A Paraconsistent Relational Data Model

Rajiv Bagai and Rajshekhar Sunderraman
Department of Computer Science
The Wichita State University
Wichita, KS 67260-0083
U.S.A.

Tel: +1-316-689-3156
Fax: +1-316-689-3984
Email: {*bagai,raj*}*@cs.twsu.edu*

Abstract

We present a generalisation of the relational data model based on a 4-valued paraconsistent logic. Our data model is capable of manipulating incomplete as well as inconsistent information. For this model, we define algebraic operators that are generalisations of the usual operators, such as union, selection, join, on ordinary relations. Our data model can underlie any database management system that deals with incomplete or inconsistent information. As another application of our model and its algebra, we present a bottom-up method for constructing the weak well-founded model of general deductive databases. This method can be very simply extended to construct the well-founded model.

Keywords: Paraconsistent relations, Relational algebra, General deductive databases, Weak well-founded model.

C.R. Categories: H.2.1, F.3.2, F.4.1, I.2.3.

1 Introduction

Two important features of the relational data model [5] for databases are its value-oriented nature and its rich set of simple, but powerful algebraic operators. Moreover, a strong theoretical foundation for the model is provided by the classical first-order logic [21]. This combination of a respectable theoretical platform, ease of implementation and the practicality of the model resulted in its immediate success, and the model has enjoyed being used by many database management systems. Even nonrelational systems are often described, for commercial purposes, as supporting relational features.

One limitation of the relational data model, however, is its lack of applicability to nonclassical situations. These are situations involving incomplete, or more importantly, even inconsistent information.

Representing and manipulating various forms of incomplete information in relational databases has been a concern of researchers since the very introduction of the relational model. *Null values* in relational databases [18] have been studied extensively. Codd [6] distinguished between *applicable* and *inapplicable* null values. Grant [12] generalised the concept of applicable nulls by introducing *partial values*. Imielinski and Lipski [13] explored *marked nulls* in relations that allow equating two null values. *Disjunctive information* has been studied in [15, 16, 23]. *Maybe information* in the context of null values was the focus of [3]. Lipski [14] distinguished between definite and maybe knowledge in the context of partial information. A study of *fuzzy* and *uncertain* information has been conducted in [11, 20].

However, unlike incomplete information, inconsistent information has not enjoyed enough research attention. While it may be argued that true knowledge systems should contain no inconsistent information, contradictions are very common in belief systems. Even experts of a domain often disgaree with each other, sometimes strongly.

Logics dealing with inconsistent information are called *paraconsistent* logics, and were studied in detail by de Costa [7] and Belnap [2]. Blair and Subrahmanian [4] proposed logic programming based on paraconsistent logic and Subrahmanian [22] extended the work to disjunctive deductive databases.

In this paper, we present a generalisation of the relational data model. Our model is based on the 4-valued paraconsistent logic of [2] and is capable of manipulating incomplete as well as inconsistent information. The incompleteness is at the tuple level, in that whether or not a particular tuple belongs to a relation may not be known. This notion of incompleteness is different from the other null-values related notions mentioned earlier. Similarly, inconsistency is also at the tuple level, in that a particular tuple may be considered to be both in and out of a relation.

We introduce *paraconsistent relations*, which are the fundamental mathematical structures underlying our model. A paraconsistent relation essentially contains two kinds of

tuples: ones that definitely belong to the relation and others that definitely do not belong to the relation. These structures are strictly more general than ordinary relations, in that for any ordinary relation there is a paraconsistent relation with the same information content, but not *vice versa*. We define algebraic operators over paraconsistent relations that extend the standard operators, such as selection, join, union, over ordinary relations.

In addition to answering queries in databases, we show another application of our algebra on paraconsistent relations. We present a bottom-up method to construct the *weak well-founded* model of general deductive databases; this model was proposed by Fitting in [9].

The rest of the paper is organised as follows. Section 2 introduces paraconsistent relations and two notions of generalising the usual relational operators, such as union, join, projection, for these relations. Section 3 presents some actual generalised algebraic operators for paraconsistent relations. These operators can be used for specifying queries for database systems built on such relations. As another interesting application of these operators, Section 4 gives a method for costructing the weak well-founded model of general deductive databases. An important step in the construction is to translate the database clauses into expressions involving the algebraic operators on paraconsistent relations. Finally, Section 5 contains some concluding remarks and directions for future work.

2 Paraconsistent Relations

In this section, we construct a set-theoretic formulation of paraconsistent relations. Unlike ordinary relations that can model worlds in which every tuple is known to either hold a certain underlying predicate or to not hold it, paraconsistent relations provide a framework for incomplete or even inconsistent information about tuples. They naturally model *belief* systems rather than *knowledge* systems, and are thus a generalisation of ordinary relations. The operators on ordinary relations can also be generalised for paraconsistent relations. However, any such generalisation of operators should maintain the belief system intuition behind paraconsistent relations. This section also develops two different notions of operator generalisations.

Let a *relation scheme* (or just *scheme*) Σ be a finite set of *attribute names*, where for any attribute name $A \in \Sigma$, $dom(A)$ is a non-empty *domain* of values for A. A *tuple* on Σ is any map $t : \Sigma \to \cup_{A \in \Sigma} dom(A)$, such that $t(A) \in dom(A)$, for each $A \in \Sigma$. Let $\tau(\Sigma)$ denote the set of all tuples on Σ.

Definition 1 An *ordinary relation* on scheme Σ is any subset of $\tau(\Sigma)$. We let $\mathcal{O}(\Sigma)$ be the set of all ordinary relations on Σ. $\square$

The above is the usual definition of relations. We call them 'ordinary' relations to distinguish them from other kinds of relations introduced below.

Definition 2 A *paraconsistent relation* on scheme Σ is a pair $R = \langle R^+, R^- \rangle$, where R^+ and R^- are any subsets of $\tau(\Sigma)$. We let $\mathcal{P}(\Sigma)$ be the set of all paraconsistent relations on Σ. $\Box$

Intuitively, R^+ may be considered as the set of all tuples for which R is believed to be true, and R^- the set of all tuples for which R is believed to be false. Note that since contradictory beliefs are possible, we do not assume R^+ and R^- to be mutually disjoint, though this condition holds in an important class of paraconsistent relations. As paraconsistent relations may contain contradictory information, they model belief systems more naturally than knowledge systems. Furthermore, R^+ and R^- may not together cover all tuples in $\tau(\Sigma)$.

Definition 3 A paraconsistent relation R on scheme Σ is called a *consistent relation* if $R^+ \cap R^- = \emptyset$. We let $\mathcal{C}(\Sigma)$ be the set of all consistent relations on Σ. Moreover, R is called a *complete relation* if $R^+ \cup R^- = \tau(\Sigma)$. If R is both consistent and complete, i.e. $R^- = \tau(\Sigma) - R^+$, then it is a *total relation*, and we let $\mathcal{T}(\Sigma)$ be the set of all total relations on Σ. $\Box$

It should be observed that (the positive parts of) total relations are essentially the ordinary relations. We make this relationship explicit by defining a one-one correspondence $\lambda_\Sigma : \mathcal{T}(\Sigma) \to \mathcal{O}(\Sigma)$, given by $\lambda_\Sigma(\langle R^+, R^- \rangle) = R^+$. This correspondence is used frequently in the following discussion.

