[P’agina intencionalmente en blanco]

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

APPLICATIONS OF ANNOTATED PREDICATE CALCULUS AND LOGIC PROGRAMS TO QUERYING INCONSISTENT DATABASES

PABLO BARCELÓ BAEZA

Tesis para optar al Grado de
Magister en Ciencias de la Ingeniería

Profesor Supervisor:
LEOPOLDO BERTOSSI D.

Santiago de Chile, 2002

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
Departamento de Ciencia de la Computación

APPLICATIONS OF ANNOTATED PREDICATE CALCULUS AND LOGIC PROGRAMS TO QUERYING INCONSISTENT DATABASES

PABLO BARCELÓ BAEZA

Tesis para optar al Grado de
Magister en Ciencias de la Ingeniería

Profesor Supervisor:
LEOPOLDO BERTOSSI D.

Santiago de Chile, 2002

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
Departamento de Ciencia de la Computación

APPLICATIONS OF ANNOTATED PREDICATE CALCULUS AND LOGIC PROGRAMS TO QUERYING INCONSISTENT DATABASES

PABLO BARCELÓ BAEZA

Tesis presentada a la Comisión integrada por los profesores:

LEOPOLDO BERTOSSI D.

ALVARO CAMPOS U.

RENATO LEWIN R.

CLAUDIO GUTIERREZ G.

ANDRÉS GUESALAGA M.

para completar las exigencias del grado de
Magister en Ciencias de la Ingeniería

Santiago de Chile, 2002

Para Pedro y Mateo, y para mis abuelos.

ACKNOWLEDGMENTS

Quiero agradecer a mi profesor Leopoldo Bertossi y a su familia en Canadá, por hacerme sentir tan bien durante los dos meses que pasé con ellos en Ottawa. Realmente disfruté el vivir esos momentos y seguramente tal estadía influyó de manera decisiva en mi vida. Gracias.

CONTENTS

Page

LIST OF FIGURES

SUMMARY

A database instance that is not a Herbrand model of its integrity constraints is said to be inconsistent. Anyway, most of its tuples continue being useful for us. We say that an answer to a query is a consistent answer if it holds in every repair of the original instance, *i.e.* in every instance of the same schema that satisfies the integrity constraints and that differs from DB by a minimal (under set inclusion) set of inserted or deleted tuples.

Since the database instance DB and the set of integrity constraints IC are mutually inconsistent, and given that classical logic trivializes logical inference in the presence of inconsistencies, we choose Annotated Predicate Calculus (APC) to specify the repairs of the original instance. The reason why APC is useful in analyzing classically inconsistent logical theories is because classical theories can be embedded in APC in various ways. The most useful types of embeddings are those where theories that are inconsistent in classical logic become consistent in APC. It then becomes possible to reason about the embedded theories and gain insight on them. It was shown in previous work there is a one to one correspondence between some distinguished models of an annotated embedding in APC of the database instance and the constraints, and the repairs of the database for universal ICs.

This work extends the previous one to handle referential integrity constraints. Then, it deals with the problem of consistent query answering as a problem of non-monotonic entailment from the annotated theory. Finally, first order disjunctive programs are introduced, motivated by the annotated theory, in such a way that certain distinguished stable models correspond to the repairs of the original instance. An implementation of these programs in DLV is given, and a extension of the programs to consider referential integrity constraints is sketched.

I. 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 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, a ground tuple $\bar{t}$ to a first order query $Q(\bar{x})$ is a *consistent* answer 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 it can handle. In (Arenas et al. 2000b), a more general methodology based on logic programs with a 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 to deal with *all* the repairs of a database is required. 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, a logic that does not collapse in the presence of contradictions is required. A "paraconsistent logic", for which an inconsistent set of premises could still have a model, is a natural candidate. In (Arenas et al. 2000a), a new declarative semantic framework was presented 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 Annotated Predicate Calculus (APC) (Kifer et al. 1992a) with an appropriate non classical truth-values lattice $Latt$.

In (Arenas et al. 2000a) it was shown that there is a one to one correspondence between some minimal models of the annotated theory and the repairs of the inconsistent database for universal ICs. 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 work is structured as follows:

- In chapter 2 the basic notions are presented, corresponding mainly to those in (Arenas et al. 2000a). In particular, the correspondence between repairs of the original instance and minimal models of an annotated theory are stated.
- In chapter 3 the methodology presented in (Arenas et al. 2000a) is extended in order to consider referential integrity constraints. The correspondence between repairs of the original database instance and the minimal models of the annotated theory is established. Then, it is shown that a more general kind of constraint, including both universal and referential integrity constraint, can be considered.
- In chapter 4, it is shown 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.
- In chapter 5, on the basis of the generated annotated theory, disjunctive first order logic programs with annotation arguments to specify the database repairs are presented. It is also shown the correspondence between some well-identified models of the program and the repairs of the original database instance. A way to obtain consistent answers from these models is also stated. Finally, an

implementation of these programs in *DLV* is shown and an extension of the programs to consider referential integrity constraints.

- In chapter 7 some related work is mentioned. In chapter 8 the conclusions of the work are presented, together with some possible future work.

II. APPLICATIONS OF ANNOTATED PREDICATE CALCULUS TO QUERYING INCONSISTENT DATABASES

The aim of this section is to present the main definitions and underlying semantics that will be used in the rest of the work. As our framework leads to the declarative and query semantics presented in (Arenas et al. 2000a), what is stated in this section corresponds exactly to what was stated there. It includes the fundamental results about the biunivocal correspondence between minimal models of the new APC theory (obtained from the original database) and the repairs of the original database. Notice every definition, construction and result stated in this chapter is taken from (Arenas et al. 2000a).

2.1 Preliminaries

The signature $\Sigma = (D,\ P \cup B,\ d)$ for the first order language consists of a fixed domain $D = \{c_1, c_2, ...\}$, a fixed database schema $P = \{p_1, \ldots, p_n\}$, denoting the database relations, a fixed set of built-in predicates $B = \{e_1, \ldots, e_m\}$ and a function $d : P \cup B \rightarrow \mathbb{N}$ specifying the cardinality of the predicates.

Definition 1. (Databases and Constraints) A *database instance DB* is a finite collection of facts, *i.e.*, of statements of the form $p(c_1, ..., c_n)$, where p is a predicate in P and $c_1, ..., c_n$ are constants in D.

An *integrity constraint* is a clause of the form

$$p_1(\bar{T}_1) \vee \cdots \vee p_n(\bar{T}_n) \vee \neg q_1(\bar{S}_1) \vee \cdots \vee \neg q_m(\bar{S}_m) \tag{2.1}$$

where each p_i $(1 \leq i \leq n)$ and q_j $(q \leq j \leq m)$ is a predicate in $P \cup B$ and $\bar{T}_1, ..., \bar{T}_n, \bar{S}_1, ..., \bar{S}_m$ are tuples (of appropriate arities) of constants or variables. As usual, it is assumed that all variables in a clause are universally quantified, so the quantifiers are omitted. □

When considered in isolation, that DB and IC are consistent theories will be assumed. Nevertheless, that might not be the case with $DB \cup IC$. The notion of sentence satisfaction in a database used in this work is the usual one. It will be denoted by $\models_\Sigma$, to differentiate it from other entailment relations used here.

