Paraconsistent Relational Model: A Quasi-Classic Logic Approach

Badrinath Jayakumar and **Rajshekar Sunderraman**
Department of Computer Science, Georgia State University,
Georgia, USA
{bjayakumar2, raj}@cs.gsu.edu

Abstract

The well-founded model for any general deductive database computed using the paraconsistent relational model, based on four-valued logic, does not support inference rules such as disjunctive syllogism. In order to support disjunctive syllogism, we utilize the generalization of the relational model to quasi-classic (QC) logic, whose inference power is much closer to classical logic. As the paraconsistent relational model is capable of representing incomplete and inconsistent data, we propose an algorithm to find QC model for inconsistent positive extended disjunctive deductive databases. To accomplish this, in addition to using existing generalized algebraic operators, we introduce two new operators $FOCUS_C$ and $FOCUS_D$.

1 Introduction

Most database systems that are currently in use are based on the relational model. This model defines the falsity of its attributes based on the absence of information (closed world assumption). However, the model is not suitable for all applications. For example, if a database does not contain a person's allergy to a medicine, when a doctor queries the person's allergy to the medicine, the database will then return false. To represent negative information in databases, we use paraconsistent databases (Bagai and Sunderraman 1995). Bagai and Sunderraman developed a framework to represent negative facts in relational databases, which is based on four-valued logic. The four-valued relation represents both positive and negative information, and negative facts are derived from open world assumption. The authors (Bagai and Sunderraman 1995) also developed an application for the paraconsistent relational model that finds the weak well-founded semantics of general deductive databases. For general deductive databases and disjunctive deductive databases, various paraconsistent semantics have been proposed (Bagai and Sunderraman 1996b; Subrahmanian 1990; Sunderraman 1997), all based upon Belnap's four-valued model (Belnap Jr 1977). Even for logic programs, especially disjunctive logic programs, various paraconsistent semantics are proposed (Alcântara, Damásio, and Pereira 2004; Arieli 2002; Damásio and Pereira 1998; Alcântara, Damásio, and Pereira 2002). All of which still come under Belnap's four-valued model.

However, such multi-valued logic does not support disjunctive syllogism (Hunter 1998). For example, suppose a knowledge base contains {passed $\vee$ failed} about a student. When new information about the student comes to the knowledge base {$\neg$failed}, the four-valued logic gives two paraconsistent minimal models (Sakama and Inoue 1995) {{passed, $\neg$failed}, {failed, $\neg$failed}}. Hence, $\neg$failed is the only logical consequence of the two models. Whether the student has passed or not cannot be inferred with four-valued logic.

It is very clear from the above example that this four-valued logic does not behave as expected in such situations. In order to get accurate models, it is required to use a stronger form of four-valued logic for improving the ability of reasoning. Therefore, we use QC logic (Hunter 2000). In this paper, the algorithm to find the model for positive extended disjunctive deductive databases is proposed by using QC logic with the paraconsistent relational model. To avoid getting into an infinite iteration, it is assumed that the program does not have recursions, and constraints are also not used. In the model construction, for any disjunctive clause, it is necessary to impose a link between each relation in union and its complementary of the relation in order to enable non-trivial classical conclusions.

The approach presented in this paper differs from the QC model (Zhang, Lin, and Ren 2009) in its algebraic nature. In this paper, we construct an algebraic equation for every clause in which expressions of the equation containing cartesian products operates with a set of tuples instead of one tuple at a time. In other words, the expression that represents disjunctive head operates on one tuple at a time. In addition to that, algrebraic expressions can be optimized by using various laws of equality.

The rest of the paper is organized as follows. In section 2, we introduce the preliminaries to understand the paper; in section 3, we define a paraconsistent disjunctive relational model; in section 4, we provide the key idea to understand the paper; in section 5, we explain the algorithm with an example; in section 6, we state the conclusion and future work for this paper.

2 Preliminaries

Before we explain the details of actual contributions of this paper, we must briefly review positive extended disjunctive deductive databases and the paraconsistent relational model (Bagai and Sunderraman 1996a; Zhang, Lin, and Ren 2009; Bagai and Sunderraman 1995) .

Syntax. Given a first order language $\mathcal{L}$, a disjunctive deductive database P (Minker and Seipel 2002) consists of logical inference rules of the form

$$r \text{ (rule) } = l_0 \vee \cdots \vee l_n \leftarrow l_{n+1} \ldots l_m$$

$l_0 \ldots l_n$ is called head of the rule and $l_{n+1} \ldots l_m$ is called body of the rule. A rule is called fact if the rule has no body. A rule is called denial rule if the rule has only body and no head. A rule is called definite clause or horn clause, if the rule has only one literal in the head and has some literal in the body. A rule is called positive disjunctive rule if the rule has both body and head. Concretely, the rule r is called positive extended disjunctive rule, if $l_0, \ldots, l_n, l_{n+1}, \ldots, l_m$ are either positive or negative ($\neg$) literals.

For the given syntax of a positive extended disjunctive deductive database, we reproduce the fixed point semantics of P (Zhang, Lin, and Ren 2009).

Fixed Point Semantics. Let P be a positive extended disjunctive deductive database and $\mathcal{I}$ be a set of interpretations, then $\mathcal{T}_P(\mathcal{I}) = \bigcup_{I \in \mathcal{I}} T_P(I)$

$$T_P(I) = \begin{cases} \emptyset, \text{ if } l_{n+1}, \ldots, l_m \subseteq I \text{ for some} \\ \text{ground constraint } \leftarrow l_{n+1} \ldots l_m \text{from } P. \\ \{J \mid \text{for each ground rule} \\ \quad r_i\colon l_0 \vee \cdots \vee l_n \leftarrow l_{n+1} \ldots l_m \text{ such that} \\ \quad \{l_{n+1} \ldots l_m\} \subseteq I, J = I \cup \bigcup_{r_i} J' \text{ where} \\ \quad J' \in Lits(focus(l_0 \vee \cdots \vee l_n, I))\}, \text{otherwise.} \end{cases}$$

In the definition of $T_P(I)$, $focus$ removes complementary literals from disjunction ($focus(l_0 \vee l_1, I) = l_0$ where $I = \{\neg l_1\}$). If all disjuncts ($l_0 \ldots l_n$) are available in I as complementary literals, then the disjunction of literals becomes the conjunction of literals. $Lits$ of conjunction gives a set of conjuncts. On the other hand, $Lits$ of disjunction is a collection of sets where every set in the collection contains a disjunct.