Operator Generalisations

It is easily seen that paraconsistent relations are a generalisation of ordinary relations, in that for each ordinary relation there is a paraconsistent relation with the same information content, but not *vice versa*. It is thus natural to think of generalising the operations on ordinary relations, such as union, join, projection etc., to paraconsistent relations. However, any such generalisation should be intuitive with respect to the belief system model of paraconsistent relations. We now construct a framework for operators on both kinds of relations and introduce two different notions of the generalisation relationship among their operators.

An n-ary *operator on ordinary relations with signature* $\langle \Sigma_1, \ldots, \Sigma_{n+1} \rangle$ is a function $\Theta : \mathcal{O}(\Sigma_1) \times \cdots \times \mathcal{O}(\Sigma_n) \to \mathcal{O}(\Sigma_{n+1})$, where $\Sigma_1, \ldots, \Sigma_{n+1}$ are any schemes. Similarly, an n-ary *operator on paraconsistent relations with signature* $\langle \Sigma_1, \ldots, \Sigma_{n+1} \rangle$ is a function $\Psi : \mathcal{P}(\Sigma_1) \times \cdots \times \mathcal{P}(\Sigma_n) \to \mathcal{P}(\Sigma_{n+1})$.

Definition 4 An operator Ψ on paraconsistent relations with signature $\langle \Sigma_1, \ldots, \Sigma_{n+1} \rangle$ is *totality preserving* if for any total relations $R_1, \ldots, R_n$ on schemes $\Sigma_1, \ldots, \Sigma_n$, respectively, $\Psi(R_1, \ldots, R_n)$ is also total. $\Box$

Definition 5 A totality preserving operator Ψ on paraconsistent relations with signature $\langle \Sigma_1, \ldots, \Sigma_{n+1} \rangle$ is a *weak generalisation* of an operator Θ on ordinary relations with the same signature, if for any total relations $R_1, \ldots, R_n$ on schemes $\Sigma_1, \ldots, \Sigma_n$, respectively, we have

$$\lambda_{\Sigma_{n+1}}(\Psi(R_1, \ldots, R_n)) = \Theta(\lambda_{\Sigma_1}(R_1), \ldots, \lambda_{\Sigma_n}(R_n)). \quad \square$$

The above definition essentially requires Ψ to coincide with Θ on total relations (which are in one-one correspondence with the ordinary relations). In general, there may be many operators on paraconsistent relations that are weak generalisations of a given operator Θ on ordinary relations. The behavior of the weak generalisations of Θ on even just the consistent relations may in general vary. We require a stronger notion of operator generalisation under which, at least when restricted to consistent relations, the behavior of all the generalised operators is the same. Before we can develop such a notion, we need that of 'completions' of a paraconsistent relation.

We associate with a consistent relation R the set of all (ordinary relations corresponding to) total relations obtainable from R by throwing in the missing tuples. Let the map $\mathbf{comps}_\Sigma : \mathcal{C}(\Sigma) \to 2^{\mathcal{O}(\Sigma)}$ be given by

$$\mathbf{comps}_\Sigma(R) = \{Q \in \mathcal{O}(\Sigma) \mid R^+ \subseteq Q \subseteq \tau(\Sigma) - R^-\}.$$

The set $\mathbf{comps}_\Sigma(R)$ contains all ordinary relations that are 'completions' of the consistent relation R. Observe that $\mathbf{comps}_\Sigma$ is defined only for consistent relations and produces sets of ordinary relations. The following observation is immediate.

Proposition 1 *For any consistent relation R on scheme Σ, $\mathbf{comps}_\Sigma(R)$ is the singleton set $\{R^+\}$ iff R is total.*

Figure 1 gives a pictorial view of the different kinds of relations and the maps λ_Σ and $\mathbf{comps}_\Sigma$.

We now need to extend operators on ordinary relations to sets of ordinary relations. For any operator $\Theta : \mathcal{O}(\Sigma_1) \times \cdots \times \mathcal{O}(\Sigma_n) \to \mathcal{O}(\Sigma_{n+1})$ on ordinary relations, we let $\mathcal{S}(\Theta) : 2^{\mathcal{O}(\Sigma_1)} \times \cdots \times 2^{\mathcal{O}(\Sigma_n)} \to 2^{\mathcal{O}(\Sigma_{n+1})}$ be a map on sets of ordinary relations defined as follows. For any sets $M_1, \ldots, M_n$ of ordinary relations on schemes $\Sigma_1, \ldots, \Sigma_n$, respectively,

$$\mathcal{S}(\Theta)(M_1, \ldots, M_n) = \{\Theta(R_1, \ldots, R_n) \mid R_i \in M_i, \text{ for all } i, 1 \leq i \leq n\}.$$

In other words, $\mathcal{S}(\Theta)(M_1, \ldots, M_n)$ is the set of Θ-images of all tuples in the cartesian product $M_1 \times \cdots \times M_n$. We are now ready to lead up to a stronger notion of operator generalisation.

Definition 6 An operator Ψ on paraconsistent relations with signature $\langle \Sigma_1, \ldots, \Sigma_{n+1} \rangle$ is *consistency preserving* if for any consistent relations $R_1, \ldots, R_n$ on schemes $\Sigma_1, \ldots, \Sigma_n$, respectively, $\Psi(R_1, \ldots, R_n)$ is also consistent. $\square$

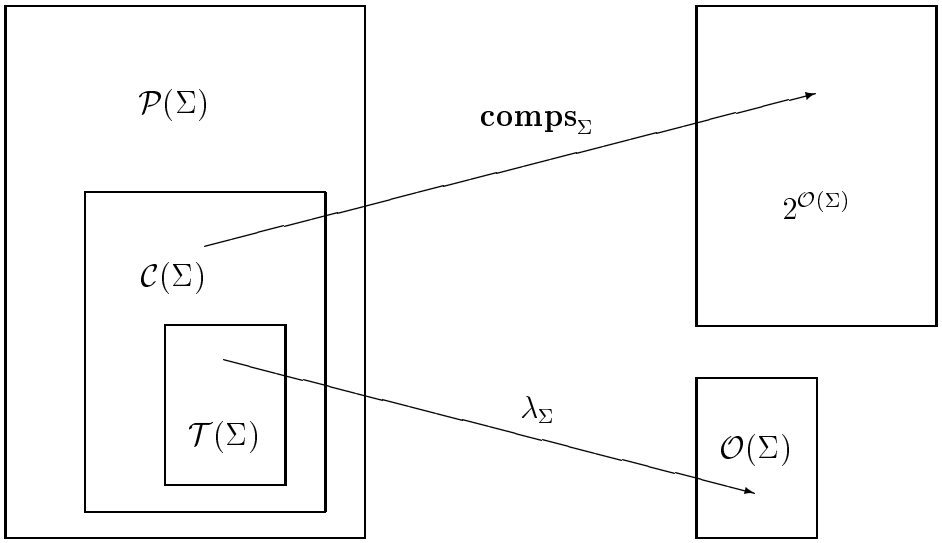

Figure 1: Classes of relations

Definition 7 A consistency preserving operator Ψ on paraconsistent relations with signature $\langle \Sigma_1, \ldots, \Sigma_{n+1} \rangle$ is a *strong generalisation* of an operator Θ on ordinary relations with the same signature, if for any consistent relations $R_1, \ldots, R_n$ on schemes $\Sigma_1, \ldots, \Sigma_n$, respectively, we have