Definition 2. (Sentence Satisfaction) It is used $\models_\Sigma$ to denote the usual notion of formula satisfaction in a database. In other words,

- $DB \models_\Sigma p(\bar{c})$, where $p \in P$, iff $p(\bar{c}) \in DB$;
- $DB \models_\Sigma q(\bar{c})$, where $q \in B$, iff $q(\bar{c})$ is true;
- $DB \models_\Sigma \neg\varphi$ iff it is not true that $DB \models_\Sigma \varphi$;
- $DB \models_\Sigma \phi \wedge \psi$ iff $DB \models_\Sigma \phi$ and $DB \models_\Sigma \psi$;
- $DB \models_\Sigma (\forall X)\phi(X)$ iff for all $d \in D$, $DB \models_\Sigma \phi(d)$;

and so on. Notice that the domain is fixed, and it is involved in the above definition. □

Definition 3. A database instance DB satisfies a set of integrity constraints IC iff it is a model for every $\varphi \in IC$. If DB does not satisfy IC, it is said that DB is *inconsistent* with IC. □

Next the relevant definitions from (Arenas et al. 1999) are recalled.

Definition 4. Given two database instances DB_1 and DB_2, the *distance* $\Delta(DB_1, DB_2)$ between them is their symmetric difference: $\Delta(DB_1, DB_2) = (DB_1 - DB_2) \cup (DB_2 - DB_1)$. □

Definition 5. The partial order $\leq_{DB}$ between instances under the same schema is defined as:

$$DB_1 \leq_{DB} DB_2 \text{ iff } \Delta(DB, DB_1) \subseteq \Delta(DB, DB_2). \qquad \square$$

That is, $\leq_{DB}$ determines the "closeness" to DB. The notion of closeness forms the basis for the concept of a repair of an inconsistent database.

Definition 6. Given database instances DB and DB', it is said that DB' is a *repair* of DB with respect to a set of integrity constraints IC iff DB' satisfies IC and DB' is $\leq_{DB}$-*minimal* in the class of database instances that satisfy IC. $\square$

Clearly if DB is consistent with respect to IC, then DB is the only repair of itself.

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
- $DB_2 = \{Book(kafka, metamorph, 1919)\}$.

$\square$

Definition 7. (Consistent Answers) Let DB be a database instance, IC be set of integrity constraints and $Q(\bar{x})$ be a query. It is said that a tuple of constants $\bar{t}$ is a *consistent answer* to the query, 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, then *true* (respectively, *false*) is a *consistent answer* to Q, denoted $DB \models_c Q$, if $DB' \models_\Sigma Q$ (respectively, $DB' \not\models_\Sigma Q$) for every repair DB' of DB. □

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

2.2 Annotated Predicate Calculus

Annotated predicate calculus (APC) (Kifer et al. 1992a) is a generalization of annotated logic programs introduced in (Blair et al. 1989). It was introduced in order to study the problem of "causes of inconsistency" in classical logical theories, which is closely related to the problem of consistent query answers being addressed in the present work.

The syntax and the semantics of APC is based on classical logic, except that the classical atomic formulas are annotated with values drawn from a *belief semilattice* (abbr. BSL) — an upper semilattice[1] with the following properties:

a. BSL contains at least the following four distinguished elements: t (true), f (false), $\top$ (contradiction), and $\bot$ (unknown);

b. For every $s \in BSL$, $\bot \leq s \leq \top$ ($\leq$ is the semilattice ordering);

c. $lub(\mathrm{t}, \mathrm{f}) = \top$, where lub denotes the least upper bound.

As usual in the lattice theory, lub imposes a partial order on BSL: $a \leq b$ iff $b = lub(a, b)$ and $a < b$ iff $a \leq b$ and a is different from b. Two typical examples of BSL (which happen to be complete lattices) are shown in Figure 2.1. In both of them, the lattice elements are ordered upwards. The specific BSL used in this paper is introduced later, in Figure 2.2.

[1]That is, the least upper bound, $lub(a, b)$, is defined for every pair of elements $a, b \in BSL$.

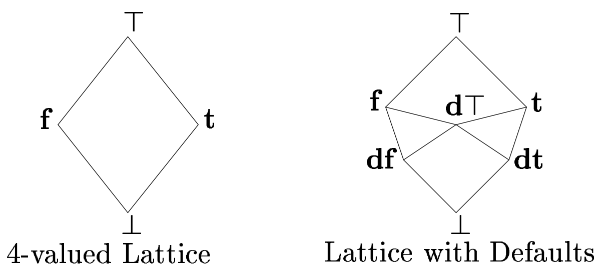

Figure 2.1 Typical Belief Semilattices

Thus, the only syntactic difference between APC and classical predicate logic is that the atomic formulas of APC are constructed from the classical atomic formulas by attaching annotation suffixes. For instance, if s, t, $\top$ are elements of the belief semilattice, then $p(X) : s$, $q : \top$, and $r(X, Y, Z) : t$ all are atomic formulas in APC.

Only Herbrand semantics of APC is defined (this is all that is needed here), and it is also assumed that the language is free of function symbols (aince only relational databases are studied in this work). The *Herbrand universe* is D, the set of all domain constants, and the *Herbrand base*, HB, is the set of all ground (*i.e.*, variable-free) atomic formulas of APC.

A *Herbrand interpretation* is any downward-closed subset of HB, where a set $I \subseteq HB$ is said to be *downward-closed* iff $p : s \in I$ implies that $p : s' \in I$ for every $s' \in BSL$ such that $s' \leq s$. Formula satisfaction can then be defined as follows, where ν is a variable assignment that gives a value in D to every variable:

- $I \models_v p : s$, where $s \in BSL$ and p is a classical atomic formula, if and only if $p : s \in I$.
- $I \models_v \phi \wedge \psi$ if and only if $I \models_v \phi$ and $I \models_v \psi$;
- $I \models_v \neg\psi$ if and only if not $I \models_v \psi$;
- $I \models_v (\forall X)\psi(X)$ if and only if $I \models_u \psi$, for *every* u that may differ from v only in its X-value.

It is thus easy to see that the definition of $\models$ looks very much classical. The only difference (which happens to have significant implications) is the syntax of atomic formulas and the requirement that Herbrand interpretations must be downward-closed. The implication $a \leftarrow b$ is also defined classically, as $a \vee \neg b$.

It turns out that whether or not APC has a complete proof theory depends on which semilattice is used. It is shown in (Kifer et al. 1992a) that for a very large and natural class of semilattices (which includes all finite semilattices), APC has a sound and complete proof theory.

The reason why APC is useful in analyzing inconsistent logical theories is because classical theories can be embedded in APC in various ways. The most useful types of embeddings are those where theories that are inconsistent in classical logic become consistent in APC. It then becomes possible to reason about the embedded theories and gain insight into the original inconsistent theory.

