Repairing Databases with Annotated Predicate Logic

Pablo Barceló
P. Universidad Católica de Chile
Depto. Ciencia de Computación
Santiago, Chile.
pbarcelo@ing.puc.cl

Leopoldo Bertossi
Carleton University
School of Computer Science
Ottawa, Canada.
bertossi@scs.carleton.ca

Abstract

Consistent answers from a relational database that violates a given set of integrity constraints are characterized [Arenas et al. 1999] as ordinary answers that can be obtained from *every* repaired version of the database. In this paper we address the problem of specifying the repairs of a database as the minimal models of a theory written in *Annotated Predicate Logic* [Kifer et al. 1992a]. The specification is then transformed into a disjunctive logic program with annotation arguments and a stable model semantics. From the program, consistent answers to first order queries are obtained.

1 Introduction

Integrity constraints (ICs) are important in the design and use of a relational database. They embody the semantics of the application domain and help maintain the correspondence between that application domain and its model provided by the database. Nevertheless, it is not strange for a database instance to become inconsistent with respect to a given, expected set of ICs. This could happen due to different factors, being one of them the integration of several data sources. The integration of consistent databases may easily lead to an inconsistent integrated database.

An important problem in databases consists in retrieving answers to queries that are "consistent" with the given ICs, even when the database as a whole does not satisfy those ICs. Very likely "most" of the data is still consistent. The notion of consistent answers to a first order (FO) query was defined in [Arenas et al. 1999], where also a computational mechanism for obtaining them was presented. Intuitively speaking, an ground tuple $\bar{t}$ is a consistent answer to a first order query $Q(\bar{x})$ in a, possibly inconsistent, relational database instance DB if it is an (ordinary) answer to $Q(\bar{x})$ in every minimal repair of DB, that is in every database instance over the same schema that differs from DB by a minimal (under set inclusion) set of inserted or deleted tuples.

That mechanism presented in [Arenas et al. 1999] has some limitations in terms of the ICs and queries that can be handled. In [Arenas et al. 2000b], a more general methodology based on logic programs with an stable model semantics was introduced. More general queries could be considered, but ICs were restricted to be "binary", i.e. universal with at most two database literals (plus built-in formulas).

For consistent query answering we need to deal with *all* the repairs of a database. In consequence, a natural approach consists in providing a manageable logical specification of the class of database repairs. The specification must include information about (from) the database and the information contained in the ICs. Since these two pieces of information are mutually inconsistent, we need a logic that does not collapse in the presence of contradictions. A logic like *Annotated Predicate Logic* (APC) [Kifer et al. 1992a], for which a classically inconsistent set of premises can still have a model, is a natural candidate.

In [Arenas et al. 2000a], a new declarative semantic framework was introduced for studying the problem of query answering in databases that are inconsistent with integrity constraints. This was done by embedding both the database instance and the integrity constraints into a single theory written in APC,

with an appropriate non classical truth-values lattice *Latt*. It was shown that, for universal ICs, there is a one to one correspondence between some minimal models of the annotated theory and the repairs of the inconsistent database. In this way, a logical specification of the database repairs was achieved. The annotated theory was used to obtain some algorithms for obtaining consistent answers to some simple first order queries.

This paper extends the results presented in [Arenas et al. 2000a] in several ways. First, in section 3, we show how to annotate and integrate referential ICs (that contain existential quantifiers) in addition to universal ICs into the annotated theory. The correspondence between minimal models of the theory and the database repairs is established. Next, in section 4, we show how to annotate queries and the formulation of the problem of consistent query answering as a problem of non-monotonic (minimal) entailment from the annotated theory. Then, in section 5.1, on the basis of the generated annotated theory, disjunctive logic programs with annotation arguments are derived in such a way that they specify the database repairs. After that, in section 5.2, we show how to use those programs to obtain consistent answers to first order queries. In section 5.3 the logic programs are transformed into classical disjunctive normal programs with a stable model semantics; and the *coherent* stable models become the database repairs. In section 2 we present the basic framework, and in section 6 we draw some conclusions, mention ongoing work, and consider related work.

The methodology presented here works for arbitrary first order queries and arbitrary universal ICs, what considerable extends the cases that could be handled in [Arenas et al. 1999, Arenas et al. 2000b, Arenas et al. 2000a].

2 Preliminaries

2.1 Database repairs and consistent answers

In the context of relational databases, we will consider a fixed relational schema $\Sigma = (D, P \cup B)$ that determines a first order language. It consists of a fixed, possibly infinite, database domain $D = \{c_1, c_2, ...\}$, a fixed set of database predicates $P = \{p_1, \ldots, p_n\}$, and a fixed set of built-in predicates $B = \{e_1, \ldots, e_m\}$.

A database instance over Σ is a finite collection DB of facts of the form $p(c_1, ..., c_n)$, where p is a predicate in P and $c_1, ..., c_n$ are constants in D. Built-in predicates have a fixed and same extension in every database instance, not subject to changes.

An *integrity constraint* (IC) is an implicitly quantified clause of the form

$$q_1(\bar{t}_1) \vee \cdots \vee q_n(\bar{t}_n) \vee \neg p_1(\bar{s}_1) \vee \cdots \vee \neg p_m(\bar{s}_m) \quad (1)$$

in the FO language of Σ, where each p_i, q_j is a predicate in $P \cup B$ and the $\bar{t}_i, \bar{s}_j$ are tuples containing constants and variables. We assume we have a fixed set IC of ICs.

We will assume that DB and IC, separately, are consistent theories. Nevertheless, it may be the case that $DB \cup IC$ is inconsistent. Equivalently, if we associate to DB a first order structure, also denoted with DB, in the natural way, i.e. by applying the closed world assumption that makes false any ground atom not explicitly appearing in the set of atoms DB, it may happen that DB, as a structure, does not satisfy the IC. We denote with $DB \models_\Sigma IC$ the fact that the database satisfies IC. In this case we say that DB is consistent wrt IC; otherwise we say DB is inconsistent.

As in [Arenas et al. 1999], we define the *distance* between two database instances DB_1 and DB_2 as their symmetric difference $\Delta(DB_1, DB_2) = (DB_1 - DB_2) \cup (DB_2 - DB_1)$.

Now, given a database instance DB, possibly inconsistent wrt IC, we say that the instance DB' is a *repair* of DB iff $DB' \models_\Sigma IC$ and $\Delta(DB, DB')$ is minimal under set inclusion in the class of instances that satisfy IC and conform to schema Σ [Arenas et al. 1999].