$$\mathbf{comps}_{\Sigma_{n+1}}(\Psi(R_1, \ldots, R_n)) = \mathcal{S}(\Theta)(\mathbf{comps}_{\Sigma_1}(R_1), \ldots, \mathbf{comps}_{\Sigma_n}(R_n)). \quad \square$$

Given an operator Θ on ordinary relations, the behavior of a weak generalisation of Θ is 'controlled' only over the total relations. On the other hand, the behavior of a strong generalisation is 'controlled' over all consistent relations. This itself suggests that strong generalisation is a stronger notion than weak generalisation. The following proposition makes this precise.

Proposition 2 *If Ψ is a strong generalisation of Θ, then Ψ is also a weak generalisation of Θ.*

Proof Let $\langle \Sigma_1, \ldots, \Sigma_{n+1} \rangle$ be the signature of Ψ and Θ, and let $R_1, \ldots, R_n$ be any total relations on schemes $\Sigma_1, \ldots, \Sigma_n$, respectively. Since all total relations are consistent, and Ψ is a strong generalisation of Θ, we have that

$$\mathbf{comps}_{\Sigma_{n+1}}(\Psi(R_1, \ldots, R_n)) = \mathcal{S}(\Theta)(\mathbf{comps}_{\Sigma_1}(R_1), \ldots, \mathbf{comps}_{\Sigma_n}(R_n)).$$

Proposition 1 gives us that for each i, $1 \leq i \leq n$, $\mathbf{comps}_{\Sigma_i}(R_i)$ is the singleton set $\{R_i^+\}$, i.e. $\{\lambda_{\Sigma_i}(R_i)\}$. Therefore, $\mathcal{S}(\Theta)(\mathbf{comps}_{\Sigma_1}(R_1), \ldots, \mathbf{comps}_{\Sigma_n}(R_n))$ is just the singleton set

$\{\Theta(\lambda_{\Sigma_1}(R_1), \ldots, \lambda_{\Sigma_n}(R_n))\}$. Hence, $\Psi(R_1, \ldots, R_n)$ is total, and $\lambda_{\Sigma_{n+1}}(\Psi(R_1, \ldots, R_n)) = \Theta(\lambda_{\Sigma_1}(R_1), \ldots, \lambda_{\Sigma_n}(R_n))$, i.e. Ψ is a weak generalisation of Θ. $\square$

Though there may be many strong generalisations of an operator on ordinary relations, they all behave the same when restricted to consistent relations. In the next section, we propose strong generalisations for the usual operators on ordinary relations. The proposed generalised operators on paraconsistent relations correspond to the belief system intuition behind paraconsistent relations.

3 Algebraic Operators on Paraconsistent Relations

In this section, we present one strong generalisation each for the usual ordinary relation operators, such as union, join, projection. To reflect generalisation, a dot is placed over an ordinary relation operator to obtain the corresponding paraconsistent relation operator. For example, $\bowtie$ denotes the natural join among ordinary relations, and $\dot{\bowtie}$ denotes natural join on paraconsistent relations.

Set-Theoretic Operators

We first introduce two fundamental set-theoretic algebraic operators on paraconsistent relations:

Definition 8 Let R and S be paraconsistent relations on scheme Σ. Then,

(a) the *union* of R and S, denoted $R \dot{\cup} S$, is a paraconsistent relation on scheme Σ, given by

$$(R \dot{\cup} S)^+ = R^+ \cup S^+, \qquad (R \dot{\cup} S)^- = R^- \cap S^-;$$

(b) the *complement* of R, denoted $\dot{-} R$, is a paraconsistent relation on scheme Σ, given by

$$(\dot{-} R)^+ = R^-, \qquad (\dot{-} R)^- = R^+. \ \square$$

An intuitive appreciation of the union operator may be obtained by interpreting relations as properties of tuples. So, $R \dot{\cup} S$ is the "either-R-or-S" property. Now since R^+ and S^+ are the sets of tuples for which the properties R and S, respectively, are believed to hold, the set of tuples for which the property "either-R-or-S" is believed to hold is clearly $R^+ \cup S^+$. Moreover, since R^- and S^- are the sets of tuples for which properties R and S, respectively, are believed to *not* hold, the set of tuples for which the property "either-R-or-S" is believed to *not* hold is similarly $R^- \cap S^-$.

In an informal symbolic notation, $R^+ = \{t \mid R(t)\}$, and $R^- = \{t \mid \textbf{not } R(t)\}$. Now, $(R \dot{\cup} S)^+ = \{t \mid R(t) \textbf{ or } S(t)\}$, which is $R^+ \cup S^+$. Similarly, $(R \dot{\cup} S)^- = \{t \mid \textbf{not } (R(t) \textbf{ or } S(t))\} = \{t \mid \textbf{not } R(t) \textbf{ and not } S(t)\}$, which is $R^- \cap S^-$.

The definition of *complement* and of all the other operators on paraconsistent relations defined later can (and should) be understood in the same way.

Proposition 3 *The operators* $\dot{\cup}$ *and unary* $\dot{-}$ *on paraconsistent relations are strong generalisations of the usual operators* $\cup$ *and unary* $-$ *on ordinary relations.*

Proof Let R and S be consistent relations on scheme Σ. Then $\textbf{comps}_\Sigma(R \dot{\cup} S)$ is the set $\{Q \mid R^+ \cup S^+ \subseteq Q \subseteq \tau(\Sigma) - (R^- \cap S^-)\}$. This set is the same as the set $\{r \cup s \mid R^+ \subseteq r \subseteq \tau(\Sigma) - R^-, S^+ \subseteq s \subseteq \tau(\Sigma) - S^-\}$, which is $\mathcal{S}(\cup)(\textbf{comps}_\Sigma(R), \textbf{comps}_\Sigma(S))$. Such a result for unary $\dot{-}$ can also be shown similarly. $\square$

For sake of completeness, we define the following two related set-theoretic operators on paraconsistent relations:

Definition 9 Let R and S be paraconsistent relations on scheme Σ. Then,

(a) the *intersection* of R and S, denoted $R \dot{\cap} S$, is a paraconsistent relation on scheme Σ, given by

$$(R \dot{\cap} S)^+ = R^+ \cap S^+, \qquad (R \dot{\cap} S)^- = R^- \cup S^-;$$

(b) the *difference* of R and S, denoted $R \dot{-} S$, is a paraconsistent relation on scheme Σ, given by

$$(R \dot{-} S)^+ = R^+ \cap S^-, \qquad (R \dot{-} S)^- = R^- \cup S^+. \square$$

Again, to obtain an intuitive grasp of such definitions, let us consider the difference operator. We should interpret $R \dot{-} S$ as the "R-but-not-S" property. Now the tuples that definitely have this property are exactly those in $R^+ \cap S^-$. Thus, $(R \dot{-} S)^+ = R^+ \cap S^-$. Moreover, the tuples that definitely *do not* have this property are the ones in $R^- \cup S^+$. Hence, $(R \dot{-} S)^- = R^- \cup S^+$.