The two embeddings defined in (Kifer et al. 1992a) are called *epistemic* and *ontological*. Under the epistemic embedding, a (classically inconsistent) set of formulas such as $S = \{p(1),\ \neg p(1),\ q(2)\}$ is embedded in APC as $S^e\{p(1) : t,\ p(1) : f,\ q(2) : t\}$ and under the ontological embedding it is embedded as $S^o = \{p(1) : t,\ \neg p(1) : t,\ q(2) : t\}$.[2] In the second case, the embedded theory is still inconsistent in APC, but in the first case it does have a model: the downward closure of $\{p(1) : \top,\ q(2) : t\}$. In this model, $p(1)$ is annotated with $\top$, which signifies that its truth value is "inconsistent." In contrast, the truth value of $q(2)$ is t. More precisely, while both $q(2)$ and $\neg q(2)$ follow from S in classical logic, because S is inconsistent, only $q(2) : t$ (but not $q(2) : f$!) is implied by S^e. Thus, $q(2)$ can be seen as a consistent answer to the query $? - q(X)$ with respect to the inconsistent database S.

In (Kifer et al. 1992a), epistemic embedding has been shown to be a suitable tool for analyzing inconsistent classical theories. However, this embedding does not adequately capture the inherent lack of symmetry present in the actual setting, where inconsistency arises due to the incompatibility of two distinct sets of formulas (the database and the constraints) and only one of these sets (the database) is allowed to change to restore consistency. To deal with this problem, a new type of

[2] $\neg p : v$ is to be always read as $\neg(p : v)$.

embedding into APC is developed. It uses a 10-valued lattice depicted in Figure 2.2, and is akin to the epistemic embedding of (Kifer et al. 1992a), but it also has certain features of the ontological embedding.

The above simple examples illustrate one important property of APC: a set of formulas, S, might be *ontologically consistent* in the sense that it might have a model, but it might be *epistemically inconsistent* (abbr. *e-inconsistent*) in the sense that $S \models p : \top$ for some p, *i.e.*, S contains at least one inconsistent fact. Moreover, S can be e-consistent (*i.e.*, it might not imply $p : \top$ for any p), but each of its models in APC might contain an inconsistent fact nonetheless (this fact must then be different in each model, if S is e-consistent).

It was shown in (Kifer et al. 1992a) that ordering models of APC theories according to the amount of inconsistency they contain can be useful in the study of the problem of recovering from inconsistency. To illustrate this order, consider $S = \{p : t,\ p : f \vee q : t,\ p : f \vee q : f\}$ and some of its models:

- $\mathcal{M}_1$, where $p : \top$ and $q : \top$ are true;
- $\mathcal{M}_2$, where $p : \top$ and $q : \bot$ are true;
- $\mathcal{M}_3$, where $p :\ t$ and $q : \top$ are true.

Among these models, both $\mathcal{M}_2$ and $\mathcal{M}_3$ contain strictly less inconsistent information than $\mathcal{M}_1$ does. In addition, $\mathcal{M}_2$ and $\mathcal{M}_3$ contain incomparable amounts of information, and they are both "minimal" with respect to the amount of inconsistent information that they have. This leads to the following definition.

Definition 8. (E-consistency Order) Given $\Delta \subseteqq BSL$, a semantic structure I_1 is *more* (or equally) *e-consistent* than I_2 with respect to Δ (denoted $I_2 \leq_\Delta I_1$) if and only if for every atom $p(t_1, \ldots, t_k)$ and $\lambda \in \Delta$, whenever $I_1 \models p(t_1, \ldots, t_k)\ : \lambda$ then also $I_2 \models p(t_1, \ldots, t_k)\ : \lambda$.

I is Δ-*minimal* in a class of semantic structures, if no semantic structure in this class is strictly more e-consistent with respect to Δ than I (i.e., for every J in the class, $I \leq_\Delta J$ implies $J \leq_\Delta I$). $\square$

2.3 Embedding Databases in APC

One way to find reliable answers to a query over an inconsistent database is to find an algorithm that implements the definition of consistent answers. While this approach has been successfully used in (Arenas et al. 1999), it is desirable to see it as part of a bigger picture, because consistent query answers were defined at the meta-level, without an independent logical justification. A more general framework might (and does, as it will be seen) help study the problem both semantically and algorithmically.

This new approach was given in (Arenas et al. 2000a), and its goal is to embed inconsistent databases into APC and study the ways to eliminate inconsistency there. A similar problem was considered in (Kifer et al. 1992a) and some key ideas from that work are going to be adapted. In particular, an embedding $\mathcal{T}$ was defined, such that the repairs of the original database are precisely the models (in the APC sense) of the embedded database. This embedding is described below.

First, a special 10-valued lattice , $\mathcal{L}^{db}$, was built, which defines the truth values appropriate for the present problem. The lattice is shown in Figure 2.2. The values $\bot$, $\top$, t and f represent undefinedness, inconsistency, truth, and falsehood, as usual. The other six truth values are explained below.

Informally, values t_c and f_c represent the truth values as they should be for the purpose of constraint satisfaction. The values t_d and f_d are the truth values as they should be according to the database DB. Finally, t_a and f_a are the advisory truth values. Advisory truth values are intended as keepers of the information that helps resolve conflicts between constraints and the database.

Notice that $lub(f_d, t_c)$ is t_a and $lub(t_d, f_c)$ is f_a. This means that, in case of a conflict between the constraints and the database, the advise is to change the truth

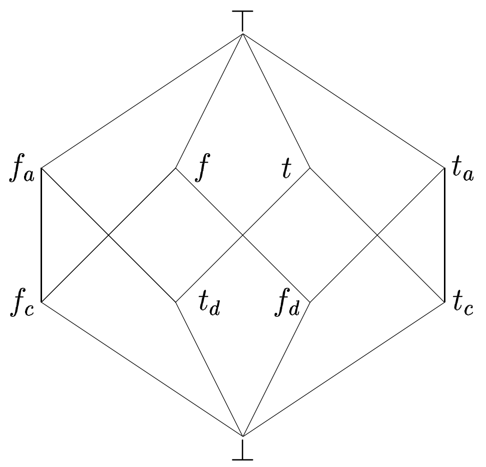

Figure 2.2 The lattice $\mathcal{L}^{db}$ with *constraints values*, *database values* and *advisory values.*

value of the corresponding fact to the one prescribed by the constraints. Intuitively, the facts that are assigned the advisory truth values are the ones that are to be removed or added to the database in order to satisfy the constraints. The gist of the actual approach is in finding an embedding of DB and IC into APC to take advantage of the above truth values.

a. **Embedding the ICs:** Given a set of integrity constraints IC, a new theory, $\mathcal{T}(\mathbf{IC})$, is defined, which contains three kinds of formulas:

1) For every constraint in IC:

$$p_1(\bar{T}_1) \vee \cdots \vee p_n(\bar{T}_n) \vee \neg q_1(\bar{S}_1) \vee \cdots \vee \neg q_m(\bar{S}_m),$$

$\mathcal{T}(\mathbf{IC})$ has the following formula:

$$p_1(\bar{T}_1) : t_c \ \vee \cdots \vee \ p_n(\bar{T}_n) : t_c \ \vee \ q_1(\bar{S}_1) : f_c \ \vee \cdots \vee \ q_m(\bar{S}_m) : f_c.$$

In other words, positive literals are embedded using the "constraint-true" truth value, t_c, and negative literals are embedded using the "constraint-false" truth value f_c.