Example 1. Consider the relational schema $Book(author, name, publYear)$, a database instance $DB = \{Book(kafka, metamorph, 1915), Book(kafka, metamorph, 1919)\}$; and the functional dependency $FD : author, name \rightarrow publYear$, that can be expressed by $IC : \neg Book(x, y, z) \vee \neg Book(x, y, w) \vee z = w$. Instance DB is inconsistent with respect to IC. The original instance has two possible repairs: $DB_1 = \{Book(kafka, metamorph, 1915)\}$, and $DB_2 = \{Book(kafka, metamorph, 1919)\}$. □

Let DB be a database instance, possibly not satisfying a set IC of integrity constraints. Given a query $Q(\bar{x})$ to DB, we say that a tuple of constants $\bar{t}$ is a *consistent answer* to $Q(\bar{x})$ in DB

[Arenas et al. 1999], denoted $DB \models_c Q(\bar{t})$, if for every repair DB' of DB, $DB' \models_\Sigma Q(\bar{t})$. If Q is a closed formula, *i.e.* a sentence, then *true* is a *consistent answer* to Q, denoted $DB \models_c Q$, if for every repair DB' of DB, $DB' \models_\Sigma Q$.

Example 2. (example 1 continued) The query Q_1 : $Book(kafka, metamorph, 1915)$ does not have *true* as a consistent answer, because it is not true in every repair. Query $Q_2(y)$: $\exists x \exists z Book(x, y, z)$ has $y = metamorph$ as a consistent answer. Query $Q_3(x) : \exists z Book(x, metamorph, z)$ has $x = kafka$ as a consistent answer. □

2.2 Annotating DBs and ICs

Annotated Predicate Calculus was introduced in [Kifer et al. 1992a] and also studied in [Blair et al. 1989] and [Kifer et al. 1992b]. It constitutes a non classical logic, where classically inconsistent information does not unravel logical inference, reasoning about causes of inconsistency becomes possible, making one of its goals to study the differences in the contribution to the inconsistency made by the different literals in a theory, what is related to the the problem of consistent query answers.

The syntax of *APC* is similar to that of classical logic, except for the fact that the atoms are annotated with values drawn from a *truth-values lattice*. The lattice *Latt* we will use throughout this paper is shown in Figure 1, first introduced in [Arenas et al. 2000a].

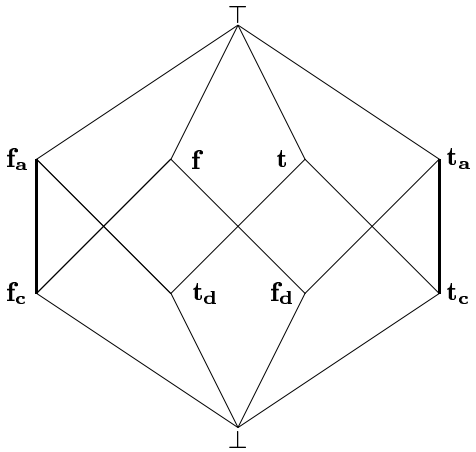

Figure 1: *Latt* with *constraints values*, *database values* and *advisory values*

Intuitively, we can think of values $\mathbf{t_c}$ and $\mathbf{f_c}$ as specifying what is needed for constraint satisfaction. The values $\mathbf{t_d}$ and $\mathbf{f_d}$ represent the truth values according to the original database. Finally, $\mathbf{t_a}$ and $\mathbf{f_a}$ are considered *advisory* truth values. These are intended to solve conflicts between the original database and what is needed for the satisfaction of the integrity constraints. Notice that $lub(\mathbf{t_d}, \mathbf{f_c}) = \mathbf{f_a}$ and $lub(\mathbf{f_d}, \mathbf{t_c}) = \mathbf{t_a}$. The intuition behind is that, in case of a conflict between the constraints and the database, we should obey the constraints, because the database instance only can be changed to restore consistency. This lack of symmetry between data and ICs is captured by the lattice. Advisory value $\mathbf{t_a}$ is an indication that the atom annotated with it must be inserted into the DB; and deleted from the DB when annotated with $\mathbf{f_a}$.

Herbrand interpretations are now sets of annotated ground atoms. The notion of formula satisfaction in an *Herbrand interpretation* I is defined classically, except for atomic formulas p, where we say that $I \models p:\mathbf{s}$, with $\mathbf{s} \in Latt$, iff for some $\mathbf{s}'$ such that $\mathbf{s} \leq \mathbf{s}'$ we have that $p:\mathbf{s}' \in I$ [Kifer et al. 1992a].

Given an *APC* theory $\mathcal{T}$, we say that an Herbrand interpretation I is a Δ-minimal model of $\mathcal{T}$, with $\Delta = \{\mathbf{t_a}, \mathbf{f_a}\}$, if I is a model of $\mathcal{T}$ and no other model of $\mathcal{T}$ has a proper subset of atoms annotated with elements in Δ, *i.e.* the set of atoms annotated with $\mathbf{t_a}$ or $\mathbf{f_a}$ in I is minimal under set inclusion.

Given a database instance DB and a set of integrity constraints IC of the form (1), an embedding $\mathcal{T}(DB, IC)$ of DB and IC into a new *APC* theory can be defined [Arenas et al. 2000a] in order to restore consistency using the annotations in the lattice. It was shown in [Arenas et al. 2000a] that there is a one-to-one correspondence between the Δ-minimal models (we will simply say "minimal" in the rest of the paper) of theory $\mathcal{T}(DB, IC)$ and the repairs of the original database instance. Repairs can be obtained from minimal models as follows:

Definition 1. Given a minimal model $\mathcal{M}$ of $\mathcal{T}(DB, IC)$, the corresponding DB instance is defined by $DB_\mathcal{M} = \{p(\bar{a}) \mid \mathcal{M} \models p(\bar{a}):\mathbf{t} \vee p(\bar{a}):\mathbf{t_a}\}$. □

Example 3. (example 1 cont.) The embedding $\mathcal{T}(DB)$ of DB into *APC* is given by the following formulas:

1. $Book(kafka, metamorph, 1915):\mathbf{t_d}$
   $Book(kafka, metamorph, 1919):\mathbf{t_d}$.

2. Predicate closure axioms:

   $((x = kafka)\text{:}\mathbf{t_d} \wedge (y = metamorph)\text{:}\mathbf{t_d} \wedge (z = 1915)\text{:}\mathbf{t_d}) \vee$
   $((x = kafka)\text{:}\mathbf{t_d} \wedge (y = metamorph)\text{:}\mathbf{t_d} \wedge (z = 1919)\text{:}\mathbf{t_d}) \vee$
   $Book(x, y, z)\text{:}\mathbf{f_d}$.

   Every ground atom that is not in DB is (possibly implicitly) annotated with $\mathbf{f_d}$.

The embedding $\mathcal{T}(IC)$ of IC into APC is given by:

3. $Book(x, y, z)\text{:}\mathbf{f_c} \vee Book(x, y, w)\text{:}\mathbf{f_c} \vee (z = w)\text{:}\mathbf{t_c}$.