The T_P definition contains the constraint. We write it for the sake of completeness, but our contribution in this paper will not address the constraint.

The following two propositions are vital for our result.

Proposition 1. *For any positive extended disjunctive deductive database P, $\mathcal{T}_P$ is finite and $\mathcal{T}_P \uparrow n$ = $\mathcal{T}_P \uparrow \omega$ where n is a successor ordinal and ω is a limit ordinal.*

Proposition 2. *For any positive extended disjunctive deductive database P, Minimal QC Model(P) = min($\mu(\mathcal{T}_P \uparrow \omega)$ [1]) where min () stands for sets with a minimum number of literals.*

As we stated earlier, we use the paraconsistent relation model (Bagai and Sunderraman 1995) to find the QC model for any given positive extended disjunctive deductive database. In the following, we define the paraconsistent relational model and its operators.

[1] $\mu(\mathcal{T}_P \uparrow \omega) = \{I \mid I \in \mathcal{T}_P \uparrow \omega \text{ and } I \in \mathcal{T}_{\mathcal{P}}(\{I\})\}$

Paraconsistent relations move forward a step to complete the database. Unlike normal relations where we only retain information believed to be true of a particular predicate, we also retain what is believed to be false of a particular predicate in the paraconsistent relational model. Let a relation 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 $t\colon \Sigma \to \bigcup_{A \in \Sigma}$ 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. An ordinary relation on scheme Σ is thus any subset of $\tau(\Sigma)$. A paraconsistent relation on a scheme Σ is a pair $< R^+, R^- >$ where R^+ and R^- are ordinary relations on Σ. Thus, R^+ represents the set of tuples believed to be true of R, and R^- represents the set of tuples believed to be false.

Algebraic Operators. Two types of algebraic operators are defined here: i)Set Theoretic Operators, and ii) Relational Theoretic Operators.

Set Theoretic Operators. Let R and S be two paraconsistent relations on scheme Σ.

Union. The union of R and S, denoted $R \dot{\cup}$S, is a paraconsistent relation on scheme Σ, given that

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

.

Complement. The complement of R, denoted $\dot{-}R$, is a paraconsistent relation on scheme Σ, given that

$$\dot{-}R^+ = R^-, \dot{-}R^- = R^+$$

Intersection. The intersection of R and S, denoted $R \dot{\cap} S$, is a paraconsistent relation on scheme Σ, given that

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

Difference. The difference of R and S, denoted $R \dot{-} S$, is a paraconsistent relation on scheme Σ, given that

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

Example 1. *Let $\{a, b, c\}$ be a common domain for all attribute names, and let R and S be the following paraconsistent relations on schemes $\{X\}$ and $\{X\}$ respectively:*

$$R^+ = \{(a), (b)\}, R^- = \{(c)\}$$
$$S^+ = \{(c), (b)\}, S^- = \{(a)\}$$

$R \dot{\cup} S$ is

$$(R \dot{\cup} S)^+ = \{(a), (b), (c)\}$$
$$(R \dot{\cup} S)^- = \{\}$$

$R \dot{\cap} S$ is

$$(R \dot{\cap} S)^+ = \{(b)\}$$
$$(R \dot{\cap} S)^- = \{(a), (c)\}$$

$\dot{-}R$ is

$$\dot{-}R^+ = \{(c)\}$$
$$\dot{-}R^- = \{(a), (b)\}$$

$R\dot{-}S$ is

$$(R\dot{-}S)^{+} = \{(a)\}$$
$$(R\dot{-}S)^{-} = \{(b), (c)\}$$

Relation Theoretic Operators. Let Σ and Δ be relation schemes such that $\Sigma \subseteq \Delta$, and R and S be paraconsistent relations on schemes Σ and Δ.

Join. The join of R and S, denoted $R\dot{\bowtie}S$, is a paraconsistent relation on scheme $\Sigma \cup \Delta$ given that

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

Projection. The projection of R onto Δ , denoted $\dot{\pi}_{\Delta}(R)$, is a paraconsistent relation on Δ given that

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

where π_{Δ} is the usual projection over Δ of ordinary relations.

Selection. Let F be any logic formula involving attribute names in Σ, constant symbols, and any of these symbols $\{==, \neg, \wedge, \vee\}$. Then the selection of R by F, denoted $\dot{\sigma}_F(R)^{+}$, is a paraconsistent relation on scheme Σ, given that

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

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

The following example is taken from Bagai and Sunderraman's paraconsistent relational data model (Bagai and Sunderraman 1995).

Example 2. *Strictly speaking, relation schemes are sets of attribute names. However, in this example we treat them as ordered sequence 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:*

$R^{+} = \{(b, b), (b, c)\}, R^{-} = \{(a, a), (a, b), (a, c)\}$
$S^{+} = \{(a, c), (c, a)\}, S^{-} = \{(c, b)\}$.