2) For every predicate symbol $p \in P$, the following formulas are in $\mathcal{T}(\mathbf{IC})$:

$$p(\bar{x}) : t_c \ \vee \ p(\bar{x}) : f_c, \ \neg\, p(\bar{x}) : t_c \ \vee \ \neg\, p(\bar{x}) : f_c.$$

Intuitively, these say that every embedded literal must be either constraint-true or constraint-false (and not both).

b. **Embedding database facts:** $\mathcal{T}(\mathbf{DB})$, the embedding of the database facts into APC is defined as follows:

1) For every fact $p(\bar{a})$, where $p \in P$: if $p(\bar{a}) \in DB$, then $p(\bar{a}) : t_d \in \mathcal{T}(\mathbf{DB})$; if $p(\bar{a}) \notin DB$, then $p(\bar{a}) : f_d \in \mathcal{T}(\mathbf{DB})$ (maybe implicitly).

c. **Embedding built-in predicates:** $\mathcal{T}(\mathcal{B})$, the result of embedding of the built-in predicates into APC is defined as follows:

1) For every built-in fact $p(\bar{a})$, where $p \in B$, the fact $p(\bar{a}) : t$ is in $\mathcal{T}(\mathcal{B})$ iff $p(\bar{a})$ is true. Otherwise, if $p(\bar{a})$ is false then $p(\bar{a}) : f \ \in \mathcal{T}(\mathcal{B})$.

2) $\neg\, p(\bar{x}) : \top \ \in \mathcal{T}(\mathcal{B})$, for every built-in $p \in B$.

The former rule simply says that built-in facts (like 1=1) that are true in classical sense must have the truth value t and the false built-in facts (*e.g.*, 2=3) must have the truth value f. The second rule states that built-in facts cannot be both true and false. This ensures that theories for built-in predicates are embedded in 2-valued fashion: every built-in fact in $\mathcal{T}(\mathcal{B})$ is annotated with either t or f, but not both.

Finally, $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$ is defined as $\mathcal{T}(\mathbf{DB}) \cup \mathcal{T}(\mathbf{IC}) \cup \mathcal{T}(\mathcal{B})$. Now it can be stated the following properties that confirm the intuition about the intended meanings of the truth values in $\mathcal{L}^{db}$.

Lemma 1. If $\mathcal{M}$ is a model of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$, then for every predicate $p \in P$ and a fact $p(\bar{a})$, the following is true: (a) $\mathcal{M} \models \neg\, p(\bar{a}) : \top$ (b) $\mathcal{M} \models p(\bar{a}) : \mathbf{t} \vee p(\bar{a}) : \mathbf{f} \vee p(\bar{a}) : \mathbf{t_a} \vee p(\bar{a}) : \mathbf{f_a}$. □

The first part of the lemma says that even if the initial database DB is inconsistent with constraints IC, every model of the embedded theory is *epistemically* *consistent* in the sense of (Kifer et al. 1992a), *i.e.*, no fact of the form $p(\bar{a}) : \top$ is true in any such model.[3] The second part says that any fact is either true, or false, or it has an advisory value of true or false. This indicates that database repairs can be constructed out of these embeddings by converting the advisory truth values to the corresponding values t and f. This idea is explored next.

Definition 9. Given a pair of database instances DB_1 and DB_2 over the same domain, $\mathcal{M}(DB_1, DB_2)$ is the Herbrand structure $< D, I_P, I_B >$, where D is the domain of the database and I_P, I_B are the interpretations for the predicates and the built-ins, respectively. I_P is defined as follows:

- If $p(\bar{a}) \in DB_1$ and $p(\bar{a}) \in DB_2$, then $p(\bar{a})\!:\!\mathbf{t} \in I_P$.
- If $p(\bar{a}) \in DB_1$ and $p(\bar{a}) \notin DB_2$, then $p(\bar{a})\!:\!\mathbf{f_a} \in I_P$.
- If $p(\bar{a}) \notin DB_1$ and $p(\bar{a}) \notin DB_2$, then $p(\bar{a})\!:\!\mathbf{f} \in I_P$.
- If $p(\bar{a}) \notin DB_1$ and $p(\bar{a}) \in DB_2$, then $p(\bar{a})\!:\!\mathbf{t_a} \in I_P$.

The interpretation I_B is defined as expected: if q is a built-in, then $q(\bar{a})\!:\!\mathbf{t} \in I_B$ iff $q(\bar{a})$ is true in classical logic, and $q(\bar{a})\!:\!\mathbf{f} \in I_B$ iff $q(\bar{a})$ is false. $\square$

Notice that $\mathcal{M}(DB_1, DB_2)$ is not symmetric. The intent is to use these structures as the basis for construction of database repairs. In fact, when DB_1 is inconsistent and DB_2 is a repair, I_P shows how the advisory truth values are to be changed to obtain a repair.

Lemma 2. Given two database instances DB and DB', if $DB' \models_\Sigma IC$, then $\mathcal{M}(DB, DB') \models \mathcal{T}(\mathbf{DB}, \mathbf{IC})$. $\square$

The implication of this lemma is that whenever IC is consistent, then the theory $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$ is also consistent in APC. Since in this work just consistent sets

[3]Note that an APC theory *can* entail $p(\bar{a}) : \top$ and be consistent in the sense that it can have a model. However, such a model must contain $p(\bar{a}) : \top$, which makes it epistemically inconsistent.

of integrity constraints are considered, the conclusion is that $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$ is always a consistent APC theory.

Next is shown how to generate repairs out of the models of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$. Given a model $\mathcal{M}$ of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$, $DB_{\mathcal{M}}$ is defined as:

$$\{p(\bar{a}) \mid p \in P \text{ and } \mathcal{M} \models\ p(\bar{a}) : t \vee p(\bar{a}) : t_a\}. \tag{2.2}$$

Note that $DB_{\mathcal{M}}$ can be an infinite set of facts (but finite when $\mathcal{M}$ corresponds to a database instance).

Lemma 3. If $\mathcal{M}$ is a model of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$ such that $DB_{\mathcal{M}}$ is finite, then $DB_{\mathcal{M}} \models_{\Sigma} IC$. $\square$

Theorem 1. Let $\mathcal{M}$ be a model of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$. If $\mathcal{M}$ is Δ-minimal, with $\Delta = \{t_a, f_a\}$ (see Definition 8), among the models of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$ and $DB_{\mathcal{M}}$ is finite, then $DB_{\mathcal{M}}$ is a repair of DB with respect to IC. $\square$

Theorem 2. If DB' is a repair of DB with respect to the set of integrity constraints IC, then $\mathcal{M}(DB, DB')$ is Δ-minimal, with $\Delta = \{t_a, f_a\}$, among the models of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$. $\square$

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

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

2. Predicate closure axioms:

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

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

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

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

Furthermore

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

The Δ-minimal (from now on, simply *minimal* models) of $\mathcal{T}(\mathbf{DB},\mathbf{IC}) = \mathcal{T}(\mathbf{DB}) \cup \mathcal{T}(\mathbf{IC}) \cup \mathcal{T}(\mathcal{B})$ are:

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

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

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

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 the atoms are wished 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.