4. $Book(x, y, z)\text{:}\mathbf{f_c} \vee Book(x, y, z)\text{:}\mathbf{t_c}$
   $\neg Book(x, y, z)\text{:}\mathbf{f_c} \vee \neg Book(x, y, z)\text{:}\mathbf{t_c}$.[1]

   These formulas specify that every fact must have one and just one constraint value.

Furthermore

5. For every true *built-in* atom φ we include $\varphi\text{:}\mathbf{t}$ in $\mathcal{T}(B)$, and $\varphi\text{:}\mathbf{f}$ for every false built-in atom, e.g. $(1915 = 1915)\text{:}\mathbf{t}$, but $(1915 = 1919)\text{:}\mathbf{f}$.

The Δ-minimal models of $\mathcal{T}(DB, IC) = \mathcal{T}(DB) \cup \mathcal{T}(IC) \cup \mathcal{T}(B)$ are:

$\mathcal{M}_1 = \{Book(kafka, metamorph, 1915)\text{:}\mathbf{t}, Book(kafka, metamorph, 1919)\text{:}\mathbf{f_a}\}$,

$\mathcal{M}_2 = \{Book(kafka, metamorph, 1915)\text{:}\mathbf{f_a}, Book(kafka, metamorph, 1919)\text{:}\mathbf{t}\}$,

plus annotated false DB atoms and built-ins in both cases . The correspondent database instances, $DB_{\mathcal{M}_1}, DB_{\mathcal{M}_2}$ are the repairs of DB shown in example 1. □

From the definition of the lattice and the fact that no atom from the database is annotated with both $\mathbf{t_d}$ and $\mathbf{f_d}$, it is possible to show that, in the minimal models of the annotated theory, a DB atom may get the annotations either $\mathbf{t}$ or $\mathbf{f_a}$ if the atom was annotated with $\mathbf{t_d}$, and either $\mathbf{f}$ or $\mathbf{t_a}$ if the atom was annotated with $\mathbf{f_d}$. In the transition from the annotated theory to its minimal models, the annotations $\mathbf{t_d}, \mathbf{f_d}$ "disappear", as we wished the atoms to be annotated in the highest possible layer in the lattice, except for $\top$ if possible. Actually, in the minimal models $\top$ can always be avoided.

[1] Since only atomic formulas are annotated, the non atomic formula $\neg p(\bar{x})\text{:}\mathbf{s}$ is to be read as $\neg(p(\bar{x})\text{:}\mathbf{s})$. We will omit the parenthesis though.

3 Annotating Referential ICs

Referential integrity constraints (RICs) like

$$p(\bar{x}) \rightarrow \exists y q(\bar{x}', y), \tag{2}$$

where the variables in $\bar{x}'$ are a subset of the variables in $\bar{x}$, cannot be expressed as an equivalent clause of the form (1). RICs are important and common in databases. For that reason, we need to extend our embedding methodology. Actually, we embed (2) into APC by means of

$$p(\bar{x})\text{:}\mathbf{f_c} \vee \exists y(q(\bar{x}', y)\text{:}\mathbf{t_c}). \tag{3}$$

Now we allow the given set of ICs to contain, in addition to ICs of the form (1), RICs like (2). The one-to-one correspondence between minimal models of the new theory $\mathcal{T}(DB, IC)$ and the repairs of DB still holds. Most important for us is to obtain repairs from minimal models.

Proposition 1. Let $\mathcal{M}$ be a model of $\mathcal{T}(DB, IC)$. If $\mathcal{M}$ is minimal and $DB_{\mathcal{M}}$ is finite, then $DB_{\mathcal{M}}$ is a repair of DB with respect to IC. □

Example 4. Consider the relational schema of Example 1 extended with table $Author(name, citizenship)$. Now, IC also contains the RIC: $Book(x, y, z) \rightarrow \exists w Author(x, w)$, expressing that every writer of a book in the database instance must be registered as an author. The theory $\mathcal{T}(IC)$ now also contains:

$$Book(x, y, z)\text{:}\mathbf{f_c} \vee \exists w(Author(x, w)\text{:}\mathbf{t_c}),$$
$$Author(x, w)\text{:}\mathbf{f_c} \vee Author(x, w)\text{:}\mathbf{t_c},$$
$$\neg Author(x, w)\text{:}\mathbf{f_c} \vee \neg Author(x, w)\text{:}\mathbf{t_c}.$$

We might also have the functional dependency $FD : name \rightarrow citizenship$, that in conjunction with the RIC, produces a foreign key constraint. The database instance $\{Book(neruda, 20lovepoems, 1924)\}$ is inconsistent wrt the given RIC. If we have the following subdomain $D(Author.citizenship) = \{chilean, canadian\}$ for the attribute "citizenship", we obtain the following database theory:

$\mathcal{T}(DB) = \{Book(neruda, 20lovepoems, 1924)\text{:}\mathbf{t_d}, Author(neruda, chilean)\text{:}\mathbf{f_d}, Author(neruda, canadian)\text{:}\mathbf{f_d}, \ldots\}$.

The minimal models of $\mathcal{T}(DB, IC)$ are:

$\mathcal{M}_1 = \{Book(neruda, 20lovepoems, 1924)\text{:}\mathbf{f_a}, Author(neruda, chilean)\text{:}\mathbf{f}$,

$Author(neruda, canadian)$:$\mathbf{f}, \dots\}$
$\mathcal{M}_2 = \{Book(neruda, 20lovepoems, 1924)$:$\mathbf{t}$,
$Author(neruda, chilean)$:$\mathbf{t_a}$,
$Author(neruda, canadian)$:$\mathbf{f}, \dots\}$
$\mathcal{M}_3 = \{Book(neruda, 20lovepoems, 1924)$:$\mathbf{t}$,
$Author(neruda, chilean)$:$\mathbf{f}$,
$Author(neruda, canadian)$:$\mathbf{t_a}, \dots\}$.

We obtain $DB_{\mathcal{M}_1} = \emptyset$, $DB_{\mathcal{M}_2} = \{Book(neruda, 20lovepoems, 1924), Author(neruda, chilean)\}$ and $DB_{\mathcal{M}_3}$ similar to $DB_{\mathcal{M}_2}$, but with a Canadian Neruda. According to proposition 1, these are repairs of the original database instance, actually the only ones. □

As in [Arenas et al. 2000a], it can be proved that when the original instance is consistent, then it is its only repair and it corresponds to a unique minimal model of the APC theory.

4 Annotation of Queries

According to proposition 1, a ground tuple $\bar{t}$ is a consistent answer to a FO query $Q(\bar{x})$ iff $Q(\bar{t})$ is true of every minimal model of $\mathcal{T}(DB, IC)$. However, if we want to pose the query directly to the theory, it is necessary to reformulate it as an annotated formula.