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

$(R\dot{\bowtie}S)^{+} = \{(b, c, a)\}$
$(R\dot{\bowtie}S)^{-} = \{(a, a, a), (a, a, b), (a, a, c), (a, b, a), (a, b, b), (a, b, c), (a, c, a), (a, c, b), (a, c, c), (b, c, b), (c, c, b)\}$

Now, $\dot{\pi}_{\langle X,Z\rangle}(R\dot{\bowtie}S)$ becomes the paraconsistent relation on scheme $\langle X, Z\rangle$:

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

Finally, $\dot{\sigma}_{\neg X=Z}(\dot{\pi}_{\langle X,Z\rangle}(R\dot{\bowtie}S))$ becomes the paraconsistent relation on scheme $\langle X, Z\rangle$:

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

In the rest of the paper, relations mean paraconsistent relations. In order to find the QC model easily in our algorithm, we create a copy for a given relation. For any given relation R, the copy of R is R'. Both R and R' are different relations with the same attributes and the same tuples. R is called an exact relation and R' is called a copy relation. In addition to that, the replica of R is R, where replica R has the same name, the same tuples, and the same attributes. We assume that a relation and its replica should not appear in the same set, but it can appear in different sets. If two relations (a relation and its replica) appear in the same set, then we merge the tuples and write it as one relation.

Our main contributions of the paper start from the disjunctive relation.

3 Disjunctive Relation

Let a disjunctive relation scheme 2^{Σ} be a finite set of attribute sets, where for any attribute set $A \subseteq 2^{\Sigma}$, dom(a) is a non-empty domain of values for each $a \in A$. Let $\tau(2^{\Sigma})$ denote the set of all tuples on 2^{Σ}. A disjunctive relation, DR, over the scheme 2^{Σ} consists of two components $\langle DR^{+}, DR^{-}\rangle$, where $DR^{+} \subseteq \tau(2^{\Sigma})$ and $DR^{-} \subseteq \tau(2^{\Sigma})$. DR^{+} is the component that consists of a set of tuples. While the tuples in DR^{+} typically represent the disjunction of facts, they also sometimes represent the conjunction of facts. At the same time, DR^{-} is the component that consists of a set of tuples. Each tuple in this component represents a conjunction of facts. In the case where the tuple is a singleton, both DR^{+} and DR^{-} have a definite fact that has neither disjunction or conjunction.

Let T be a tuple in DR, then for all $t \in T$, $Att(t)$ be an attribute set that represents the element in the tuple of the disjunctive relation DR, and let $Att(R)$ be an attribute set that represents the relation R over the scheme Σ.

The disjunctive relational model is very much different from the disjunctive database introduced by Molinaro et al. (Molinaro, Chomicki, and Marcinkowski 2009). The disjunctive database (Molinaro, Chomicki, and Marcinkowski 2009) is based on the relational model (not the paraconsistent relational model), whereas disjunctive relational model is based on the paraconsistent relational model. Moreover unlike the disjunctive relational model, disjunctive databases (Molinaro, Chomicki, and Marcinkowski 2009) are well-suited for repairs. At the same time, the disjunctive relational model is also different from the relational model introduced by Sunderraman that can handle disjunction (Sunderraman 1997). In Sunderraman's relation model for disjunction, all the disjuncts should have the same arity in the relation. However, in the disjunctive relational model, it can be different.

In the following, we define rename operators, mapping, and necessary functions, which all play a key role in constructing the QC model. In addition to that, we give an example that relates the usage of the operators, mapping and the functions which help to comprehend the algorithm.

Rename Operators. Rename operators change the attributes for any relation. We define two rename operators: i) Attribute Rename, and ii) Copy Attribute Rename.

Attribute Rename (Θ). Let R be a relation over scheme Σ and $\Sigma = \{A_1 \ldots A_m, R.A_1 \ldots R.A_m\}$, Then

$$\Theta_{A_1 \ldots A_m \rightarrow R.A_1 \ldots R.A_m}(R)$$

and

$$\Theta_{R.A_1 \ldots R.A_m \rightarrow A_1 \ldots A_m}(R)$$

This operator (Θ) is used to maintain uniqueness of attributes between any two relations.

Copy Attribute Rename (Ω). Let R be a relation over scheme Σ and $\Sigma = \{A_1 \ldots A_m, R'_1.A_1 \ldots R'_n.A_m\}$. Then

$$\Omega_{A_1 \ldots A_m \rightarrow R'.A_1 \ldots R'.A_m}(R)$$

and

$$\Omega_{R'.A_1 \ldots R'.A_m \rightarrow A_1 \ldots A_m}(R)$$

Tuple Mapping to Disjunctive Relation. The algebraic equivalent for disjunction ($\vee$) is union. So, we represent the disjunctive information in P as paraconsistent unions ($\dot{\cup}$) of relations. But it is not very flexible to construct the QC model with paraconsistent unions ($\dot{\cup}$) of relations. So, we map the information in relations to a disjunctive relation DR. Let $R_1 \ldots R_n$ be relations over schemes $\Sigma_1 \ldots \Sigma_n$ where every $\Sigma_i \subseteq \Sigma$ and $1 \leq i \leq n$. Then a set of attribute sets for any DR obtained from $R_1 \dot{\cup} \ldots \dot{\cup} R_n$ is $\{\Sigma_1 \ldots \Sigma_n\}$. Now, we map the tuples of relations containing paraconistent unions to a disjunctive relation. For each $t \in T$, T is a tuple for any disjunctive relation (DR). Then t: $\Sigma \rightarrow \cup_{A \in Att(R_i)} dom(A)$ such that $t(A) \in dom(A)$ for every i in $R_1 \dot{\cup} \ldots \dot{\cup} R_n$ where $Att(t) = Att(R_i)$. Informally, a disjunctive relation can be considered a collection of relations that has unions. It is intuitive to map each disjunctive relation back to its base relations because every $t \in T$ of any disjunctive relation represents the corresponding tuple in the relation (R_i).

In our approach, we need to find the name of the underlying relation for any given element in T. Hence, the following defintion:

NRelation. For any $t \in T$ where T is a tuple for any disjunctive relation (DR) that is mapped from $R_1 \dot{\cup} \ldots \dot{\cup} R_n$.