[4] As just atomic formulas are annotated, the non-atomic formula $\neg p(\bar{x}){:}\mathbf{s}$ is to be read as $\neg(p(\bar{x}){:}\mathbf{s})$. The parenthesis will be omitted in this work.

III. EXISTENTIAL CONSTRAINTS

The way the theory $\mathcal{T}(DB, IC)$ annotates the constraints is defined for the case the constraints are clauses. The inclusion dependency $\forall\bar{x}\big(p(\bar{x}) \rightarrow \exists y q(\bar{x}, y)\big)$ can not be expressed as a clause. This implies that a different way to embed it in APC must be found.

3.1 Annotating Referential Integrity Constraints

A referential integrity constraint (RIC) like

$$\forall\bar{x}\big(p(\bar{x}) \rightarrow \exists y q(\bar{x}', y)\big) \tag{3.1}$$

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 (2.1). For that reason, to extend the methodology for embedding formulas is required. A RIC like (3.1) will be embedded into APC by means of:

$$p(\bar{x}){:}\mathbf{f_c} \vee \exists y(q(\bar{x}', y){:}\mathbf{t_c}). \tag{3.2}$$

Now IC is allowed to contain, in addition to ICs of the form (2.1), RICs like (3.1). The one-to-one correspondence between minimal models of the new theory $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$ and the repairs of DB still holds. This is proved as follows:

Lemma 4. Given two database instances DB and DB', if $DB' \models_\Sigma IC$, then $\mathcal{M}(DB, DB') \models \mathcal{T}(\mathbf{DB}, \mathbf{IC})$.

Proof: We have to show $\mathcal{M}(DB, DB') \models p(\bar{x}){:}\mathbf{f_c} \vee \exists y(q(\bar{x}', y){:}\mathbf{t_c})$. The rest of the proof was considered in (Arenas et al. 2000a). We have that IC contains the formula $p(\bar{x}) \rightarrow \exists y q(\bar{x}', y)$. As $DB' \models_\Sigma IC$ we must analyze two cases. The first one is $DB' \models_\Sigma \neg p(\bar{a})$. Then $I_P(p(\bar{a})) = \mathbf{f}$ or $I_P(p(\bar{a})) = \mathbf{f_a}$, so $\mathcal{M}(DB, DB') \models p(\bar{a}){:}\mathbf{f_c}$. The second case is $DB' \models_\Sigma q(\bar{a}', b_1), \ldots, q(\bar{a}', b_n)$ for elements $b_1, \ldots, b_n$ in the domain $(n \geq 1)$. Hence, $I_P(q(\bar{a}', b_i)) = \mathbf{t}$ or $I_P(q(\bar{a}', b_i)) = \mathbf{t_a}$, for every $1 \leq i \leq n$. Then,

$\mathcal{M}(DB, DB') \models \exists y(q(\bar{a}', y){:}\mathbf{t_c})$. Since the analysis was done for an arbitrary value $\bar{a}$, we have that $\mathcal{M}(DB, DB') \models \mathcal{T}(\mathbf{DB}, \mathbf{IC})$. $\square$

Lemma 5. If $\mathcal{M}$ is a model of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$ such that $DB_{\mathcal{M}}$ is finite, then $DB_{\mathcal{M}} \models_{\Sigma} IC$.

Proof: We have to show $DB_{\mathcal{M}} \models_{\Sigma} p(\bar{x}) \rightarrow \exists y q(\bar{x}', y)$. The rest of the proof was given in (Arenas et al. 2000a). Let us suppose first $\mathcal{M} \models p(\bar{a}){:}\mathbf{f_c}$. Then, we either have $\mathcal{M} \models p(\bar{a}){:}\mathbf{f}$ or $\mathcal{M} \models p(\bar{a}){:}\mathbf{f_a}$. Hence, $DB_{\mathcal{M}} \models_{\Sigma} \neg p(\bar{a})$, and from there $DB_{\mathcal{M}} \models_{\Sigma} p(\bar{a}) \rightarrow \exists y q(\bar{a}', y)$. Let us suppose now $\mathcal{M} \models \exists y(q(\bar{a}', y){:}\mathbf{t_c})$. Therefore, $\mathcal{M} \models q(\bar{a}', b){:}\mathbf{t}$ or $\mathcal{M} \models q(\bar{a}', b){:}\mathbf{t_a}$ for some element b in the domain. Hence $DB_{\mathcal{M}} \models_{\Sigma} q(\bar{a}', b)$, and from there $DB_{\mathcal{M}} \models_{\Sigma} p(\bar{a}) \rightarrow \exists y q(\bar{a}', y)$. Since this is valid for any value $\bar{a}$, we have that $DB_{\mathcal{M}} \models_{\Sigma} p(\bar{x}) \rightarrow \exists y q(\bar{x}', y)$. $\square$

The following results shows the one-to one correspondence between most e-consistent models of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$ and repairs of DB.

Proposition 1. If DB' is a repair of DB with respect to the set of integrity constraints IC, then $\mathcal{M}(DB, DB')$ is Δ-minimal among the models of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$.

Proof: By Lemma 4, we conclude that $\mathcal{M}(DB, DB') \models \mathcal{T}(\mathbf{DB}, \mathbf{IC})$. Let us assume that $\mathcal{M}(DB, DB')$ is not Δ-minimal in the class of models of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$. Then, there exists $\mathcal{M} \models \mathcal{T}(\mathbf{DB}, \mathbf{IC})$, such that $\mathcal{M} <_{\Delta} \mathcal{M}(DB, DB')$. By using this it is possible to prove that $\Delta(DB, DB_{\mathcal{M}}) \subsetneq \Delta(DB, DB')$.

a. Let us suppose that $p(\bar{a}) \in \Delta(DB, DB_{\mathcal{M}})$. Then $p(\bar{a}) \in DB$ and $p(\bar{a}) \notin DB_{\mathcal{M}}$, or $p(\bar{a}) \notin DB$ and $p(\bar{a}) \in DB_{\mathcal{M}}$. In the first case we can conclude that $p(\bar{a}){:}\mathbf{t_d} \in \mathcal{T}(\mathbf{DB}, \mathbf{IC})$ and $\mathcal{M} \models p(\bar{a}){:}\mathbf{f} \vee p(\bar{a}){:}\mathbf{f_a}$. If we suppose that $\mathcal{M} \models p(\bar{a}){:}\mathbf{f}$, then $\mathcal{M} \not\models p(\bar{a}){:}\mathbf{t_d}$, a contradiction. Thus, we have that $\mathcal{M} \models p(\bar{a}){:}\mathbf{f_a}$. But $\mathcal{M} <_{\Delta} \mathcal{M}(DB, DB')$, and therefore $\mathcal{M}(DB, DB') \models p(\bar{a}){:}\mathbf{f_a}$. Then, we conclude that $p(\bar{a}) \notin DB'$, and therefore in this case it is possible to conclude that $p(\bar{a}) \in \Delta(DB, DB')$. In the second case we can conclude that $p(\bar{a}){:}\mathbf{f_d} \in \mathcal{T}(\mathbf{DB}, \mathbf{IC})$ and $\mathcal{M} \models p(\bar{a}){:}\mathbf{t} \vee p(\bar{a}){:}\mathbf{t_a}$. If we suppose that $\mathcal{M} \models p(\bar{a}){:}\mathbf{t}$, then $\mathcal{M} \not\models p(\bar{a}){:}\mathbf{f_d}$, a contradiction. Thus, we have that $\mathcal{M} \models p(\bar{a})\!:\!\mathbf{t_a}$. But $\mathcal{M} <_\Delta \mathcal{M}(DB, DB')$, and therefore $\mathcal{M}(DB, DB') \models p(\bar{a})\!:\!\mathbf{t_a}$. Then, we conclude that $p(\bar{a}) \in DB'$, and therefore $p(\bar{a}) \in \Delta(DB, DB')$. Thus, we conclude that $\Delta(DB, DB_{\mathcal{M}}) \subseteq \Delta(DB, DB')$.