Definition 2. Given a FO query $Q(\bar{x})$ in language Σ, we denote by $Q^{an}(\bar{x})$ the annotated formula obtained from Q by simultaneously replacing, for $p \in P$, the negative literal $\neg p(\bar{s})$ by the APC formula $p(\bar{s})$:$\mathbf{f} \vee p(\bar{s})$:$\mathbf{f_a}$, and the positive literal $p(\bar{s})$ by the APC formula $p(\bar{s})$:$\mathbf{t} \vee p(\bar{s})$:$\mathbf{t_a}$. For $p \in B$, the literal $p(\bar{s})$ is replaced by the APC formula $p(\bar{s})$:$\mathbf{t}$. □

According to this definition, logically equivalent versions of a query could have different annotated versions, but it can be shown (proposition 2), that they retrieve the same consistent answers.

Example 5. (example 1 cont.) If we want the consistent answers to the query $Q(x)$: $\neg\exists y\exists z\exists w\exists t(Book(x, y, z) \wedge Book(x, w, t) \wedge y \neq w)$, asking for those authors that have at most one book in DB, we generate the annotated query $Q^{an}(\bar{x})$: $\neg\exists y\exists z\exists w\exists t((Book(x, y, z) : \mathbf{t} \vee Book(x, y, z) : \mathbf{t_a}) \wedge (Book(x, w, t) : \mathbf{t} \vee Book(x, w, t) : \mathbf{t_a}) \wedge (y \neq w) : \mathbf{t})$, to be posed to the annotated theory with its minimal model semantics. □

Definition 3. If φ is a sentence in the language of $\mathcal{T}(DB, IC)$, we say that $\mathcal{T}(DB, IC)$ Δ-minimally entails φ, written $\mathcal{T}(DB, IC) \models_\Delta \varphi$, iff every Δ-minimal model $\mathcal{M}$ of $\mathcal{T}(DB, IC)$, such that $DB_{\mathcal{M}}$ is finite, satisfies φ, *i.e.* $\mathcal{M} \models \varphi$. □

Now we characterize consistent query answers wrt the annotated theory.

Proposition 2. Let DB be a database instance, IC a set of integrity constraints and $Q(\bar{x})$ a query in FO language Σ. It holds:

$$DB \models_c Q(\bar{t}) \text{ iff } \mathcal{T}(DB, IC) \models_\Delta Q^{an}(\bar{t}). \quad \square$$

Example 6. (example 5 continued) For consistently answering the query $Q(x)$, we pose the query $Q^{an}(x)$ to the minimal models of $\mathcal{T}(DB, IC)$. The answer we obtain from *every* minimal model is $x = kafka$. □

According to this proposition, in order to consistently answer queries, we are left with the problem of evaluating minimal entailment wrt the annotated theory. In [Arenas et al. 2000a] some limited FO queries were evaluated, but no annotated queries were generated. The original query were answered using ad hoc algorithms that were extracted from theory $\mathcal{T}(DB, IC)$. No advantage was taken from a characterization of consistent answers in terms of minimal entailment from $\mathcal{T}(DB, IC)$. In the next section we will address this issue by taking the original DB instance with the ICs into a logic program that is generated taking advantage of the annotations provided by $\mathcal{T}(DB, IC)$. The query to be posed to the logic program will be built from Q^{an}.

5 Query Answering

In this section we will consider ICs of the form (1), more precisely of the form

$$\bigvee_{i=1}^{n} \neg p_i(\bar{t_i}) \vee \bigvee_{j=1}^{m} q_j(\bar{s_j}) \vee \varphi, \tag{4}$$

where, for every i and j, p_i and q_j are predicates in P, and φ is a formula containing predicates in B only.

5.1 Logic programming specification of repairs

In order to generate a first order logic program that gives an account of annotations, for each predicate

$p(\bar{x}) \in P$, we introduce a new, predicate $p(\bar{x}, \cdot)$, with an extra argument for annotations. This defines a new FO language, Σ^{ext}, for extended Σ. The repair logic program, $\Pi(DB, IC)$, for DB and IC, is written with predicates from Σ^{ext} and contains the following clauses:

1. For every atom $p(\bar{a}) \in DB$, $\Pi(DB, IC)$ contains the fact $p(\bar{a}, \mathbf{t_d}) \leftarrow$.

2. For every predicate $p \in P$, $\Pi(DB, IC)$ contains the clauses:

   $p(\bar{x}, \mathbf{t}^\star) \leftarrow p(\bar{x}, \mathbf{t_d})$

   $p(\bar{x}, \mathbf{t}^\star) \leftarrow p(\bar{x}, \mathbf{t_a})$

   $p(\bar{x}, \mathbf{f}^\star) \leftarrow p(\bar{x}, \mathbf{f_a})$,

   where $\mathbf{t}^\star, \mathbf{f}^\star$ are new, auxiliary elements in the domain of annotations.

3. For every constraint of the form (4), $\Pi(DB, IC)$ contains the clause:

   $$\bigvee_{i=1}^{n} p_i(\bar{t}_i, \mathbf{f_a}) \ \vee \ \bigvee_{j=1}^{m} q_j(\bar{s}_j, \mathbf{t_a}) \ \longleftarrow \\ \bigwedge_{i=1}^{n} p_i(\bar{t}_i, \mathbf{t}^\star) \ \wedge \ \bigwedge_{j=1}^{m} q_j(\bar{s}_j, \mathbf{f}^\star) \ \wedge \ \bar{\varphi},$$

   where $\bar{\varphi}$ represents the negation of φ.

Intuitively, the clauses in 3. say that when the IC is violated (the body), then the DB has to be repaired according to one of the alternatives shown in the head. Since there may be interactions between constraints, these single repairing steps may not be enough to restore the consistency of the DB. We have to make sure that the repairing process continues and stabilizes in a state where all the ICs hold[2]. This is the role of the clauses in 2. containing the new annotations $\mathbf{t}^\star$, that groups together those atoms annotated with $\mathbf{t_d}$ and $\mathbf{t_a}$, and $\mathbf{f}^\star$, that does the same with $\mathbf{f_d}$ and $\mathbf{f_a}$ (with the help of Definition 4 below).

The following example shows the interaction of a FD and an inclusion dependency. When atoms are deleted in order to satisfy the FD, the inclusion dependency could be violated, and in a second step it should be repaired. At that second step, the annotations $\mathbf{t}^\star$ and $\mathbf{f}^\star$, computed at the first step where the FD was repaired, will detect the violation of the inclusion dependency and perform the corresponding repairing process.

[2] In [Arenas et al. 2000b] a direct specification of database repairs by means of disjunctive logic programs with a stable model semantics was presented. Those programs contained both repair triggering rules and "stabilizing" rules.