In our informal symbolic notation, $(R \dot{-} S)^+ = \{t \mid R(t) \textbf{ and not } S(t)\}$, which is $R^+ \cap S^-$. Similarly, $(R \dot{-} S)^- = \{t \mid \textbf{not } (R(t) \textbf{ and not } S(t))\} = \{t \mid \textbf{not } R(t) \textbf{ or } S(t)\}$, which is $R^- \cup S^+$.

Although we have given independent definitions of intersection and difference, these two operators can be derived from the fundamental operators union and complement as intuitively expected.

Proposition 4 *For any paraconsistent relations R and S on a common scheme, we have*

$$\begin{aligned} R \mathbin{\dot\cap} S &= \dot-(\dot-R \mathbin{\dot\cup} \dot-S), \text{ and} \\ R \mathbin{\dot-} S &= \dot-(\dot-R \mathbin{\dot\cup} S). \end{aligned}$$

Proof $(\dot-(\dot-R \mathbin{\dot\cup} \dot-S))^+ = (\dot-R \mathbin{\dot\cup} \dot-S)^- = (\dot-R)^- \cap (\dot-S)^- = R^+ \cap S^+ = (R \mathbin{\dot\cap} S)^+$. Similarly, $(\dot-(\dot-R \mathbin{\dot\cup} \dot-S))^- = (R \mathbin{\dot\cap} S)^-$. The second part of the result can be shown similarly. □

Table 1 gives some more algebraic laws involving the set-theoretic operators. The symbols Φ and $\mho$ in the table denote the *empty* and *universal* total relations, respectively, i.e. $\Phi = \langle \emptyset, \tau(\Sigma) \rangle$ and $\mho = \langle \tau(\Sigma), \emptyset \rangle$.

Table 1: Laws of Algebra of Paraconsistent Relations

1*a*.	$R \mathbin{\dot\cup} S = S \mathbin{\dot\cup} R$	commutative laws
b.	$R \mathbin{\dot\cap} S = S \mathbin{\dot\cap} R$	
2*a*.	$(R \mathbin{\dot\cup} S) \mathbin{\dot\cup} T = R \mathbin{\dot\cup} (S \mathbin{\dot\cup} T)$	associative laws
b.	$(R \mathbin{\dot\cap} S) \mathbin{\dot\cap} T = R \mathbin{\dot\cap} (S \mathbin{\dot\cap} T)$	
3*a*.	$R \mathbin{\dot\cup} (S \mathbin{\dot\cap} T) = (R \mathbin{\dot\cup} S) \mathbin{\dot\cap} (R \mathbin{\dot\cup} T)$	distributive laws
b.	$R \mathbin{\dot\cap} (S \mathbin{\dot\cup} T) = (R \mathbin{\dot\cap} S) \mathbin{\dot\cup} (R \mathbin{\dot\cap} T)$	
4*a*.	$R \mathbin{\dot\cup} R = R$	idempotent laws
b.	$R \mathbin{\dot\cap} R = R$	
5*a*.	$R \mathbin{\dot\cup} \Phi = R$	identity laws
b.	$R \mathbin{\dot\cup} \mho = \mho$	
c.	$R \mathbin{\dot\cap} \Phi = \Phi$	
d.	$R \mathbin{\dot\cap} \mho = R$	
6.	$\dot-(\dot- R) = R$	double complementation
7*a*.	$R \mathbin{\dot\cup} \dot- R = \mho$	
b.	$R \mathbin{\dot\cap} \dot- R = \Phi$	
8*a*.	$\dot- \mho = \Phi$	
b.	$\dot- \Phi = \mho$	
9*a*.	$\dot-(R \mathbin{\dot\cup} S) = \dot- R \mathbin{\dot\cap} \dot- S$	De Morgan laws
b.	$\dot-(R \mathbin{\dot\cap} S) = \dot- R \mathbin{\dot\cup} \dot- S$	

Relation-Theoretic Operators

If Σ and Δ are relation schemes such that $\Sigma \subseteq \Delta$, then for any tuple $t \in \tau(\Sigma)$, we let t^Δ denote the set $\{t' \in \tau(\Delta) \mid t'(A) = t(A), \text{ for all } A \in \Sigma\}$ of all extensions of t. We

extend this notion for any $T \subseteq \tau(\Sigma)$ by defining $T^\Delta = \cup_{t \in T}\, t^\Delta$. We now define some relation-theoretic algebraic operators on paraconsistent relations.

Definition 10 Let R and S be paraconsistent relations on schemes Σ and Δ, respectively. Then, the *natural join* (or just *join*) of R and S, denoted $R \bowtie S$, is a paraconsistent relation on scheme $\Sigma \cup \Delta$, given by

$$(R \bowtie S)^+ = R^+ \bowtie S^+, \qquad (R \bowtie S)^- = (R^-)^{\Sigma \cup \Delta} \cup (S^-)^{\Sigma \cup \Delta},$$

where $\bowtie$ is the usual natural join among ordinary relations. □

It is instructive to observe that $(R \bowtie S)^-$ contains all extensions of tuples in R^- and S^-, because at least one of R and S is believed false for these extended tuples. The following proposition is straightforward.

Proposition 5 $\bowtie$ *is a strong generalisation of* $\bowtie$.

Proof Let R and S be consistent relations on schemes Σ and Δ, respectively. Then $\mathbf{comps}_{\Sigma \cup \Delta}(R \bowtie S)$ is the set

$$\{Q \in \mathcal{O}(\Sigma \cup \Delta) \mid R^+ \bowtie S^+ \subseteq Q \subseteq \tau(\Sigma \cup \Delta) - ((R^-)^{\Sigma \cup \Delta} \cup (S^-)^{\Sigma \cup \Delta})\},$$

and $\mathcal{S}(\bowtie)(\mathbf{comps}_\Sigma(R), \mathbf{comps}_\Delta(S)) = \{r \bowtie s \mid r \in \mathbf{comps}_\Sigma(R), s \in \mathbf{comps}_\Delta(S)\}$.