b. Since $\mathcal{M}(DB, DB') \not\leq_\Delta \mathcal{M}$, there exists $p(\bar{a})$ such that $\mathcal{M}(DB, DB') \models p(\bar{a})\!:\!\mathbf{t_a} \vee p(\bar{a})\!:\!\mathbf{f_a}$ and $\mathcal{M} \models p(\bar{a})\!:\!\mathbf{t} \vee p(\bar{a})\!:\!\mathbf{f}$. By using the first fact it is possible to conclude that $p(\bar{a}) \in \Delta(DB, DB')$. If we suppose that $p(\bar{a}) \in DB$, then $p(\bar{a})\!:\!\mathbf{t_d} \in \mathcal{T}(\mathbf{DB}, \mathbf{IC})$, and therefore by considering the second fact it is possible to deduce that $\mathcal{M}$ must satisfy $p(\bar{a})\!:\!\mathbf{t}$. Thus, we can conclude that in this case $p(\bar{a}) \in DB_{\mathcal{M}}$, and therefore $p(\bar{a}) \notin \Delta(DB, DB_{\mathcal{M}})$. By the other hand, if we suppose that $p(\bar{a}) \notin DB$, then $p(\bar{a})\!:\!\mathbf{f_d} \in \mathcal{T}(\mathbf{DB}, \mathbf{IC})$, and therefore by considering the second fact it is possible to deduce that $\mathcal{M}$ must satisfy $p(\bar{a})\!:\!\mathbf{f}$. Thus, in this case $p(\bar{a}) \notin DB_{\mathcal{M}}$, and therefore $p(\bar{a}) \notin \Delta(DB, DB_{\mathcal{M}})$. Finally, we conclude that $\Delta(DB, DB') \not\subseteq \Delta(DB, DB_{\mathcal{M}})$.

We know that DB' is a database instance, and therefore $\Delta(DB, DB')$ must be a finite set. Thus, $\Delta(DB, DB_{\mathcal{M}})$ is a finite set, and therefore $DB_{\mathcal{M}}$ is a database instance. With the help of Lemma 5, we deduce that $DB_{\mathcal{M}} \models IC$. But this a contradiction, since DB' is a repair of DB with respect to IC and $\Delta(DB, DB_{\mathcal{M}}) \subsetneq \Delta(DB, DB')$ □.

Proposition 2. Let $\mathcal{M}$ be a model of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$. If $\mathcal{M}$ is minimal and $DB_{\mathcal{M}}$ is finite, then $DB_{\mathcal{M}}$ is a repair of DB with respect to IC.

Proof: By Lemma 5, we conclude that $DB_{\mathcal{M}} \models_\Sigma IC$. Now, we need to prove that $DB_{\mathcal{M}}$ is minimal. Let us suppose this is not true. Then, there is a database instance DB^* such that $DB^* \models_\Sigma IC$ and $\Delta(DB, DB^*) \subsetneq \Delta(DB, DB_{\mathcal{M}})$.

a. From Lemma 4, we conclude that $\mathcal{M}(DB, DB^*) \models \mathcal{T}(\mathbf{DB}, \mathbf{IC})$.

b. Now, we are going to prove that $\mathcal{M}(DB, DB^*) <_\Delta \mathcal{M}$.

If $\mathcal{M}(DB, DB^*) \models p(\bar{a})\!:\!\mathbf{t_a}$, then we can conclude that $p(\bar{a}) \notin DB$ and $p(\bar{a}) \in DB^*$, and therefore $p(\bar{a}) \in \Delta(DB, DB^*)$. But $\Delta(DB, DB^*) \subsetneq \Delta(DB, DB_{\mathcal{M}})$, and therefore $p(\bar{a}) \in DB_{\mathcal{M}}$. Thus, we can conclude that $\mathcal{M} \models p(\bar{a})\!:\!\mathbf{t} \vee p(\bar{a})\!:\!\mathbf{t_a}$.

If we suppose that $\mathcal{M} \models p(\bar{a}):\mathbf{t}$, then $\mathcal{M} \not\models p(\bar{a}):\mathbf{f_d}$, but we know that $\mathcal{M} \models \mathcal{T}(\mathbf{DB},\mathbf{IC})$ and $p(\bar{a}){:}\mathbf{f_d} \in \mathcal{T}(\mathbf{DB},\mathbf{IC})$, since $p(\bar{a}) \notin DB$, a contradiction. Therefore, $\mathcal{M} \models p(\bar{a}){:}\mathbf{t_a}$.

If $\mathcal{M}(DB, DB^*) \models p(\bar{a}){:}\mathbf{f_a}$, then we can conclude that $p(\bar{a}) \in DB$ and $p(\bar{a}) \notin DB^*$, and therefore $p(\bar{a}) \in \Delta(DB, DB^*)$. But $\Delta(DB, DB^*) \subsetneq \Delta(DB, DB_{\mathcal{M}})$, and therefore $p(\bar{a}) \notin DB_{\mathcal{M}}$. Thus, we can conclude that $\mathcal{M} \models p(\bar{a}){:}\mathbf{f} \vee p(\bar{a}){:}\mathbf{f_a}$. If we suppose that $\mathcal{M} \models p(\bar{a}):\mathbf{f}$, then $\mathcal{M} \not\models p(\bar{a}):\mathbf{t_d}$, but we know that $\mathcal{M} \models \mathcal{T}(\mathbf{DB},\mathbf{IC})$ and $p(\bar{a}){:}\mathbf{t_d} \in \mathcal{T}(\mathbf{DB},\mathbf{IC})$, since $p(\bar{a}) \in DB$, a contradiction. Therefore, $\mathcal{M} \models p(\bar{a}){:}\mathbf{f_a}$. Thus, we deduce that $\mathcal{M}(DB, DB^*) \leq_\Delta \mathcal{M}$.