Example 7. (example 1 cont.) We extend the schema with table $Eurbook(author, name, publYear)$, for European books. Now, DB also contains the literal $Eurbook(\ kafka, metamorph, 1919)\}$. If in addition to the ICs we had before, we include in IC the set inclusion dependency $\forall xyz\ (Eurbook(x, y, z) \rightarrow Book(x, y, z))$, we obtain the following program $\Pi(DB, IC)$:

1. $EurBook(kafka, metamorph, 1919, \mathbf{t_d}) \leftarrow$

   $Book(kafka, metamorph, 1919, \mathbf{t_d}) \leftarrow$

   $Book(kafka, metamorph, 1915, \mathbf{t_d}) \leftarrow$.

2. $Book(x, y, z, \mathbf{t}^\star) \leftarrow Book(x, y, z, \mathbf{t_d})$

   $Book(x, y, z, \mathbf{t}^\star) \leftarrow Book(x, y, z, \mathbf{t_a})$

   $Book(x, y, z, \mathbf{f}^\star) \leftarrow Book(x, y, z, \mathbf{f_a})$

   $Eurbook(x, y, z, \mathbf{t}^\star) \leftarrow Eurbook(x, y, z, \mathbf{t_d})$

   $Eurbook(x, y, z, \mathbf{t}^\star) \leftarrow Eurbook(x, y, z, \mathbf{t_a})$

   $Eurbook(x, y, z, \mathbf{f}^\star) \leftarrow Eurbook(x, y, z, \mathbf{f_a})$.

3. $Book(x, y, z, \mathbf{f_a}) \vee Book(x, y, w, \mathbf{f_a}) \leftarrow$
   $Book(x, y, z, \mathbf{t}^\star) \wedge Book(x, y, w, \mathbf{t}^\star) \wedge z \neq w$

   $Eurbook(x, y, z, \mathbf{f_a}) \vee Book(x, y, z, \mathbf{t_a}) \leftarrow$
   $Eurbook(x, y, z, \mathbf{t}^\star) \wedge Book(x, y, z, \mathbf{f}^\star)$. □

In order to have a semantics for our repair programs, we define their models. Since the negative information in a database instance is only implicitly available and we want to avoid explicitly representing it, we need to specify when negative information of the form $p(\bar{t}, \mathbf{f}^\star)$ is true of a model.

Definition 4. (a) Let I be an Herbrand interpretation for Σ^{ext} and φ a FO formula in Σ^{ext}. The definition of $\star$-satisfaction of φ by I, denoted $I \models_\star \varphi$, is as usual, except that for a ground atomic formula $p(\bar{a}, \mathbf{f}^\star)$ it holds: $I \models_\star p(\bar{a}, \mathbf{f}^\star)$ iff $p(\bar{a}, \mathbf{f}^\star) \in I$ or $p(\bar{a}, \mathbf{t_d}) \notin I$.
(b) An Herbrand interpretation $\mathcal{M}$ is a *$\star$-model* of $\Pi(DB, IC)$ if for every (ground instantiation of a) clause $(\bigvee_{i=1}^{n} a_i \leftarrow \bigwedge_{j=1}^{m} b_j) \in \Pi(DB, IC)$, $\mathcal{M} \not\models_\star \bigwedge_{j=1}^{m} b_j$ or $\mathcal{M} \models_\star \bigvee_{i=1}^{n} a_i$. □

Definition 5. (a) An atom $p(\bar{a})$ in a model $\mathcal{M}$ of a program is *plausible* if it belongs to the head of a clause in $\Pi(DB, IC)$ such that $\mathcal{M}$ $\star$-satisfies the body of the clause. A model of a program is plausible if every atom in it is plausible.
(b) A model is *coherent* if it does not contain both $p(\bar{a}, \mathbf{t_a})$ and $p(\bar{a}, \mathbf{f_a})$. □

We will be interested only in the Herbrand $\star$-models of the program that are minimal wrt set inclusion[3] and plausible and *coherent*. Notice that in a coherent model we may still find both atoms $p(\bar{a}, \mathbf{t}^\star)$ and $p(\bar{a}, \mathbf{f}^\star)$. Notice also that a plausible atom may belong to a supported disjunctive head [Lobo 1998], without the other disjuncts being forced to be false.

It is possible to prove that every minimal, plausible and coherent $\star$-model of $\Pi(DB, IC)$ is a model of $\Pi(DB, IC)$ (in the usual sense). Furthermore, it is easy to see from the definition of a $\star$-model of a program that we could keep the classical notion of satisfaction by including in the program the additional clauses $p(\bar{x}, \mathbf{f}^\star) \leftarrow \mathit{not}\ p(\bar{x}, \mathbf{t_d})$, that would include in the models all the negative information we usually keep implicit via the closed world assumption. Moreover, we would be left with a normal disjunctive program, for which a stable model semantics could be used [Gelfond et al. 1988] (see section 5.3 below).

Example 8. (example 7 cont.) The coherent plausible minimal $\star$-models of the program presented in example 7 are:

$$\begin{aligned}\mathcal{M}_1 = \{&\mathit{Eurbook}(\mathit{kafka}, \mathit{metamorph}, \mathit{1919}, \mathbf{t_d}),\\ &\mathit{Eurbook}(\mathit{kafka}, \mathit{metamorph}, \mathit{1919}, \mathbf{t}^\star),\\ &\mathit{Book}(\mathit{kafka}, \mathit{metamorph}, \mathit{1919}, \mathbf{t_d}),\\ &\mathit{Book}(\mathit{kafka}, \mathit{metamorph}, \mathit{1919}, \mathbf{t}^\star),\\ &\mathit{Book}(\mathit{kafka}, \mathit{metamorph}, \mathit{1915}, \mathbf{t_d}),\\ &\mathit{Book}(\mathit{kafka}, \mathit{metamorph}, \mathit{1915}, \mathbf{t}^\star),\\ &\mathit{Book}(\mathit{kafka}, \mathit{metamorph}, \mathit{1915}, \mathbf{f_a}),\\ &\mathit{Book}(\mathit{kafka}, \mathit{metamorph}, \mathit{1915}, \mathbf{f}^\star)\}.\end{aligned}$$

$$\begin{aligned}\mathcal{M}_2 = \{&\mathit{Eurbook}(\mathit{kafka}, \mathit{metamorph}, \mathit{1919}, \mathbf{t_d}),\\ &\mathit{Eurbook}(\mathit{kafka}, \mathit{metamorph}, \mathit{1919}, \mathbf{t}^\star),\\ &\mathit{Eurbook}(\mathit{kafka}, \mathit{metamorph}, \mathit{1919}, \mathbf{f_a}),\\ &\mathit{Eurbook}(\mathit{kafka}, \mathit{metamorph}, \mathit{1919}, \mathbf{f}^\star),\\ &\mathit{Book}(\mathit{kafka}, \mathit{metamorph}, \mathit{1919}, \mathbf{t_d}),\\ &\mathit{Book}(\mathit{kafka}, \mathit{metamorph}, \mathit{1919}, \mathbf{t}^\star),\\ &\mathit{Book}(\mathit{kafka}, \mathit{metamorph}, \mathit{1919}, \mathbf{f_a}),\\ &\mathit{Book}(\mathit{kafka}, \mathit{metamorph}, \mathit{1919}, \mathbf{f}^\star),\\ &\mathit{Book}(\mathit{kafka}, \mathit{metamorph}, \mathit{1915}, \mathbf{t_d}),\\ &\mathit{Book}(\mathit{kafka}, \mathit{metamorph}, \mathit{1915}, \mathbf{t}^\star)\}.\end{aligned}$$ □

Notice that, in contrast to the minimal models of the annotated theory $\mathcal{T}(DB, IC)$, the $\star$-models of the program will include the database contents with its original annotations ($\mathbf{t_d}$). Every time there is an atom in a model annotated with $\mathbf{t_d}$ or $\mathbf{t_a}$, it will appear annotated with $\mathbf{t}^\star$. From these models we should be able to "read" database repairs. Every $\star$-model of the logic program has to be interpreted.