$NRelation(t)$:= $\{R_i \mid Att(t) = Att(R_i)$ for any i in $R_1 \dot{\cup} \ldots \dot{\cup} R_n\ \}$

$NRelation$ returns the corresponding relation name (either positive or negative) for the given t where $t \in T$ and $T \in DR$.

As we stated earlier, the positive component of disjunctive relations contains tuples that are typically disjunctive, but the positive component of disjunctive relations can be conjunctive as well. As we are specifically handling the inconsistencies associated with disjunction, we collect the tuples that are disjunctive.

DISJ. Let DR be a disjunctive relation that is mapped from $R_1 \dot{\cup} \ldots \dot{\cup} R_n$. Then

$$DISJ(DR) = \{T \mid \forall T \in DR^+ \text{ such that } T \text{ is disjunctive }\}$$

The following example is very specific, but helps to understand the algorithm clearly.

Example 3. *Let R_1, R_2 and C are relations over schemes $\{X\}$, $\{Y, Z\}$ and $\{X, Y, Z\}$ and domain for every attribute is $\{a, b, c\}$. Then, we have the following equation:*

$(\dot{\pi}_{\{X,Y,Z\}}(R_1(X) \dot{\cup} \dot{-} R_2(Y, Z)))[X, Y, Z] = (\dot{\pi}_{\{X,Y,Z\}}(C(X, Y, Z)))^+[X, Y, Z]$

where $C^+ = \{(a, b, c)\}$, $R_1^- = \{(b)\}$ *and* $R_2^+ = \{(a, c), (b, c)\}$

Solution. Before the tuples of C are distributed to R_1 and R_2, it is imperative to note that R_1 and R_2 contain definite tuples, which are not disjunctive (conjunctive). The first step is to map the definite tuples of R_1 and R_2 to a disjunctive relation. The definite tuples have no disjunction (conjunction) in any disjunctive relation. So,we rename the attributes (Θ) of R_1 and R_2. Then we map the definite tuples to DR.

In the rest of the paper, we differentiate positive and negative parts of a relation (disjunctive relation) with a double line in every relation (disjunctive relation) diagram.

$DR =$

$\{R_1.X\}$	$\{R_2.Y, R_2.Z\}$
(b)	
	(a, c)
	(b, c)

The next step is to distribute the tuples from C to each individual relation in any union after applying Θ to R_1 and R_2. It is necessary to apply Θ before the distribution of tuples from C because we changed the attributes of R_1 and R_2 before we map the definite tuples.

$R_1 =$

$\{X\}$
(a)
(b)

and $\dot{-}R_2 =$

$\{Y, Z\}$
(b, c)
(a, c)
(b, c)

The next step is to again rename (Θ) the attributes.

$R_1 =$

$\{R_1.X\}$
(a)
(b)

and $\dot{-}R_2 =$

$\{R_2.Y, R_2.Z\}$
(b, c)
(a, c)
(b, c)

Then we map the newly added tuples of $R_1 \dot{\cup} \dot{-}R_2$ to DR.

$DR =$

$\{R_1.X\}$	$\{R_2.Y, R_2.Z\}$
(a) $\vee$	(b, c)
(b)	
	(a, c)
	(b, c)

To find the relation name for any given element in the tuple T where $T \in DR$, we use $NRelation$. So, $NRelation((b, c))$ is $\dot{-}R_2$.

In addition to that, $DISJ(DR)$ is $(a) \vee (b, c)$.

In the following section we introduce $FOCUS_C$ and $FOCUS_D$, which are very essential for handling inconsistencies.

4 Key Idea for QC logic

It is very important to note that the paraconsistent relation portrays a belief system rather than a knowledge system. The key idea of QC logic is given by the resolution rule of inference, which computes the focused belief. If the assumptions are considered as beliefs for the resolution, then the resolvent is called the focused belief. This ensures non-trivial reasoning in QC logic. As an individual can be both true and false for a given relation in the relational model, we decouple the link during the model construction. This is accomplished with the help of $FOCUS_D$ and $FOCUS_C$.

$FOCUS_D$**.** Let DR be a disjunctive relation on scheme 2^Σ and MR be a set of relations. Then

$FOCUS_D(DR, MR) = \{T \mid \forall T \in DISJ(DR) \wedge \exists t \in T \wedge \exists R \in MR \wedge Att(R) = Att(NRelation(t)) \wedge (NRelation(t)$ is positive $\wedge\, t \in R^- \rightarrow (T = T \setminus t)) \vee (NRelation(t)$ is negative $\wedge\, t \in R^+ \rightarrow (T = T \setminus t))))\}$

As a special case, for a given tuple T where $T \in DR^+$, if $FOCUS_D$ removes every element t in tuple T, then we convert the tuple T into a conjunction of the elements in the tuple. This is similar to $focus$ that we defined in the Preliminaries section.

CONJ. Let DR be a disjunctive relation that is mapped from $R_1 \dot\cup \ldots \dot\cup R_n$. For any $T \in DR$,

$CONJ(T)$:= $\{ t_1 \wedge \cdots \wedge t_n \mid \forall t_i \in T \wedge n \leq |T| \}$

Using **CONJ**, we define $FOCUS_C$.

$FOCUS_C$**.** Let DR be a disjunctive relation on scheme 2^Σ and MR be a set of relations. Then

$FOCUS_C(DR, MR) = \{CONJ(T) \mid \forall T \in DR^+ \wedge \forall t \in T \wedge \exists R \in MR \wedge Att(R) = Att(NRelation(t)) \wedge ((NRelation(t)$ is positive $\wedge\, t \in R^-) \vee (NRelation(t)$ is negative $\wedge\, t \in R^+))\}$

$FOCUS_D$ removes any element t , where $t \in T$ and $T \in DR$, that satisfies the predicate of $FOCUS_D$. Similarly, $FOCUS_C$ introduces conjunction among every $t \in T$, where $T \in DR$, that satisfies the predicate of $FOCUS_C$. In any DR, any tuple T that contains conjunction should never be affected by $FOCUS_D$.