Let $Q \in \mathbf{comps}_{\Sigma \cup \Delta}(R \bowtie S)$. Then $\pi_\Sigma(Q) \in \mathbf{comps}_\Sigma(R)$, where π_Σ is the usual projection over Σ of ordinary relations. Similarly, $\pi_\Delta(Q) \in \mathbf{comps}_\Delta(S)$. Therefore, $Q \in \mathcal{S}(\bowtie)(\mathbf{comps}_\Sigma(R), \mathbf{comps}_\Delta(S))$.

Now let $Q \in \mathcal{S}(\bowtie)(\mathbf{comps}_\Sigma(R), \mathbf{comps}_\Delta(S))$. Then $R^+ \bowtie S^+ \in Q$ and $Q \cap (R^-)^{\Sigma \cup \Delta} = Q \cap (S^-)^{\Sigma \cup \Delta} = \emptyset$, because R and S are consistent. Therefore, $Q \in \mathbf{comps}_{\Sigma \cup \Delta}(R \bowtie S)$. □

Definition 11 Let R be a paraconsistent relation on scheme Σ, and Δ be any scheme. Then, the *projection* of R onto Δ, denoted $\dot{\pi}_\Delta(R)$, is a paraconsistent relation on Δ, given by

$$\dot{\pi}_\Delta(R)^+ = \pi_\Delta((R^+)^{\Sigma \cup \Delta}), \qquad \dot{\pi}_\Delta(R)^- = \{t \in \tau(\Delta) \mid t^{\Sigma \cup \Delta} \subseteq (R^-)^{\Sigma \cup \Delta}\},$$

where π_Δ is the usual projection over Δ of ordinary relations. □

It should be noted that, contrary to usual practice, the above definition of projection is not just for subschemes. However, if $\Delta \subseteq \Sigma$, then it coincides with the intuitive projection operation. In this case, $\dot{\pi}_\Delta(R)^-$ consists of those tuples in $\tau(\Delta)$, all of whose extensions are in R^-. We now define our last relation-theoretic operation.

Definition 12 Let R be a paraconsistent relation on scheme Σ, and let F be any logic formula involving attribute names in Σ, constant symbols (denoting values in the attribute domains), equality symbol $=$, negation symbol $\neg$, and connectives $\vee$ and $\wedge$. Then, the *selection* of R by F, denoted $\dot{\sigma}_F(R)$, is a paraconsistent relation on scheme Σ, given by

$$\dot{\sigma}_F(R)^+ = \sigma_F(R^+), \qquad \dot{\sigma}_F(R)^- = R^- \cup \sigma_{\neg F}(\tau(\Sigma)),$$

where σ_F is the usual selection of tuples satisfying F from ordinary relations. □

Proposition 6 *The operators $\dot{\pi}$ and $\dot{\sigma}$ are strong generalisations of π and σ, respectively.*

Proof Similar to that of Proposition 5. □

Example 1 Strictly speaking, relation schemes are sets of attribute names, but in this example we treat them as ordered sequences of attribute names, so tuples can be viewed as the usual lists of values. Let $\{a, b, c\}$ be a common domain for all attribute names, and let R and S be the following paraconsistent relations on schemes $\langle X, Y\rangle$ and $\langle Y, Z\rangle$, respectively:

$$\begin{array}{rclcrcl} R^+ & = & \{(b,b),(b,c)\}, & & R^- & = & \{(a,a),(a,b),(a,c)\}, \\ S^+ & = & \{(a,c),(c,a)\}, & & S^- & = & \{(c,b)\}. \end{array}$$

Then, $R \bowtie S$ is the following paraconsistent relation on scheme $\langle X, Y, Z\rangle$:

$$\begin{array}{rcl} (R \bowtie S)^+ & = & \{(b,c,a)\}, \\ (R \bowtie S)^- & = & \{(a,a,a),(a,a,b),(a,a,c),(a,b,a),(a,b,b),(a,b,c), \\ & & \quad (a,c,a),(a,c,b),(a,c,c),(b,c,b),(c,c,b)\}. \end{array}$$

Observe how $(R \bowtie S)^-$ blows up to contain extensions of all tuples in R^- and S^-. Now, $\dot{\pi}_{\langle X,Z\rangle}(R \bowtie S)$ becomes the following paraconsistent relation on scheme $\langle X, Z\rangle$:

$$\dot{\pi}_{\langle X,Z\rangle}(R \bowtie S)^+ = \{(b,a)\}, \qquad \dot{\pi}_{\langle X,Z\rangle}(R \bowtie S)^- = \{(a,a),(a,b),(a,c)\}.$$

The tuples in the negative component of the projected paraconsistent relation are such that all their extensions were present in the negative component of the original paraconsistent relation. Finally, $\dot{\sigma}_{\neg X=Z}(\dot{\pi}_{\langle X,Z\rangle}(R \bowtie S))$ becomes the following paraconsistent relation on the same scheme:

$$\begin{array}{rcl} \dot{\sigma}_{\neg X=Z}(\dot{\pi}_{\langle X,Z\rangle}(R \bowtie S))^+ & = & \{(b,a)\}, \\ \dot{\sigma}_{\neg X=Z}(\dot{\pi}_{\langle X,Z\rangle}(R \bowtie S))^- & = & \{(a,a),(a,b),(a,c),(b,b),(c,c)\}. \end{array}$$

All tuples that do not satisfy the selection condition always make it to the negative component of the selected paraconsistent relation. □

4 An Application

In this section we give an application of the algebra on paraconsistent relations. We briefly present a bottom-up method for constructing the weak well-founded model of any general deductive database. For a somewhat detailed exposition on general deductive databases the reader is referred to [17], and on the weak well-founded model to [9].

General Deductive Databases

Deductive databases are an extension of relational databases, in that in addition to manipulating explicitly represented facts, deductive databases provide ways to deduce facts from other facts in the database. We now give a very brief introduction to deductive databases. More details can be found in [8, 10, 17],

Let L be a given underlying language with a finite set of constant, variable, and predicate symbols, but no function symbols. A *term* is either a variable or a constant. An *atom* is of the form $p(t_1, \ldots, t_n)$, where p is a predicate symbol and the $t_i's$ are terms. A *literal* is either a *positive literal* A or a *negative literal* $\neg A$, where A is an atom. For any literal l we let l' denote its complementary literal, i.e. if l is positive then $l' = \neg l$, otherwise $l = \neg l'$.

Definition 13 A *deductive database* is a finite set of clauses of the form

$$a \leftarrow b_1, b_2, \ldots, b_m$$

where $m \geq 0$ and a and each b_i is an atom. $\Box$

A term, atom, literal, or clause is called *ground* if it contains no variables. The *Herbrand Universe* of the underlying language is the set of all ground terms; the *Herbrand Base* of the language is the set of all ground atoms; A *Herbrand Interpretation* of the language is any subset of the Herbrand Base. A *ground instance* of a term, atom, literal, or clause Q is the term, atom, literal, or clause, respectively, obtained by replacing each variable in Q by a constant. For any deductive database P, we let $P^\star$ denote the set of all ground instances of clauses in P. Note that since the underlying language has no function symbols, unlike logic programs, $P^\star$ is always finite.