[3] To distinguish them from the Δ-minimal model of the annotated theory.

Definition 6. Given a coherent plausible $\star$-model $\mathcal{M}$ of $\Pi(DB, IC)$, its *interpretation*, $i(\mathcal{M})$, is a new Herbrand interpretation obtained from $\mathcal{M}$ as follows:

1. If $p(\bar{a}, \mathbf{f_a})$ belongs to $\mathcal{M}$, then $p(\bar{a}, \mathbf{f}^{\star\star})$ belongs to $i(\mathcal{M})$.
2. If neither $p(\bar{a}, \mathbf{t_d})$ nor $p(\bar{a}, \mathbf{t_a})$ belongs to $\mathcal{M}$, then $p(\bar{a}, \mathbf{f}^{\star\star})$ belongs to $i(\mathcal{M})$.
3. If $p(\bar{a}, \mathbf{t_d})$ belongs to $\mathcal{M}$ and $p(\bar{a}, \mathbf{f_a})$ does not belong to $\mathcal{M}$, then $p(\bar{a}, \mathbf{t}^{\star\star})$ belongs to $i(\mathcal{M})$.
4. If $p(\bar{a}, \mathbf{t_a})$ belongs to $\mathcal{M}$, then $p(\bar{a}, \mathbf{t}^{\star\star})$ belongs to $i(\mathcal{M})$. □

Notice that the interpreted models contain two new annotations, $\mathbf{t}^{\star\star}, \mathbf{f}^{\star\star}$, in the last arguments. The first one groups together those atoms annotated either with $\mathbf{t_a}$ or with $\mathbf{t_d}$ but not $\mathbf{f_a}$. Intuitively, the latter correspond to those annotated with $\mathbf{t}$ in the models of $\mathcal{T}(DB, IC)$. A similar role plays the other new annotation wrt the "false" annotations. These new annotations will simplify the expression of the queries to be posed to the program (see section 5.2). Without them, instead of simply asking $p(\bar{x}, \mathbf{t}^{\star\star})$ (for the tuples in a repair), we would have to ask for $p(\bar{x}, \mathbf{t_a}) \vee (p(\bar{x}, \mathbf{t_d}) \wedge \neg p(\bar{x}, \mathbf{f_a}))$.

Example 9. (example 8 cont.) The interpreted models are:

$$\begin{aligned}i(\mathcal{M}_1) = \{&\mathit{Eurbook}(\mathit{kafka}, \mathit{metamorph}, \mathit{1919}, \mathbf{t}^{\star\star}),\\ &\mathit{Book}(\mathit{kafka}, \mathit{metamorph}, \mathit{1919}, \mathbf{t}^{\star\star}),\\ &\mathit{Book}(\mathit{kafka}, \mathit{metamorph}, \mathit{1915}, \mathbf{f}^{\star\star})\}\end{aligned}$$

$$\begin{aligned}i(\mathcal{M}_2) = \{&\mathit{Eurbook}(\mathit{kafka}, \mathit{metamorph}, \mathit{1919}, \mathbf{f}^{\star\star}),\\ &\mathit{Book}(\mathit{kafka}, \mathit{metamorph}, \mathit{1919}, \mathbf{f}^{\star\star}),\\ &\mathit{Book}(\mathit{kafka}, \mathit{metamorph}, \mathit{1915}, \mathbf{t}^{\star\star})\}.\end{aligned}$$ □

The interpreted models could be easily obtained by adding new rules to the program $\Pi(DB, IC)$. This will be shown in section 5.2. From an interpreted model of the program we can obtain a database instance:

Definition 7.

$$DB^{\Pi}_{i(\mathcal{M})} = \{p(\bar{a}) \mid i(\mathcal{M}) \models p(\bar{a}, \mathbf{t}^{\star\star})\}.$$ □

Example 10. (example 9 cont.) The following database instances obtained from definition 7 are the repairs of DB:

$DB^{\Pi}_{i(\mathcal{M}_1)} = \{Eurbook(kafka, metamorph, 1919), Book(kafka, metamorph, 1919)\}$,

$DB^{\Pi}_{i(\mathcal{M}_2)} = \{Book(kafka, metamorph, 1915)\}$. □

Theorem 1. If $\mathcal{M}$ is a coherent minimal plausible $\star$-model of $\Pi(DB, IC)$, and $DB^{\Pi}_{i(\mathcal{M})}$ is finite, then $DB^{\Pi}_{i(\mathcal{M})}$ is a repair of DB with respect to IC. Furthermore, the repairs obtained in this way are all the repairs of DB. □

5.2 The query program

Given a first order query Q, we want the consistent answers from DB. In consequence, we need those atoms that are simultaneously true in every interpreted coherent minimal plausible $\star$-model of the program $\Pi(DB, IC)$. They are obtained through the query $Q^{\star\star}$, obtained from Q by replacing, for $p \in P$, every positive literal $p(\bar{s})$ by $p(\bar{s}, \mathbf{t}^{\star\star})$ and every negative literal $\neg p(\bar{s})$ by $p(\bar{s}, \mathbf{f}^{\star\star})$. This query corresponds to the annotated version Q^{an} of Q (see section 4). Now $Q^{\star\star}$ can be transformed into a query program $\Pi(Q^{\star\star})$ by a standard transformation [Lloyd 1987, Abiteboul et al. 1995]. This query program will be run in combination with a program, Π^{int}, that specifies the interpreted models. This program can be obtained extending $\Pi(DB, IC)$ with the following rules:

$p(\bar{x}, \mathbf{t}^{\star\star}) \leftarrow p(\bar{x}, \mathbf{t_a})$,
$p(\bar{x}, \mathbf{t}^{\star\star}) \leftarrow p(\bar{x}, \mathbf{t_d}),\ not\ p(\bar{x}, \mathbf{f_a})$,
$p(\bar{x}, \mathbf{f}^{\star\star}) \leftarrow p(\bar{x}, \mathbf{f_a})$,
$p(\bar{x}, \mathbf{f}^{\star\star}) \leftarrow not\ p(\bar{x}, \mathbf{t_d}),\ not\ p(\bar{x}, \mathbf{t_a})$.