Example 4. *Extending from Example 3. Let* $MR = \{R'_2\}$ *where* ${R'_2}^+ = \{(a, c), (b, c)\}$. R'_2 *is called a copy relation.*

Solution.

$MR = \{R'_2\}$ where

$R'_2 =$

$\{Y, Z\}$
(a, c)
(b, c)

We know that,

$DR =$

$\{R_1.X\}$	$\{R_2.Y, R_2.Z\}$
(a) $\vee$	(b, c)
(b)	
	(a, c)
	(b, c)

The attributes of R'_2 is different from R_1 and R_2. Apply $\Theta(R'_2)$, $\Omega(\Theta(R_1))$ and $\Omega(\Theta(R_1))$. So, $FOCUS_C(FOCUS_D)$ can compare the attributes and perform necessary actions on T.

Now we apply focus to DR,

DR= $FOCUS_D$ (DR, MR)

In this case, in DR, the underlying relation for (b, c) is $\dot{-}R_2$ but (b, c) lies in the positive part of R'_2 in MR. Therefore, $FOCUS_D$ removes it.

$DR =$

$\{R_1.X\}$	$\{R_2.Y, R_2.Z\}$
(a)	
(b)	
	(a, c)
	(b, c)

Apply $\Theta(R'_2)$, $\Theta(\Omega(R_1))$ and$\Theta(\Omega(R_2))$. These operations revert the relation back to its old attribute names. To reiterate, DR^+ contains tuples which in turn can contain disjunction. From the base DR, multiple DR can be obtained by applying disjunction in tuples. Each newly created DR from the base DR should not lose any tuple set; otherwise, it leads to incorrect models. The following definition addresses the issue.

Proper Disjunctive Relation (PDR). Let DR be a base disjunctive relation. A proper disjunctive relation is a set, which contains all disjunctive relations that can be formed from DR by applying disjunction in tuples. Concretely, for every disjunctive relation (DR_i), which is obtained from DR by applying disjunction, $\tau(DR^+) = \tau(DR_i^+)$ where 1 $\leq$ i $\leq$ $(2^n - 1)^{\tau(DR^+)}$ such DR_i is a PDR^i.

Example 5. *Continuing from Example 4.*

Solution. The next step is to create a set of proper disjunctive relation from DR.

$PDR = \{PDR^1\}$

$PDR^1 =$

$\{R_1.X\}$	$\{R_2.Y, R_2.Z\}$
(a)	
(b)	
	(a, c)
	(b, c)

The size of PDR is 1. Correspondingly, there should be one replica of base relations. $\{\{(\dot\pi_{\{X,Y,Z\}}(R_1(R_1.X)\dot\cup\dot{-}R_2(R_2.Y, R_2.Z)))[X, Y, Z]\}\}$. For every p in PDR, reverse map tuples to a set of base relations.

$R_1 =$

$\{R_1.X\}$
(a)
(b)

and $\dot{-}R_2 =$

$\{R_2.Y, R_2.Z\}$
(a, c)
(b, c)

Finally, rename (Θ) each attribute name of every relation back to its old name. Hence, R_1 attribute is $< X >$ and R_2 attribute is $< Y, Z >$.

To individualize the relation, we have the following definition.

Relationalize. Let $R_1 \dot\cup \ldots \dot\cup R_n$ and $R_1, \ldots, R_n$ be relations on scheme Σ.

$Relationalize(\dot\pi_{\{\Sigma\}}(R_1 \dot\cup \ldots \dot\cup R_n)[\Sigma]) : = \{R_1, \ldots, R_n\}$

The relationalize operator removes the unions among relations and the projection for it. By doing so, the operator produces a set of relations. If there is a select operation associated with the expression, then apply the operation before $Relationalize$ is applied. $Relationalize$ is in accordance to $Lits$, which is one of the key operators for finding the QC model (Zhang, Lin, and Ren 2009).

Example 6. *Continuing from Example 5.*

Solution. $Relationalize(\dot\pi_{\{Y,X\}}[R_1 \dot\cup \dot{-}R_2](Y, X))$ = $\{R_1,$ $R_2\}$ where

$$R_1 = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline \hline (b) \\ \hline \end{array} \text{ and } \dot{-}R_2 = \begin{array}{|c|}\hline \{Y,Z\} \\ \hline \hline (a,c) \\ \hline (b,c) \\ \hline \end{array} \text{ or } R_2 = \begin{array}{|c|}\hline \{Y,Z\} \\ \hline (a,c) \\ \hline (b,c) \\ \hline \hline \end{array}$$

Now R_1 and R_2 are called focused relations because R_1 and R_2 have no inconsistency.

During QC model construction, we encounter a set of redundant relation sets. In order to remove it, we define the following.

Minimize. Let $\{R1_1 \dots R1_m\}$ and $\{R2_1 \dots R2_n\}$ be two sets of relations where $m \leq n$.

$Minimize(\{\{R1_1 \dots R1_n\}, \{R2_1 \dots R2_m\}\}) : = \{\{R1_1 \dots R1_m\} \mid R1_i = R2_j \wedge Att(R1_i) = Att(R2_j) \wedge \tau(R1_i) = \tau(R2_j)$ such that $\forall i, 1 \leq i \leq m \wedge \exists j, 1 \leq j \leq n\}$

By using the definitions and operators in Section 3 and Section 4, we propose an algorithm in the following section.