One way to give the semantics of a definite deductive database is the least fixpoint of the following immediate consequence function T_P on Herbrand interpretations:

Definition 14 Let I be a Herbrand interpretation for a deductive database P. Then $T_P(I)$ is a Herbrand interpretation, given by

$$T_P(I) \;=\; \{a \mid \text{for some clause } a \leftarrow b_1, \ldots, b_m \text{ in } P^\star, \{b_1, \ldots, b_m\} \subseteq I\}. \; \Box$$

It is well-known that T_P always possesses a least fixpoint with respect to the partial order of set inclusion. The least fixpoint can be shown to be the *minimal model* for P. This model is also known to be $T_P \uparrow \omega$, where the ordinal powers of T_P are defined as follows:

Definition 15 For any ordinal α,

$$T_P \uparrow \alpha = \begin{cases} \emptyset & \text{if } \alpha = 0, \\ T_P(T_P \uparrow (\alpha - 1)) & \text{if } \alpha \text{ is a successor ordinal,} \\ \cup_{\beta<\alpha}(T_P \uparrow \beta), & \text{if } \alpha \text{ is a limit ordinal. } \Box \end{cases}$$

The expressive power of the clauses of a deductive databases can be increased by allowing negated atoms in their bodies. This results in a more general class of deductive databases defined below.

Definition 16 A *general deductive database* is a finite set of clauses of the form

$$a \leftarrow l_1, l_2, \ldots, l_m$$

where a is an atom, $m \geq 0$ and each l_i is a literal. $\Box$

Weak Well-founded Model of General Deductive Databases

One of the semantics given for general deductive databases is described next. It was originally presented in [9] for logic programs.

Definition 17 A *partial interpretation* is a pair $I = \langle I^+, I^- \rangle$, where I^+ and I^- are any subsets of the Herbrand Base. $\Box$

A partial interpretation I is *consistent* if $I^+ \cap I^- = \emptyset$. For any partial interpretations I and J, we let $I \cap J$ be the partial interpretation $\langle I^+ \cap J^+, I^- \cap J^- \rangle$, and $I \cup J$ be the partial interpretation $\langle I^+ \cup J^+, I^- \cup J^- \rangle$. We also say $I \subseteq J$ whenever $I^+ \subseteq J^+$ and $I^- \subseteq J^-$.

Definition 18 Let S be a set partially ordered by $\subseteq$. A map $T : S \rightarrow S$ is *monotonic* if, for any $X, Y \in S$, $X \subseteq Y$ implies $T(X) \subseteq T(Y)$. $\Box$

For any general deductive database P, recall that $P^\star$ is the set of all ground instances of clauses in P. The weak well-founded model of P is the least fixpoint of the immediate consequence function T_P^F on consistent partial interpretations defined as follows:

Definition 19 Let I be a partial interpretation. Then $T_P^F(I)$ is a partial interpretation, given by

$$\begin{aligned} T_P^F(I)^+ &= \{a \mid \text{for some clause } a \leftarrow l_1, \ldots, l_m \text{ in } P^\star, \text{ for each } i, 1 \leq i \leq m, \\ &\qquad \text{if } l_i \text{ is positive then } l_i \in I^+, \text{ and} \\ &\qquad \text{if } l_i \text{ is negative then } l_i' \in I^-\}, \\ T_P^F(I)^- &= \{a \mid \text{for every clause } a \leftarrow l_1, \ldots, l_m \text{ in } P^\star, \text{ there is some } i, 1 \leq i \leq m, \\ &\qquad \text{such that if } l_i \text{ is positive then } l_i \in I^-, \text{ and} \\ &\qquad\qquad \text{if } l_i \text{ is negative then } l_i' \in I^+\}. \ \square \end{aligned}$$

It can be shown that T_P^F preserves consistency and always possesses a least fixpoint. This least fixpoint is called the *weak well-founded model* for P. The model can also be shown to be $T_P^F \uparrow \omega$, where the ordinal powers of T_P^F are defined as follows:

Definition 20 For any ordinal α,

$$T_P^F \uparrow \alpha = \begin{cases} \langle \emptyset, \emptyset \rangle & \text{if } \alpha = 0, \\ T_P^F(T_P^F \uparrow (\alpha - 1)) & \text{if } \alpha \text{ is a successor ordinal,} \\ \langle \cup_{\beta<\alpha}(T_P^F \uparrow \beta)^+, \ \cup_{\beta<\alpha}(T_P^F \uparrow \beta)^- \rangle & \text{if } \alpha \text{ is a limit ordinal. } \square \end{cases}$$

In [1], the *upward closure ordinal* of the immediate consequence function is defined as the least ordinal α such that $T_P^F \uparrow \alpha$ is a fixpoint of T_P^F. The following observation for deductive databases is relevant:

Proposition 7 *For any general deductive database P, the upward closure ordinal of T_P^F is finite, i.e. there is a number $n \geq 0$, such that $T_P^F \uparrow n = T_P^F \uparrow \omega$.* $\square$

Thus a mechanism that "computes" the ordinal powers of T_P^F can be employed to construct the weak well-founded model of P.

Construction of the Weak Well-founded Model

We now describe a method for constructing the weak well-founded model for a given general deductive database P. In this model, paraconsistent relations are the semantic objects associated with the predicate symbols occurring in P.

The method involves two steps. The first step is to convert P into a set of paraconsistent relation definitions for the predicate symbols occurring in P. These definitions are of the form

$$\mathtt{p} = D_{\mathtt{p}},$$

where $\mathtt{p}$ is a predicate symbol of P, and $D_{\mathtt{p}}$ is an algebraic expression involving predicate symbols of P and paraconsistent relation operators. The second step is to iteratively evaluate the expressions in these definitions to incrementally construct the paraconsistent relations associated with the predicate symbols.

A scheme Σ is a *Herbrand scheme* if $dom(A)$ is the Herbrand Universe, for all $A \in \Sigma$. The schemes of the paraconsistent relations that we associate with the predicate symbols are set internally. Let $\Gamma = \langle \nu_1, \nu_2, \ldots \rangle$ be an infinite sequence of some distinct attribute names. For any $n \geq 1$, let Γ_n be the Herbrand scheme $\{\nu_1, \ldots, \nu_n\}$. We use the following scheme renaming operators.