The extended program, that now contains weak negation, is basically stratified due to the different ground annotations in the predicates. Thus, its models can be computed by extending the models of the original program in a uniform manner.

Example 11. (example 7 cont.) The program Π^{int} is obtained adding to the program $\Pi(DB, IC)$ of example 7 the following clauses:

$Eurbook(x, y, z, \mathbf{t}^{\star\star}) \leftarrow Eurbook(x, y, z, \mathbf{t_a})$

$Eurbook(x, y, z, \mathbf{f}^{\star\star}) \leftarrow Eurbook(x, y, z, \mathbf{f_a})$

$Eurbook(x, y, z, \mathbf{t}^{\star\star}) \leftarrow Eurbook(x, y, z, \mathbf{t_d}),\ not\ Eurbook(x, y, z, \mathbf{f_a})$

$Eurbook(x, y, z, \mathbf{f}^{\star\star}) \leftarrow not\ Eurbook(x, y, z, \mathbf{t_d}),\ not\ Eurbook(x, y, z, \mathbf{t_a})$

$Book(x, y, z, \mathbf{t}^{\star\star}) \leftarrow Book(x, y, z, \mathbf{t_a})$

$Book(x, y, z, \mathbf{f}^{\star\star}) \leftarrow Book(x, y, z, \mathbf{f_a})$

$Book(x, y, z, \mathbf{t}^{\star\star}) \leftarrow Book(x, y, z, \mathbf{t_d}),\ not\ Book(x, y, z, \mathbf{f_a})$

$Book(x, y, z, \mathbf{f}^{\star\star}) \leftarrow not\ Book(x, y, z, \mathbf{t_d}),\ not\ Book(x, y, z, \mathbf{t_a})$.

For the query $Q(y) : \exists z Book(kafka, y, z)$, we generate $Q^{\star\star}(y) : \exists z Book(kafka, y, z, \mathbf{t}^{\star\star})$, that is transformed into the query program $\Pi(Q^{\star\star})$: $Answer(y) \leftarrow Book(kafka, y, z, \mathbf{t}^{\star\star})$.

The coherent minimal plausible $\star$-models of $\Pi^{int} \cup \Pi(Q^{\star})$ are $\mathfrak{M}_1 = \mathcal{M}_1 \cup i(\mathcal{M}_1) \cup \{Answer(metamorph)\}$ and $\mathfrak{M}_2 = \mathcal{M}_2 \cup i(\mathcal{M}_2) \cup \{Answer(metamorph)\}$, where $\mathcal{M}_1, \mathcal{M}_2$ and $i(\mathcal{M}_1), i(\mathcal{M}_2)$ are given in examples 8 and 9, resp. We can see that $y = metamorph$ is a consistent answer. □

5.3 Computing from the program

The repair programs $\Pi(DB, IC)$ introduced in section 5.1 are based on a non classical notion of satisfaction (definition 4). In order to compute from the program using a stable model semantics for disjunctive programs, we build a new program $\Pi^{\neg}(DB, IC)$, obtained from the original one by adding the clause $p(\bar{x}, \mathbf{f}^{\star}) \leftarrow not\ p(\bar{x}, \mathbf{t_d})$ that gives an account of the closed world assumption. It holds:

Proposition 3. If $\mathcal{M}$ is a coherent stable model of $\Pi^{\neg}(DB, IC)$, then $DB^{\Pi}_{\mathcal{M}}$ is a repair of DB with respect to IC; and every repair can be obtained in this way. □

In consequence, the database repairs can be specified using disjunctive logic programs with a stable model semantics [Gelfond et al. 1988, Gelfond et al. 1991], and an implementation of this semantics, like *DLV* [Eiter et al. 2000], can be used to compute both repairs and consistent answers. Notice that *DLV* implements denial constraints [Buccafurri et al. 2000], which can be used to keep the coherent stable models only, by pruning those models that do not satisfy $\leftarrow p(\bar{x}, \mathbf{t_a}), p(\bar{x}, \mathbf{f_a})$.

Example 12. Consider the database instance $\{p(a)\}$ that is inconsistent wrt the set inclusion dependency $\forall x\ (p(x) \rightarrow q(x))$. The program $\Pi^{\neg}(DB, IC)$ contains the following clauses:

1. The following rules do not depend on ICs

   $p(x, \mathbf{f}^{\star}) \leftarrow p(x, \mathbf{f_a})$

   $p(x, \mathbf{t}^{\star}) \leftarrow p(x, \mathbf{t_a})$

$p(x, \mathbf{t}^\star) \leftarrow p(x, \mathbf{t_d})$
$q(x, \mathbf{f}^\star) \leftarrow q(x, \mathbf{f_a})$
$q(x, \mathbf{t}^\star) \leftarrow q(x, \mathbf{t_a})$
$q(x, \mathbf{t}^\star) \leftarrow q(x, \mathbf{t_d})$

2. A single rule capturing the IC
$p(x, \mathbf{f_a}) \vee q(x, \mathbf{t_a}) \leftarrow p(x, \mathbf{t}^\star), q(x, \mathbf{f}^\star).$

3. Database contents
$p(a, \mathbf{t_d}) \leftarrow$

4. The new rules for the closed world assumption
$p(x, \mathbf{f}^\star) \leftarrow \ not\ p(x, \mathbf{t_d})$
$q(x, \mathbf{f}^\star) \leftarrow \ not\ q(x, \mathbf{t_d}).$

5. Denial constraints for coherence
$\leftarrow p(\bar{x}, \mathbf{t_a}), p(\bar{x}, \mathbf{f_a})$
$\leftarrow q(\bar{x}, \mathbf{t_a}), q(\bar{x}, \mathbf{f_a}).$

6. Rules for interpreting the models
$p(x, \mathbf{t}^{\star\star}) \leftarrow p(x, \mathbf{t_a})$
$p(x, \mathbf{t}^{\star\star}) \leftarrow p(x, \mathbf{t_d}),\ not\ p(x, \mathbf{f_a})$
$p(x, \mathbf{f}^{\star\star}) \leftarrow p(x, \mathbf{f_a})$
$p(x, \mathbf{f}^{\star\star}) \leftarrow \ not\ p(x, \mathbf{t_d}),\ not\ p(x, \mathbf{t_a})$
$q(x, \mathbf{t}^{\star\star}) \leftarrow q(x, \mathbf{t_a})$
$q(x, \mathbf{t}^{\star\star}) \leftarrow q(x, \mathbf{t_d}),\ not\ q(x, \mathbf{f_a})$
$q(x, \mathbf{f}^{\star\star}) \leftarrow q(x, \mathbf{f_a})$
$q(x, \mathbf{f}^{\star\star}) \leftarrow \ not\ q(x, \mathbf{t_d}),\ not\ q(x, \mathbf{t_a}).$ □

It can be seen that the programs with annotations we have obtained are very simple in terms of their dependency on the ICs.