5 QC Model for Positive Extended Disjunctive Deductive Databases

By using the algebra of the relational model, we present a bottom up method for constructing the QC model for the positive extended disjunctive deductive database. The algorithm that we present in this section is an extension of the algorithm proposed by Bagai and Sunderraman (Bagai and Sunderraman 1995). The reader is requested to refer to QC logic programs (Zhang, Lin, and Ren 2009) and QC logic (Hunter 2000). The QC model's construction involves two steps. The first step is to convert P into a set of relation definitions for the predicate symbols occuring in P. These definitions are of the form

$$U_r = D_{U_r}$$

where U_r is the paraconsistent union of the disjunctive head predicate symbols of P, and D_{U_r} is an algebraic expression involving predicate symbols of P . Here r refers to the equation number, $1 \leq r \leq$ N, where N refers to a total number of equations. The second step is to iteratively evaluate the expressions in these definitions to incrementally construct the relations associated with the predicate symbols. The first step is called SERALIZE and the second step is called Model Construction.

Algorithm. SERALIZE

Input. A positive extended disjunctive deductive database clause $l_0 \vee \cdots \vee l_n \leftarrow l_{n+1} \dots l_m$. For any i, $0 \leq i \leq m$, l_i is either of the form $p_i(A_{i1} \dots A_{ik_i})$ or $\neg p_i(A_{i1} \dots A_{ik_i})$. 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 $p_i(B_{i1} \dots B_{ik_i})$, and F_i be the conjunction $C_{i1} \wedge \cdots \wedge C_{ik_i}$. If l_i is a positive literal, then Q_i is 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 syntatic 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.
3. Let U be the union ($\dot{\cup}$) of the Q_i's thus obtained, $0 \leq i \leq n$. The output expression is $(\dot{\sigma}_{F_1}(\dot{\pi}_{DV}(U)))$ $[B_{01} \dots B_{n_{kn}}]$ where DV is the set of distinct variables occurring in all l_i.
4. Let E be the natural join ($\dot{\bowtie}$) of the Q_i's thus obtained, $n{+}1 \leq i \leq m$. The output expression is $(\dot{\sigma}_{F_1}(\dot{\pi}_{DV}(E)))$ $[B_{01} \dots B_{n_{kn}}]$. As in step 2, if all conjuncts are true, then $\dot{\sigma}_{F1}$ is dropped from the output expression.

From the algebraic expression of the algorithm, we construct a system of equations.

For any positive extended disjunctive deductive database P, EQN (P) is a set of all equations of the form $U_r = D_{U_r}$, where U_r is a union of the head predicate symbols of P, and D_{U_r} is the union $\dot{\cup}$ of all expressions obtained by the algorithm *SERIALIZE* for clauses in P with the same U_r in their head. If all literals in the head are the same for any two rules, then U_r is the same for those two rules.

The final step is then to construct the model by incrementally constructing the relation values in P. For any positive extended disjunctive deductive database, P_E are the non disjunctive-facts (clauses in P without bodies), and P_B are the disjunctive rules (clauses in P with bodies). P_E^* refers to a set of all ground instances of clauses in P_E. Then, P_I = $P_E^* \cup P_B$.

The following algorithm finds the QC model for P.

ALGORITHM. Model Construction

Input. A positive extended disjunctive deductive database (P)

Output. Minimal QC Model for P.

Method: The values are computed by the following steps.

1. (Initialization)
   (a) Compute EQN(P_I) using the algorithm *SERIALIZE* for each clause in P_I.
   (b) SModel= $\emptyset$, For each predicate symbol p in P_E, set p^+= $\{\{a1 \dots ak\} \mid p(a1 \dots, ak) \in P_E^*\}$, and p^-= $\emptyset$ or p^-= $\{\{a1 \dots ak\} \mid \neg p(a1, \dots ak) \in P_E^*\}$ and p^+= $\emptyset$
   SModel=p
   End for.
2. (Rule Application)
   (a) DModel= $\emptyset$.
   For every SModel (SModel $\neq \emptyset$), create copies of the relations in SModel and replace the SModel with the copies.
   (b) For every equation r of the form $U_r = D_{U_r}$, create DR_r and insert the tuples from the copies in SModel into the corresponding exact relation in the equation r. Then map the definite tuples for the relations in U_r to

DR_r. Compute the expression D_{U_r} and set the relations in U_r with $D^+_{U_r}$.

(c) Map the newly added tuples of U_r to DR_r. Apply Θ and Ω to every relation in U_r. Also apply Θ to every relation in SModel. Then

$$DR_r = FOCUS_C(DR_r, SModel)$$

$$DR_r = FOCUS_D(DR_r, SModel)$$

Repeat $FOCUS_D$ until there is no change in DR_r. When there is no change is DR_r, apply Θ to every relation in SModel and apply Ω and Θ to every relation in U_r.

(d) Create a set of proper disjunctive relations (PDR_r) from the focused DR_r.

(e) Delete all tuples for the relations in U_r and create multiple replicas of U_r, which is denoted by the set C_r, where $|C_r| = |PDR_r|$.

(f) Re-map each p in PDR_r to C where $C \in C_r$.
For every $C \in C_r$,
C= $Relationalize(C)$
/* C_r contains a collection of set of relations. */
DModel = DModel $\bigcup C_r$
/* Merging relations of every equation */

(g) Once all equations are evaluated for the current SModel, perform the following: i) for every $M \in$ DModel and for every exact relation for SModel that is not in M, create the exact relation in M, and ii) for every $M \in$ DModel and for every exact relation for SModel that is in M, insert the tuples from the copy relation in SModel into the exact relation. Then add DModel to TempModel.

(h) Once every SModel is applied, start from step 2 (a) with SModel=$Minimize$ (TempModel) and stop when there is no change in SModel.

3. Minimal QC Model: Pick one (many) set (s) in SModel whose sum of the size of all relations in the set (s) is (are) minimal.

It is very intuitive from the algorithm that if the computation of D_{U_r} is empty for any SModel, then discard the SModel. We found that the algorithm should be extended a little to accommodate disjunctive facts, duplicate variables in disjunctive literals, and constants in disjunctive literals.

The following example shows how the algorithm works. In the example, to show the difference between any two sets, we superscript the set with a number.

