using maximal consistent subsets of the union of knowledge bases. But then the question is how to choose the maximal consistent subsets. We give integrity constraints preferential treatment by insisting that IC must be included in every maximal consistent subset. One way to accomplish this result is to take all the consistent subsets of $P_1 \cup \ldots \cup P_n \cup IC$ that contain IC. Another possibility is to take the maximal consistent subsets of the union without IC and then apply IC to each such set and take the maximal consistent sets for the result. It turns out that these two approaches yield the same result. A third approach takes the maximal consistent subsets of the union as in the second case without IC and eliminates any such set not consistent with IC. This approach yields a different result.

5. EXTENDING THE LOGIC

5.1. *Modal Logic*

We already mentioned that in Grant and Subrahmanian (1995b) a modal logic is used to describe the optimistic and cautious semantics. A more elaborate approach to the use of modal logic for answering queries in an inconsistent knowledge base is presented in Cholvy (1998). In that modal system, called Q', the axioms about $\Box$ are:

$$\Box\neg F \rightarrow \neg\Box F \text{ and } \Box F \wedge \Box(F \rightarrow G) \rightarrow \Box G.$$

$\Diamond$ is defined as $\neg\Box\neg$.

The inference rules are Modus Ponens and Necessitation: $\vdash A \Rightarrow \vdash \Box A$.

The two modalities used for answering queries are $\mathcal{S}$ and $\mathcal{D}$ as follows:

$$\mathcal{S}F = \Box F \text{ and } \mathcal{D}F = \Diamond F \wedge \neg\Box F.$$

$\mathcal{S}F$ is interpreted to mean that F's truth is sure; $\mathcal{D}F$ is interpreted to mean that F's truth is doubtful.

As far as the knowledge base KB is concerned, the approach here is to find all minimal inconsistent subsets of KB, $I_1, \ldots, I_m$, and then let $C = KB - \bigcup_{i=1}^{m} I_i$. The formulas in C are not in any minimal inconsistent subset, hence they are taken to be *sure*. The formulas in the maximal consistent subsets of $\bigcup_{i=1}^{m} I_i$. are taken to be *doubtful*.The answers to a query are divided into 2 groups: the sure answers and the doubtful answers. We also get 2 groups for the tuples that are not answers: the tuples that are surely not answers, and all other tuples.

5.2. *Priorities*

In Section 4.2 we considered the problem of inconsistency that arises when consistent knowledge bases with integrity constraints are combined. Such a combination can be achieved by using maximal consistent subsets. The approach presented there treats all formulas and atoms of the knowledge base on an equal basis. However, in many applications there may be a preference of some formulas or atoms over others that should be taken into consideration. A formal treatment of combining databases with preference appeared recently in Pradhan et al. (1995). We do not discuss here another method, presented in Naqvi and Rossi (1990) where earlier information is superceded by later information and a procedural semantics is used.

In this work each database consists of a set of propositional atoms and a set of integrity constraints. Let $D_1, \ldots, D_n$ be the databases and IC the integrity constraints. A *priority* $>$ is defined between an atom A and a set of atoms $Y = B_1, \ldots, B_m$, as $A > Y$, meaning that the statement A is preferred over the set of statements Y. Then, if $X = A_1, \ldots, A_n$, $X > Y$ stands for $A_1 > Y, \ldots, A_n > Y$.The priority relation is assumed to be irreflexive but not necessarily transitive.

For the integrity constraint $IC_i :\leftarrow A_1, \ldots, A_k$, we define $\text{Obj}(IC_i) = \{A_1, \ldots, A_k\}$. Clearly, a database D satisfies IC if $\text{Obj}(IC) \not\subseteq D$. For a set of databases $D_1, \ldots, D_n$ and integrity constraints IC, an *option* is a maximal subset of $D_1 \cup \ldots \cup D_n$ that satisfies every element of IC. An option D contains an *A-block* if there is $X \subseteq D$ and $IC_i \in IC$ such that $\text{Obj}(IC_i) = A \cup X$, i.e., the addition of A causes the violation of an integrity constraint. Then a database D is said to *satisfy the priority* $A > Y$ if either $A \in D$ or $Y \cap D = \emptyset$ or $D - Y$ contains an A-block. We write O for the set of options with respect to a given set of databases and integrity constraints.

Consider now the case where an option $D \in O$ does not satisfy the priority $A > Y$. There are four ways in which O can be modified to reflect the priority. First, the *option elimination* semantics removes D from O. Second, the *option ordering* semantics imposes a ranking on members of O that gives D a lower status. It turns out that the formal definitions of these two semantics yield equivalent results. The third method, *option transformation* transforms D to satisfy the priority. For a set of acyclic priorities all three semantics can be shown to be equivalent. The fourth semantics, *option amplification*, adds an atom to satisfy an unsatisfied priority. This semantics is different from the other three.