Definition 21 Let $\Sigma = \{A_1, \ldots, A_n\}$ be any Herbrand scheme. Then,

(a) for any paraconsistent relation R on scheme Γ_n, $R(A_1, \ldots, A_n)$ is the paraconsistent relation

$$\langle \quad \{t \in \tau(\Sigma) \mid \text{for some } t' \in R^+, t(A_i) = t'(\nu_i), \text{ for all } i, 1 \leq i \leq n\}, \\ \{t \in \tau(\Sigma) \mid \text{for some } t' \in R^-, t(A_i) = t'(\nu_i), \text{ for all } i, 1 \leq i \leq n\} \quad \rangle$$

on scheme Σ, and

(b) for any paraconsistent relation R on scheme Σ, $R[A_1, \ldots, A_n]$ is the paraconsistent relation

$$\langle \quad \{t \in \tau(\Gamma_n) \mid \text{for some } t' \in R^+, t(\nu_i) = t'(A_i), \text{ for all } i, 1 \leq i \leq n\}, \\ \{t \in \tau(\Gamma_n) \mid \text{for some } t' \in R^-, t(\nu_i) = t'(A_i), \text{ for all } i, 1 \leq i \leq n\} \quad \rangle$$

on scheme Γ_n. □

Before describing our method to convert the given database P into a set of definitions for the predicate symbols in P, let us look at an example. Suppose the following are the only clauses with the predicate symbol $\mathtt{p}$ in their heads:

$$\mathtt{p(X)} \leftarrow \mathtt{r(X,Y)},\ \neg\mathtt{p(Y)}$$
$$\mathtt{p(Y)} \leftarrow \mathtt{s(Y,a)}$$

From these clauses the algebraic definition constructed for the symbol $\mathtt{p}$ is the following:

$$\mathtt{p} \quad = \quad (\dot{\pi}_{\mathtt{\{X\}}}(\mathtt{r(X,Y)} \bowtie \dot{-}\mathtt{p(Y)}))[\mathtt{X}] \ \ \dot{\cup} \ \ (\mathtt{s(Y,Z)}))))[\mathtt{Y}]$$

Such a conversion exploits the close connection between attribute names in relation schemes and variables in clauses, as pointed out in [24]. The expression thus constructed can be used to arrive at a better approximation of the paraconsistent relation $\mathtt{p}$ from some approximations of $\mathtt{p}$, $\mathtt{r}$ and $\mathtt{s}$.

The algebraic expression for the predicate symbol $\mathtt{p}$ is a union ($\dot{\cup}$) of the expressions obtained from each clause containing the symbol $\mathtt{p}$ in its head. It therefore suffices to give an algorithm for converting one clause into an expression.

Algorithm CONVERT

Input: A general deductive database clause $l_0 \leftarrow l_1, \ldots, l_m$.
Let l_0 be of the form $p_0(A_{01}, \ldots, A_{0k_0})$, and each l_i, $1 \leq i \leq m$, be either of the form $p_i(A_{i1}, \ldots, A_{ik_i})$, or of the form $\neg p_i(A_{i1}, \ldots, A_{ik_i})$. For any i, $0 \leq i \leq m$, let V_i be the set of all variables occurring in l_i.

Output: An algebraic expression involving paraconsistent relations.

Method: The expression is constructed by the following steps:

1. For each argument A_{ij} of literal l_i, construct argument B_{ij} and condition C_{ij} as follows:

   (a) If A_{ij} is a constant a, then B_{ij} is any brand new variable and C_{ij} is $B_{ij} = a$.

   (b) If A_{ij} is a variable, such that for each k, $1 \leq k < j$, $A_{ik} \neq A_{ij}$, then B_{ij} is A_{ij} and C_{ij} is *true*.

   (c) If A_{ij} is a variable, such that for some k, $1 \leq k < j$, $A_{ik} = A_{ij}$, then B_{ij} is a brand new variable and C_{ij} is $A_{ij} = B_{ij}$.

2. Let $\hat{l}_i$ be the atom $p_i(B_{i1}, \ldots, B_{ik_i})$, and F_i be the conjunction $C_{i1} \wedge \cdots \wedge C_{ik_i}$. If l_i is a positive literal, then let Q_i be the expression $\dot{\pi}_{V_i}(\dot{\sigma}_{F_i}(\hat{l}_i))$. Otherwise, let Q_i be the expression $\dot{-}\dot{\pi}_{V_i}(\dot{\sigma}_{F_i}(\hat{l}_i))$.

   As a syntactic optimisation, if all conjuncts of F_i are *true* (i.e. all arguments of l_i are distinct variables), then both $\dot{\sigma}_{F_i}$ and $\dot{\pi}_{V_i}$ are reduced to identity operations, and are hence dropped from the expression. For example, if $l_i = \neg$`p(X,Y)`, then $Q_i = \dot{-}$`p(X,Y)`.

3. Let E be the natural join ($\bowtie$) of the Q_i's thus obtained, $1 \leq i \leq m$. The output expression is $(\dot{\sigma}_{F_0}(\dot{\pi}_V(E)))[B_{01}, \ldots, B_{0k_0}]$, where V is the set of variables occurring in $\hat{l}_0$.

   As in step 2, if all conjuncts in F_0 are *true*, then $\dot{\sigma}_{F_0}$ is dropped from the output expression. However, $\dot{\pi}_V$ is never dropped, as the clause body may contain variables not in V. □

From the algebraic expressions obtained by Algorithm **CONVERT** for clauses in the given general deductive database, we construct a system of equations defining paraconsistent relations as follows.

Definition 22 For any general deductive database P, $EQN(P)$ is a set of all equations of the form $\mathtt{p} = D_{\mathtt{p}}$, where `p` is a predicate symbol of P, and $D_{\mathtt{p}}$ is the union ($\dot{\cup}$) of all expressions obtained by Algorithm **CONVERT** for clauses in P with symbol `p` in their head. The algebraic expression $D_{\mathtt{p}}$ is also called a *definition* of `p`. □

It is evident that a predicate symbol may have many definitions. It can be shown that the above method for converting a general deductive database P into definitions for its predicate symbols terminates, and that the definitions produced mimic the T_P^F map defined in [9].

The second and final step in our model construction process is to incrementally construct the paraconsistent relations defined by the given database. For any general deductive database P, we let P_E and P_I denote its extensional and intensional portions, respectively. P_E is essentially the set of clauses of P with empty bodies, and P_I is the set of all other clauses of P. Without loss of generality, we assume that no predicate symbol occurs both in P_E as well as in P_I. Let us recall that $P_E^\star$ is the set of all ground instances of clauses in P_E.

The overall construction algorithm is rather straightforward. It treats the predicate symbols in a given database as imperative "variable names" that may contain a paraconsistent relation as value. Thus, any variable $\mathtt{p}$ has two set-valued fields, namely $\mathtt{p}^+$ and $\mathtt{p}^-$.

Algorithm CONSTRUCT

Input: A general deductive database P.

Output: Paraconsistent relation values for the predicate symbols of P.