Example 7. *Let P be a positive extended disjunctive deductive database. It has the following facts and rules:*
$r(a,c), p(a), p(c), \neg f(a,b), s(c)$
$w(X) \vee g(X) \vee \neg p(X) \leftarrow r(X,Y), s(Y)$
$w(X) \vee g(X) \vee \neg p(X) \leftarrow \neg f(X,Y)$

Solution. By step 1 (a) in initialization,
$w(X) \vee g(X) \vee \neg p(X) \leftarrow r(X,Y), s(Y)$ is serialized to $(\dot\pi_{\{X\}}(w(X)\dot\cup g(X)\dot\cup \dot- p(X))[X]$= $(\dot\pi_{\{X\}}(r(X,Y)\dot\bowtie s(Y)))^+[X]$
and $w(X) \vee g(X) \vee \neg p(X) \leftarrow \neg f(X,Y)$ is serialized to $(\dot\pi_{\{X\}}(w(X)\dot\cup g(X)\dot\cup \dot- p(X))[X]$= $(\dot\pi_{\{X\}}(\dot- f(X,Y)))^+[X]$

Both equations that are obtained after serialization have the same left-hand side expression. So, it is written as one equation (as show in (1)). EQN(P_I) returns:

1. $(\dot\pi_{\{X\}}(w(X)\dot\cup g(X)\dot\cup \dot- p(X))[X]$= $(\dot\pi_{\{X\}}(r(X,Y)\dot\bowtie s(Y)))^+[X]\dot\cup(\dot\pi_{\{X\}}(\dot- f(X,Y)))^+[X]$

After step 1 (b) in initialization, SModel = $\{r, p, s, f\}$ where

$r =$

$\{X,Y\}$
(a,c)

$p =$

$\{X\}$
(a)
(c)

$s =$

$\{Y\}$
(c)

$f =$

$\{X,Y\}$
(a,b)

After step 2(a), SModel = $\{r', p', s', f'\}$ (COPIES) where

$r' =$

$\{X,Y\}$
(a,c)

$p' =$

$\{X\}$
(a)
(c)

$s' =$

$\{Y\}$
(c)

$f' =$

$\{X,Y\}$
(a,b)

In step 2 (b), there is only one SModel and an equation. It is necessary to insert the tuples from the copies in SModel to the corresponding relations in the equation. DModel= $\emptyset$. Then map the definite tuples to DR_1 for the current SModel.

$DR_1 =$

$\{w.X\}$	$\{g.X\}$	$\{p.X\}$
		(a)
		(c)

Compute the equation and assign it to U_1. Map the newly added (disjunctive) tuples to DR_1.

$DR_1 =$

$\{w.X\}$	$\{g.X\}$	$\{p.X\}$
(a) $\vee$	(a) $\vee$	(a)
		(a)
		(c)

By step 2 (c), DR_1 = $FOCUS_D(DR_1, SModel)$

$DR_1 =$

$\{w.X\}$	$\{g.X\}$	$\{p.X\}$
(a) $\vee$	(a)	
		(a)
		(c)

By step 2 (d), PDR_1 = $\{PDR_1^1, PDR_1^2, PDR_1^3\}$

$PDR_1^1 =$

$\{w.X\}$	$\{g.X\}$	$\{p.X\}$
(a)		
		(a)
		(c)

$PDR_1^2 =$

$\{w.X\}$	$\{g.X\}$	$\{p.X\}$
	(a)	
		(a)
		(c)

$PDR_1^3 =$

$\{w.X\}$	$\{g.X\}$	$\{p.X\}$
(a) $\vee$	(a)	
		(a)
		(c)

Map every p in PDR_1 back to a set of base relations. We skip a step (2 (d)) here. After relationalizing the set of relations (step 2 (f)), we write:
C_1= $\{\{ w, p \}^1, \{g, p\}^2, \{w, g, p \}^3\}$

$\{\ w, p\ \}^1$

$w = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline\hline \end{array}$ $p = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline (c) \\ \hline\hline \end{array}$

$\{\ g, p\ \}^2$

$g = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline\hline \end{array}$ $p = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline (c) \\ \hline\hline \end{array}$

$\{\ w, g, p\ \}^3$

$w = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline\hline \end{array}$ $g = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline\hline \end{array}$ $p = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline (c) \\ \hline\hline \end{array}$

DModel= DModel $\bigcup C_1$

By step 2 (g),

DModel = $\{\{w, p, r, s, f\}^1$, $\{g, p, r, s, f\}^2$, $\{w, g, p, r, s, f\ \}^3\}$

$\{\ w, p, r, s, f\ \}^1$

$w = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline\hline \end{array}$ $p = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline (c) \\ \hline\hline \end{array}$ $r = \begin{array}{|c|}\hline \{X, Y\} \\ \hline (a, c) \\ \hline\hline \end{array}$ $s = \begin{array}{|c|}\hline \{Y\} \\ \hline (c) \\ \hline\hline \end{array}$

$f = \begin{array}{|c|}\hline \{X, Y\} \\ \hline\hline (a, b) \\ \hline \end{array}$

$\{\ g, p, r, s, f\ \}^2$

$g = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline\hline \end{array}$ $p = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline (c) \\ \hline\hline \end{array}$ $r = \begin{array}{|c|}\hline \{X, Y\} \\ \hline (a, c) \\ \hline\hline \end{array}$ $s = \begin{array}{|c|}\hline \{Y\} \\ \hline (c) \\ \hline\hline \end{array}$

$f = \begin{array}{|c|}\hline \{X, Y\} \\ \hline\hline (a, b) \\ \hline \end{array}$

$\{\ w, g, p, r, s, f\ \}^3$

$w = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline\hline \end{array}$ $g = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline\hline \end{array}$ $p = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline (c) \\ \hline\hline \end{array}$ $r = \begin{array}{|c|}\hline \{X, Y\} \\ \hline (a, c) \\ \hline\hline \end{array}$

$s = \begin{array}{|c|}\hline \{Y\} \\ \hline (c) \\ \hline\hline \end{array}$ $f = \begin{array}{|c|}\hline \{X, Y\} \\ \hline\hline (a, b) \\ \hline \end{array}$

Add DModel to TempModel.