5.3. *Introspection*

The approach of You et al. (1995) uses inconsistency as a meta-level concept. Two introspective operators are added to the language **C** and $\mathbf{C}_d$. For a formula A, $\mathbf{C}A$ means that A is contradictory because both A and $\neg A$ are derivable. The meaning of $\mathbf{C}_d A$ is that A is derivable from inconsistent premises. These operators can then be used to construct additional formulas. A model theoretic semantics is developed, and it is shown that every positive non-disjunctive program has a unique para-model. The intuitiveness of the definitions and their applications are shown by various examples where this approach gives more informative information than other approaches.

6. SUMMARY

We surveyed applications of paraconsistency in data and knowledge bases. We considered work done in extending the set of truth values, using maximal consistent subsets of an inconsistent knowledge base, and extending first-order logic with additional operators. The proper handling of paraconsistency remains an important issue in working with large or combined data and knowledge bases (Subrahmanian 1994).

ACKNOWLEDGEMENT

This work was supported by the Army Research Office under Grants DAAH-04-95-10174, DAAH-04-96-10297, and DAAH04-96-1-0398, by the Army Research Laboratory under contract number DAAL01-97-K0135, by an NSF Young Investigator award IRI-93-57756, and by a TASC/DARPA grant J09301S98061.

REFERENCES

Abe, J., N. C. A. da Costa, and V. S. Subrahmanian: 1991, 'Remarks on Annotated Logics', *Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik* **37**, 561–570.

Baral, C., S. Kraus, and J. Minker: 1991, 'Combining Multiple Knowledge Bases', *IEEE TKDE* **3**(2), 208–220.

Baral, C., S. Kraus, J. Minker, and V. S. Subrahmanian: 1992, 'Combining Knowledge Bases Consisting of First Order Theories', *Computational Intelligence* **8**(1), 45–71.

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

Bagai, R. and R. Sunderraman: 1995, 'A Paraconsistent Relational Data Model', *Inter. J. of Computer Math.* **55**, 39–55.

Cholvy, L.: 1998, 'A General Framework for Reasoning about Contradictory Information and Some of its Applications', *Proceedings of ECAI Workshop "Conflicts Among Agents"*, Brighton, U.K.

da Costa, N. C. A. and V. S. Subrahmanian: 1990, 'Paraconsistent Logics as a Formalism for Reasoning about Inconsistent Knowledge Bases', *Journal of Artificial Intelligence in Medicine* **1**(4), 167–174.

da Costa, N. C. A., V. S. Subrahmanian, and C. Vago: 1991, 'The Paraconsistent Logics $\mathbf{P}\mathcal{T}$', *Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik* **37**, 139-148.

Damasio, C. and L. M. Pereira: 1998, 'A Survey of Paraconsistent Semantics for Logic Programs', in D. M. Gabbay and P. Smets (eds), *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, Vol. II, Kluwer Academic, Dordrecht, pp. 241–320.

Grant, J.: 1978, 'Classifications for Inconsistent Theories', *Notre Dame J. of Formal Logic* **XIX**(3), 435–444.

Grant, J. and V. S. Subrahmanian: 1995a, 'Reasoning About Inconsistent Knowledge Bases', *IEEE TKDE* **7**(1), 177–189.

Grant, J. and V. S. Subrahmanian: 1995b, 'The Optimistic and Cautious Semantics for Inconsistent Knowledge Bases', *Acta Cybernetica* **12**(1), 37–55.

Kifer, M. and E. L. Lozinskii: 1989, 'RI: A Logic for Reasoning with Inconsistency', *Proc. LICS 1989*, pp. 253–262.

Kifer, M. and V. S. Subrahmanian: 1989, 'Theory of Generalized Annotated Logic Programming and its Applications', *Journal of Logic Programming* **12**, 335–368.

Naqvi, S. A. and F. Rossi: 1990, 'Reasoning in Inconsistent Databases', *Proc. of 1990 NACLP*, pp. 255–272.

Pradhan, S., J. Minker, and V. S. Subrahmanian: 1995, 'Combining Databases with Prioritized Information', *Journal of Intelligent Information Systems* **4**, 231–260.

Subrahmanian, V. S.: 1994, 'Amalgamating Knowledge Bases', *ACM Transactions on Database Systems 19*(2), 291–331.

Wagner, G.: 1993, 'Reasoning with Inconsistency in Extended Deductive Database', in M. Pereira and A. Nerode (eds), *Proc. 2nd LPNMR Workshop*, MIT Press, Cambridge, MA.

You, J.-H., S. Ghosh, L.-Y. Yuan, and R. Goebel: 1995, 'An Introspective Framework for Paraconsistent Logic Programs', *Proc. of Intern. Symp. on Logic Programming*, pp. 384–398.

John Grant
Department of Computer and Information Sciences and
Department of Mathematics,
Towson University, Towson, MD
U.S.A.

V. S. Subrahmanian
Department of Computer Science,
Institute for Advanced Computer Studies, and
Institute for Systems Research,
University of Maryland MD
U.S.A.
E-mail: jgrant@towson.edu