Finally, we know that there exists $p(\bar{a})$ such that it is not in $\Delta(DB, DB^*)$ and it is in $\Delta(DB, DB_{\mathcal{M}})$. Thus, $p(\bar{a}) \in DB$ and $p(\bar{a}) \in DB^*$, and therefore $\mathcal{M}(DB, DB^*) \models p(\bar{a}):\mathbf{t}$, or $p(\bar{a}) \notin DB$ and $p(\bar{a}) \notin DB^*$, and therefore $\mathcal{M}(DB, DB^*) \models p(\bar{a}):\mathbf{f}$. Then, we have that $\mathcal{M}(DB, DB^*) \not\models p(\bar{a}):\mathbf{t_a}$ and $\mathcal{M}(DB, DB^*) \not\models p(\bar{a}){:}\mathbf{f_a}$. Additionally, since $p(\bar{a}) \in \Delta(DB, DB_{\mathcal{M}})$, we can conclude that $p(\bar{a}) \in DB$ and $p(\bar{a}) \notin DB_{\mathcal{M}}$, or $p(\bar{a}) \notin DB$ and $p(\bar{a}) \in DB_{\mathcal{M}}$. In the first case we conclude that $\mathcal{M} \models p(\bar{a}){:}\mathbf{f_a}$. In the second case we conclude that $\mathcal{M} \models p(\bar{a}){:}\mathbf{t_a}$. Thus, $\mathcal{M} \models p(\bar{a}){:}\mathbf{t_a} \vee p(\bar{a}){:}\mathbf{f_a}$. Therefore we deduce that $\mathcal{M} \not\leq_\Delta \mathcal{M}(DB, DB^*)$.

Finally, we deduce that $\mathcal{M}$ is not Δ-minimal in the class of the models of $\mathcal{T}(\mathbf{DB},\mathbf{IC})$, a contradiction. □

Example 4. Consider the relational schema of Example 1 extended with table *Author*(*name*, *citizenship*). Now, *IC* also contains the RIC:

$$\forall x \forall y \forall z \big(Book(x,y,z) \rightarrow \exists w Author(x,w)\big)$$

expressing that every writer of a book in the database instance must be registered as an author. The theory $\mathcal{T}(\mathbf{IC})$ now also contains:

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

$$Author(x,w){:}\mathbf{f_c} \vee Author(x,w){:}\mathbf{t_c}\ , \quad \neg Author(x,w){:}\mathbf{f_c} \vee \neg Author(x,w){:}\mathbf{t_c}.$$

It might also be possible to have the $FD: \ name \rightarrow citizenship$, and thus to have a foreign key constraint. The database instance $\{Book(neruda, 20lovepoems, 1924)\}$ is inconsistent wrt the given RIC. If the following subdomain was given

$$D(Author.citizenship) = \{chilean, canadian\}$$

for the attribute "citizenship", the following database theory is obtained:

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

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

$$\begin{aligned}\mathcal{M}_1 \ &= \ \{Book(neruda, 20lovepoems, 1924){:}\mathbf{f_a}, \ Author(neruda, chilean){:}\mathbf{f}, \\ &\quad Author(neruda, canadian){:}\mathbf{f}, \ldots\} \\ \mathcal{M}_2 \ &= \ \{Book(neruda, 20lovepoems, 1924){:}\mathbf{t}, \ Author(neruda, chilean){:}\mathbf{t_a}, \\ &\quad Author(neruda, canadian){:}\mathbf{f}, \ldots\} \\ \mathcal{M}_3 \ &= \ \{Book(neruda, 20lovepoems, 1924){:}\mathbf{t}, \ Author(neruda, chilean){:}\mathbf{f}, \\ &\quad Author(neruda, canadian){:}\mathbf{t_a}, \ldots\}.\end{aligned}$$

It is obtained $DB_{\mathcal{M}_1} = \emptyset$, $DB_{\mathcal{M}_2} = \{Book(neruda, 20lovepoems, 1924), Author$ $(neruda, chilean)\}$ and $DB_{\mathcal{M}_3}$ similar to $DB_{\mathcal{M}_2}$, but with a Canadian Neruda. According to proposition 2, 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.

3.2 Annotating General Database Constraints

The class of ICs found in database praxis is contained in the class of constraints of the form:

$$\forall \bar{x}\ \big(\varphi(\bar{x}) \rightarrow \exists \bar{z} \psi(\bar{y})\big) \tag{3.3}$$

where φ and ψ are (possibly empty) conjunctions of literals in $P \cup B$, and $\bar{z} = \bar{y} - \bar{x}$. This class includes the ones identified in (Abiteboul et al. 1995, chapter 10), and all those of the form (2.1). In this class are found, among others, range constraints (e.g. $\forall x\ (p(x) \rightarrow x > 30)$), join dependencies, functional dependencies, inclusion dependencies and referential integrity constraints.

If $\varphi(\bar{x})$ is $\bigwedge_{i=1}^{k} p_i(\bar{x}_i) \wedge \bigwedge_{i=k+1}^{m} \neg p_i(\bar{x}_i)$, with $\bar{x} = \bigcup_{i=1}^{m} \bar{x}_i$, and $\psi(\bar{y})$ is $\bigwedge_{j=1}^{l} q_j(\bar{y}_j) \wedge \bigwedge_{j=l+1}^{r} \neg q_j(\bar{y}_j)$, with $\bar{y} = \bigcup_{j=1}^{r} \bar{y}_j$, the annotation methodology can be extended, embedding these constraints into APC as follows:

$$\bigvee_{i=1}^{k} p_i(\bar{x}_i){:}\mathbf{f_c} \vee \bigvee_{i=k+1}^{m} p_i(\bar{x}_i){:}\mathbf{t_c}\ \vee\ \exists \bar{z}(\bigwedge_{j=1}^{l} q_j(\bar{y}_j){:}\mathbf{t_c} \wedge \bigwedge_{j=l+1}^{r} q_j(\bar{y}_j){:}\mathbf{f_c}).$$

Now, if the given set of ICs is allowed to contain any constraint of the form (3.3), then it is possible to prove that the one-to-one correspondence between minimal models of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$ and the repairs of DB still holds. The following results are equal to lemmas 4 and 5, but for the case IC contains constraints of the form 3.3.

Lemma 6. Given two database instances DB and DB', if $DB' \models_\Sigma IC$, then $\mathcal{M}(DB, DB') \models \mathcal{T}(\mathbf{DB}, \mathbf{IC})$.