By step 2 (h), SModel=$Minimize$ (TempModel)

The algorithm stops when there is no change in SModel. We then skip further iterations and write the final result:

Minimal QC Model = $\{\ \{\ w, p, r, s, f\}^1, \{g, p, r, s, f\}^2\ \}$

$\{\ w, p, r, s, f\ \}^1$

$w = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline\hline \end{array}$ $p = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline (c) \\ \hline\hline \end{array}$ $r = \begin{array}{|c|}\hline \{X, Y\} \\ \hline (a, c) \\ \hline\hline \end{array}$ $s = \begin{array}{|c|}\hline \{Y\} \\ \hline (c) \\ \hline\hline \end{array}$

$f = \begin{array}{|c|}\hline \{X, Y\} \\ \hline\hline (a, b) \\ \hline \end{array}$

$\{\ g, p, r, s, f\ \}^2$

$g = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline\hline \end{array}$ $p = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline (c) \\ \hline\hline \end{array}$ $r = \begin{array}{|c|}\hline \{X, Y\} \\ \hline (a, c) \\ \hline\hline \end{array}$ $s = \begin{array}{|c|}\hline \{Y\} \\ \hline (c) \\ \hline\hline \end{array}$

$f = \begin{array}{|c|}\hline \{X, Y\} \\ \hline\hline (a, b) \\ \hline \end{array}$

In other words, Minimal QC Model = $\{\{w(a), p(a), p(c), r(a, c), s(c), \neg f(a, b)\}, \{g(a), p(a), p(c), r(a, c), s(c), \neg f(a, b)\}\}$

Gelfond and Lifschitz adopt the way of trivializing results (Gelfond and Lifschitz 1991) while the algorithm tolerates inconsistencies. However, we observe that we have not proven the CORRECTNESS of the algorithm. Our immediate future work is to prove that the algorithm mimics fixed point semantics (Proposition 1 and Proposition 2) (Zhang, Lin, and Ren 2009).

6 Conclusion

In this paper, we proposed an algorithm to find the QC model for any positive extended disjunctive deductive database. We also introduced the disjunctive relational model to represent the relations containing paraconsistent unions. The algorithm that we presented here is based on the algorithm that is used to compute the well-founded model for general deductive databases by using the relational model (Bagai and Sunderraman 1996a). In query-intensive applications, this precomputation of the model enables efficient processing of subsequent queries. Though we find the model for any given positive extended disjunctive deductive database, the algorithm does not find models for the databases with recursions and constraints. One direction of future work could be expanding the algorithm to allow recursions and constraints. Moreover, the model that we construct is too strong; it causes disjunction introduction to fail, but it is supported by QC logic. To compute the QC entailment, it is necessary to have both weak and strong models. So another direction of future work could be finding the weak models for the same program so that QC entailment could be achieved. The creation of many proper disjunctive databases are expensive, given the QC logic model computation, and are probably not worth the extra computation. We notice that we have not stated or proven the complexities of the algorithm, and we have also left it for future work.

References

Alcântara, J.; Damásio, C. V.; and Pereira, L. M. 2002. Paraconsistent logic programs. In *Logics in Artificial Intelligence*. Springer. 345–356.

Alcântara, J.; Damásio, C. V.; and Pereira, L. M. 2004. A declarative characterisation of disjunctive paraconsistent answer sets. In *ECAI*, volume 16, 951. Citeseer.

Arieli, O. 2002. Paraconsistent declarative semantics for extended logic programs. *Annals of Mathematics and Artificial Intelligence* 36(4):381–417.

Bagai, R., and Sunderraman, R. 1995. A paraconsistent relational data model. *International Journal of Computer Mathematics* 55(1-2):39–55.

Bagai, R., and Sunderraman, R. 1996a. Bottom-up computation of the fitting model for general deductive databases. *Journal of Intelligent Information Systems* 6(1):59–75.

Bagai, R., and Sunderraman, R. 1996b. Computing the well-founded model of deductive databases. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 4(02):157–175.

Belnap Jr, N. D. 1977. A useful four-valued logic. In *Modern uses of multiple-valued logic*. Springer. 5–37.

Damásio, C. V., and Pereira, L. M. 1998. A survey of paraconsistent semantics for logic programs. In *Reasoning with Actual and Potential Contradictions*. Springer. 241–320.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New generation computing* 9(3-4):365–385.

Hunter, A. 1998. Paraconsistent logics, handbook of defeasible reasoning and uncertainty management systems: volume 2: reasoning with actual and potential contradictions.

Hunter, A. 2000. Reasoning with contradictory information using quasi-classical logic. *Journal of Logic and Computation* 10(5):677–703.

Minker, J., and Seipel, D. 2002. Disjunctive logic programming: A survey and assessment. In *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part I*, 472–511. Springer-Verlag.

Molinaro, C.; Chomicki, J.; and Marcinkowski, J. 2009. Disjunctive databases for representing repairs. *Annals of Mathematics and Artificial Intelligence* 57(2):103–124.

Sakama, C., and Inoue, K. 1995. Paraconsistent stable semantics for extended disjunctive programs. *Journal of Logic and Computation* 5(3):265–285.

Subrahmanian, V. 1990. Paraconsistent disjunctive deductive databases. In *Multiple-Valued Logic, 1990., Proceedings of the Twentieth International Symposium on*, 339–346. IEEE.

Sunderraman, R. 1997. Modeling negative and disjunctive information in relational databases. In *Database and Expert Systems Applications*, 337–346. Springer.

Zhang, Z.; Lin, Z.; and Ren, S. 2009. Quasi-classical model semantics for logic programs–a paraconsistent approach. In *Foundations of Intelligent Systems*. Springer. 181–190.