Method: The values are computed by the following steps:

1. *(Initialisation)*
   (a) Compute $EQN(P_I)$ using Algorithm **CONVERT** for each clause in P_I.
   (b) For each predicate symbol $\mathtt{p}$ in P_E, set

$$\begin{aligned} \mathtt{p}^+ &= \{\langle a_1, \ldots, a_k\rangle \mid \mathtt{p}(a_1, \ldots, a_k) \leftarrow \ \in P_E^\star\}, \text{ and} \\ \mathtt{p}^- &= \{\langle b_1, \ldots, b_k\rangle \mid k \text{ is the arity of } \mathtt{p}, \text{ and } \mathtt{p}(b_1, \ldots, b_k) \leftarrow \ \notin P_E^\star\}. \end{aligned}$$

   (c) For each predicate symbol $\mathtt{p}$ in P_I, set $\mathtt{p}^+ = \emptyset$, and $\mathtt{p}^- = \emptyset$.
2. For each equation of the form $\mathtt{p} = D_\mathtt{p}$ in $EQN(P_I)$, compute the expression $D_\mathtt{p}$ and set $\mathtt{p}$ to the resulting paraconsistent relation.
3. If step 2 involved a change in the value of some $\mathtt{p}$, goto 2.
4. Output the final values of all predicate symbols in P_E and P_I. □

Again, we omit the proof of termination of Algorithm **CONSTRUCT** and that it constructs the weak well-founded model of the given database.

5 Conclusions and Future Work

We have presented a generalisation of the relational data model, which is capable of manipulating incomplete or even inconsistent information. Paraconsistent relations, based on Belnap's 4-valued logic [2], form the mathematical structures underlying the model. A paraconsistent relation essentially contains two kinds of tuples: ones for which an underlying predicate is believed to be true, and ones for which that predicate is believed to be false. These structures are strictly more general than ordinary relations, in that for any ordinary relation there is a paraconsistent relation with the same information content, but not *vice versa*. Since paraconsistent relations are capable of containing contradictory information, these structures model belief systems more naturally than knowledge systems.

We developed two notions of generalising operators on ordinary relations for paraconsistent relations. Of these, the stronger notion guarantees that any generalised operator is "well-behaved" for paraconsistent relation operands that contain consistent information.

For some well-known operators on ordinary relations, such as union, join, projection, we introduced generalised operators on paraconsistent relations. These generalised operators maintain the belief system intuition behind paraconsistent relations, and are shown to be "well-behaved" in the sense mentioned above.

Our data model can be used to represent relational information that may be incomplete or inconsistent. As usual, the algebraic operators can be used to construct queries to any database systems for retrieving paraconsistent information. As another application of paraconsistent relations and the algebra on them, we presented a method for constructing the weak well-founded model for general deductive databases [9]. This method requires translating the clauses of the database into expressions involving the generalised operators introduced earlier. A minor modification to this method can tailor it for constructing the well-founded model for such databases [25].

Recently there has been some interest in studying extended logic programs in which the head of clauses can have one or more literals [19]. This leads to two notions of negation: *implicit* negation (corresponding to negative literals in the body) and *explicit* negation (corresponding to negative literals in the head). One possible direction for further research is to extend the paraconsistent relational model presented in this paper to include disjunctive tuples as in [16], thereby providing a framework under which the semantics of extended logic programs could be constructed in a bottom-up manner. Allowing explicit negation in a disjunctive deductive database/logic program usually cuts down on the number of minimal models, sometimes quite drastically, and as a consequence increases the efficiency of query processing. This will be one of the main motivations in exploring the possibility of paraconsistent relations with disjunctive tuples.

References

[1] R. Bagai, M. Bezem, and M. H. van Emden. On downward closure ordinals of logic programs. *Fundamenta Informaticae*, XIII(1):67–83, 1990.

[2] N. D. Belnap. A useful four-valued logic. In G. Eppstein and J. M. Dunn, editors, *Modern Uses of Many-valued Logic*, pages 8–37. Reidel, Dordrecht, 1977.

[3] J. Biskup. A foundation of Codd's relational maybe-operations. *ACM Transactions on Database Systems*, 8(4):608–636, December 1983.

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

[5] E. F. Codd. A relational model for large shared data banks. *Communications of the ACM*, 13(6):377–387, June 1970.

[6] E.F. Codd. Missing information (applicable and inapplicable) in relational databases. *SIGMOD Record*, 15(4):53–78, December 1986.

[7] N. C. A. da Costa. On the theory of inconsistent formal systems. *Notre Dame Journal of Formal Logic*, 15:621–630, 1977.

[8] S. K. Das. *Deductive databases and logic programming*. Addison-Wesley, New York, 1992.

[9] M. Fitting. A Kripke-Kleene semantics for logic programs. *Journal of Logic Programming*, 4:295–312, 1985.

[10] H. Gallaire, J. Minker, and J.M. Nicolas. Logic and databases: A deductive approach. *ACM Computing Surveys*, 16(2):151–184, June 1984.

[11] E. Gelenbe and G. Hebrail. A probability model of uncertainity in databases. In *Proceedings of the International Conference on Data Engineering*. IEEE Computer Society Press, 1986.

[12] J. Grant. Partial values in a tabular database model. *Information Processing Letters*, 9(2):97–99, August 1979.

[13] T. Imieliński and W. Lipski. Incomplete information in relational databases. *Journal of the ACM*, 31(4):761–791, October 1984.

[14] W. Lipski. On semantic issues connected with incomplete information databases. *ACM Transactions on Database Systems*, 4:262–296, 1979.

[15] K.-C. Liu and R. Sunderraman. On representing indefinite and maybe information in relational databases. In *Proceedings of the Fourth International Conference on Data Engineering, Los Angeles, California, February 1988*, pages 250–257, 1988.

[16] K.-C. Liu and R. Sunderraman. Indefinite and maybe information in relational databases. *ACM Transactions on Database Systems*, 15(1):1–39, 1990.

[17] J. W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, second edition, 1987.

[18] D. Maier. *The Theory of Relational Databases.* Computer Science Press, 1983.

[19] J. Minker and C. Ruiz. On extended disjunctive logic programs. In J. Komorowski and Z.W. Ras, editors, *Proceedings of the Seventh International Symposium on Methodologies for Intelligent Systems*, pages 1–18. Lecture Notes in AI, Springer-Verlag, New York, June 1993.

[20] K.V.S.V.N. Raju and A.K. Majumdar. Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems. *ACM Transactions on Database Systems*, 13(2):129–166, 1988.

[21] R. Reiter. Towards a logical reconstruction of relational database theory. In M. Brodie, J. Mylopoulos, and J.W. Schmidt, editors, *On Conceptual Modeling*, pages 191–238. Springer-Verlag, Berlin and New York, 1984.

[22] V. S. Subrahmanian. Paraconsistent disjunctive deductive databases. *Theoretical Computer Science*, 93:115–141, 1992.

[23] R. Sunderraman. Deductive databases with conditional facts. In M. Worboys and A. F. Grundy, editors, *Advances in Databases*, pages 162–175. Lecture Notes in Computer Science, 696, Springer-Verlag, 1993. (Proceedings of the 11th British National Conference on Databases).

[24] J. D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume 1. Computer Science Press, 1988.

[25] A. van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):621–650, 1991.