As mentioned before, consistent answers can be obtained "running" the query program introduced in section 5.2 in combination with the repair program $\Pi^{\neg}(DB, IC)$, under the skeptical stable model semantics, that sanctions as true what is true of all stable models.

6 Conclusions

Extending work presented in [Arenas et al. 2000a], we have shown how to annotate referential ICs in order to obtain a specification in annotated predicate logic of the class of repairs of a relational database. The correspondence between the minimal models of the annotated first order specification and the database repairs is established. Ongoing work considers the extension of the annotated embedding methodology to the class of all ICs found in DB praxis [Abiteboul et al. 1995, chap. 10].

We formulated the problem of consistent query answering as a problem of non monotonic entailment of a modified query from the annotated theory.

We have presented a general treatment of consistent query answering for first order queries and universal ICs. This is done by means of disjunctive logic programs with a stable model semantics, where annotations are now arguments, that specify the database repairs in the case of universal ICs. In consequence, consistent query answers can be obtained by "running" the program. Ongoing work considers the extension of the logic programs to include referential (and more general) ICs.

In [Greco et al. 2001], a general methodology for specifying database repairs wrt universal ICs is presented. There, disjunctive logic programs with stable model semantics are used. They also consider the problem of specifying preferences between possible repairs. Independently, [Arenas et al. 2000b] also presents a specification of database repairs by means of disjunctive logic programs with a stable model semantics. The programs capture the repairs for binary universal ICs. The programs presented here also work for the whole class of universal ICs, but they are much simpler than those presented in [Greco et al. 2001, Arenas et al. 2000b]; this is due to the simplicity of the stabilizing rules, that take full advantage of the relationship between the annotations. The simplicity is expressed in a much smaller number of rules and the syntactic properties of the programs.

In [Blair et al. 1989, Kifer et al. 1992b, Leach et al. 1996] paraconsistent and annotated logic programs are introduced. In particular, in [Subrahmanian 1994] those programs are used to integrate databases, a problem closely related to inconsistency handling. It is not clear how to use those definitive non disjunctive programs to capture database repairs. Furthermore, notice that the programs presented in this paper have a completely classical semantics.

In [Gaasterland 1994], annotated logic (programs) were used to specify and obtain answers matching user needs and preferences. Deeper relationships to our work deserve to be explored.

Acknowledgments: Work supported by DIPUC, FONDECYT Grant 1000593, and Carleton University Start-Up Grant 9364-01. We are grateful to Alberto Mendelzon and Marcelo Arenas for stimulating and useful conversations, and comments.

References

[Abiteboul et al. 1995] Abiteboul, S.; Hull, R. and Vianu, V. "Foundations of Databases". Addison-Wesley, 1995.

[Arenas et al. 1999] Arenas, M.; Bertossi, L. and Chomicki, J. "Consistent Query Answers in Inconsistent Databases". In *Proc. ACM Symposium on Principles of Database Systems (ACM PODS'99), Philadelphia)*, pages 68-79, 1999.

[Arenas et al. 2000a] Arenas, M.; Bertossi, L. and Kifer, M. "Applications of Annotated Predicate Calculus to Querying Inconsistent Databases". In *'Computational Logic - CL2000' Stream: 6th International Conference on Rules and Objects in Databases (DOOD'2000)*. Springer Lecture Notes in Artificial Intelligence 1861, pages 926-941.

[Arenas et al. 2000b] Arenas, M.; Bertossi, L. and Chomicki, J. "Specifying and Querying Database Repairs using Logic Programs with Exceptions". In *Flexible Query Answering Systems. Recent Developments*, H.L. Larsen, J. Kacprzyk, S. Zadrozny, H. Christiansen (eds.), Springer, 2000, pp. 27–41.

[Blair et al. 1989] Blair, H.A. and Subrahmanian, V.S. "Paraconsistent Logic Programming". *Theoretical Computer Science,* 68:135-154, 1989.

[Buccafurri et al. 2000] Buccafurri, F.; Leone, N. and Rullo, P. "Enhancing Disjunctive Datalog by Constraints". *IEEE Transactions on Knowledge and Data Engineering*, 2000, 12(5) : 845-860.

[Eiter et al. 2000] Eiter, T.; Faber, W.; Leone, N. and Pfeifer, G. "Declarative Problem-Solving in DLV". In *Logic-Based Artificial Intelligence*, J. Minker (ed.), Kluwer, 2000, pp. 79–103.

[Gaasterland 1994] Gaasterland, T. and Lobo, J. "Qualified Answers That Reflect User Needs and Preferences". In Proc. 20th International Conference of Very Large Databases (VLDB'94). Morgan Kaufmann Publishers, 1994, pages 309–320.

[Gelfond et al. 1988] Gelfond, M. and Lifschitz, V. "The Stable Model Semantics for Logic Programming". In *Logic Programming, Proceedings of the Fifth International Conference and Symposium*, R. A. Kowalski and K. A. Bowen (eds.), MIT Press 1988, pp. 1070–1080.

[Gelfond et al. 1991] Gelfond, M. and Lifschitz, V. "Classical Negation in Logic Programs and Disjunctive Databases". *New Generation Computing*, 1991, 9:365–385.

[Greco et al. 2001] Greco, G.; Greco, S. and Zumpano, E. "A Logic Programming Approach to the Integration, Repairing and Querying of Inconsistent Databases". In Proc. 17th International Conference on Logic Programming, ICLP'01, Ph. Codognet (ed.), LNCS 2237, Springer, 2001, pp. 348–364.

[Kifer et al. 1992a] Kifer, M. and Lozinskii, E.L. "A Logic for Reasoning with Inconsistency". *Journal of Automated reasoning*, 9(2):179-215, November 1992.

[Kifer et al. 1992b] Kifer, M. and Subrahmanian, V.S. "Theory of Generalized Annotated Logic Programming and its Applications". *Journal of Logic Programming*, 12(4):335-368, April 1992.

[Leach et al. 1996] Leach, S.M. and Lu, J.J. "Query Processing in Annotated Logic Programming: Theory and Implementation". *Journal of Intelligent Information Systems*, 6, January 1996, pp. 33–58.

[Lloyd 1987] Lloyd, J.W. "Foundations of Logic Programming". Springer Verlag, 1987.

[Lobo 1998] Lobo, J.; Minker, J. and Rajasekar, A. "Semantics for Disjunctive and Normal Disjunctive Logic Programs". In *Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 5*, D. Gabbay et al. (eds.). Oxford University Press, 1998.

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