Proof: We have to show $\mathcal{M}(DB, DB') \models \bigvee_{i=1}^{k} p_i(\bar{x}_i) : \mathbf{f_c} \vee \bigvee_{i=k+1}^{m} p_i(\bar{x}_i) : \mathbf{t_c} \vee \exists \bar{z}(\bigwedge_{j=1}^{l} q_j(\bar{y}_j) : \mathbf{t_c} \wedge \bigwedge_{j=1+1}^{r} q_j(\bar{y}_j) : \mathbf{f_c})$. As $DB' \models_\Sigma IC$, we must analyze some cases. The first one is $DB' \models_\Sigma \neg p_i(\bar{a})$, for some $1 \leq i \leq k$. Then $I_P(p_i(\bar{a})) = \mathbf{f}$ or $I_P(p_i(\bar{a})) = \mathbf{f_a}$, so $\mathcal{M}(DB, DB') \models p_i(\bar{a}){:}\mathbf{f_c}$. Another case is $DB' \models_\Sigma p_i(\bar{a})$, for some $k + 1 \leq i \leq m$. Then $I_P(p_i(\bar{a})) = \mathbf{t}$ or $I_P(p_i(\bar{a})) = \mathbf{t_a}$, so $\mathcal{M}(DB, DB') \models p_i(\bar{a}){:}\mathbf{t_c}$. The last case is $DB' \models_\Sigma \bigwedge_{j=1}^{l} q_j(\bar{c}_j) \wedge \bigwedge_{j=l+1}^{r} \neg q_j(\bar{c}_j)$. As $q_j(\bar{y}_j)$ can be written as $q_j(\bar{x_j}', \bar{z}_j)$, where $\bar{x_j}'$ are the variables in $\bar{x}$ and in $\bar{y}_j$, and $\bar{z}_j$ are the rest of the variables in $\bar{y_j}$, we have that $I_P(q_j((\bar{a}_j)', \bar{b}_j)) = \mathbf{t}$ or $I_P(q_j((\bar{a}_j)', \bar{b}_j)) = \mathbf{t_a}$, for every $1 \leq j \leq l$, and $I_P(q_j((\bar{a}_j)', \bar{b}_j)) = \mathbf{f}$ or $I_P(q_j((\bar{a}_j)', \bar{b}_j)) = \mathbf{f_a}$, for every $l+1 \leq j \leq r$. Then, $\mathcal{M}(DB, DB') \models \exists \bar{z}(\bigwedge_{j=1}^{l} q_j((\bar{a}_j)', z_j){:}\mathbf{t_c} \wedge \bigwedge_{j=l+1}^{r} q_j((\bar{a}_j)', z_j){:}\mathbf{f_c})$. Since the analysis was done for an arbitrary value $\bar{a}$, we have that $\mathcal{M}(DB, DB') \models \mathcal{T}(\mathbf{DB}, \mathbf{IC})$. □

Lemma 7. If $\mathcal{M}$ is a model of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$ such that $DB_{\mathcal{M}}$ is finite, then $DB_{\mathcal{M}} \models_{\Sigma} IC$.

Proof: We have to show $DB_{\mathcal{M}} \models_{\Sigma} \bigvee_{i=1}^{k} \neg p_i(\bar{x}_i) \vee \bigvee_{i=k+1}^{m} p_i(\bar{x}_i) \vee \exists \bar{z}(\bigwedge_{j=1}^{l} q_j(\bar{y}_j) \wedge \bigwedge_{j=l+1}^{r} \neg q_j(\bar{y}_j))$. The rest of the proof was given in (Arenas et al. 2000a). Let us suppose first $\mathcal{M} \models p_i(\bar{a}){:}\mathbf{f_c}$, for some $1 \leq i \leq k$. Then, we either have $\mathcal{M} \models p_i(\bar{a}){:}\mathbf{f}$ or $\mathcal{M} \models p_i(\bar{a}){:}\mathbf{f_a}$. Hence, $DB_{\mathcal{M}} \models_{\Sigma} \neg p_i(\bar{a})$. Let us suppose now $\mathcal{M} \models p_i(\bar{a}){:}\mathbf{t_c}$, for some $k+1 \leq i \leq n$. Then, we either have $\mathcal{M} \models p_i(\bar{a}){:}\mathbf{t}$ or $\mathcal{M} \models p_i(\bar{a}){:}\mathbf{t_a}$. Hence, $DB_{\mathcal{M}} \models_{\Sigma} p_i(\bar{a})$. Finally, let us suppose $\mathcal{M} \models \exists \bar{z}(\bigwedge_{j=1}^{l} q_j((\bar{a}_j)', \bar{z}_j){:}\mathbf{t_c} \wedge \bigwedge_{j=l+1}^{r} q_j((\bar{a}_j)', \bar{z}_j){:}\mathbf{f_c})$. Therefore, $\mathcal{M} \models q_j((\bar{a}_j)', \bar{b}_j){:}\mathbf{t}$ or $\mathcal{M} \models q_j((\bar{a}_j)', \bar{b}_j){:}\mathbf{t_a}$, for every $1 \leq j \leq l$, and $\mathcal{M} \models q_j((\bar{a}_j)', \bar{b}_j){:}\mathbf{f}$ or $\mathcal{M} \models q_j((\bar{a}_j)', \bar{b}_j){:}\mathbf{f_a}$, for every $l+1 \leq j \leq r$, . Hence $DB_{\mathcal{M}} \models_{\Sigma} \exists \bar{z}(\bigwedge_{j=1}^{l} q_j((\bar{a}_j)', \bar{z}_j) \wedge \bigwedge_{j=l+1}^{r} \neg q_j((\bar{a}_j)', \bar{z}_j))$. Since this is valid for any value $\bar{a}$, we have that $DB_{\mathcal{M}} \models_{\Sigma} IC$. □

The proofs of propositions 1 and 2, stating the biunivocal correspondence between minimal models of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$ and the repairs of the original instance, also hold now and can be proved in the same way that for referential integrity constraints.

IV. ANNOTATION OF QUERIES

According to chapter 2, 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 $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$. However, to pose the query directly to the theory, it is necessary to reformulate it as an annotated formula.

Definition 10. Given a FO query $Q(\bar{x})$ in language Σ, the annotated formula obtained from Q by simultaneously replacing, for $p \in P$, the negative literal $\neg p(\bar{s})$ by the APC formula $p(\bar{s})\text{:}\mathbf{f} \vee p(\bar{s})\text{:}\mathbf{f_a}$, and the positive literal $p(\bar{s})$ by the APC formula $p(\bar{s})\text{:}\mathbf{t} \vee p(\bar{s})\text{:}\mathbf{t_a}$, is denoted by $Q^{an}(\bar{x})$. For $p \in B$, the literal $p(\bar{s})$ is replaced by the APC formula $p(\bar{s})\text{:}\mathbf{t}$. $\Box$

According to this definition, logically equivalent versions of a query could have different annotated versions, but in proposition 3 we prove that they retrieve the same consistent answers.

Example 5. (example 1 cont.) In order to consistently answer the query $Q(x)$: $\neg\exists y\exists z\exists w\exists t(\mathit{Book}(x,y,z) \wedge \mathit{Book}(x,w,t) \wedge y \neq w)$, asking for those authors that have at most one book in DB, the annotated query $Q^{an}(\bar{x})$: $\neg\exists y\exists z\exists w\exists t((\mathit{Book}(x,y,z)\text{:}\mathbf{t} \vee \mathit{Book}(x,y,z)\text{:}\mathbf{t_a}) \wedge (\mathit{Book}(x,w,t)\text{:}\mathbf{t} \vee \mathit{Book}(x,w,t)\text{:}\mathbf{t_a}) \wedge (y \neq w)\text{:}\mathbf{t})$ is generated, to be posed to the annotated theory with its minimal model semantics. $\Box$

Definition 11. If φ is a sentence in the language of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$, then $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$ Δ-minimally entails φ, written $\mathcal{T}(\mathbf{DB}, \mathbf{IC}) \models_\Delta \varphi$, iff every Δ-minimal model $\mathcal{M}$ of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$, such that $DB_{\mathcal{M}}$ is finite, satisfies φ, *i.e.* $\mathcal{M} \models \varphi$. $\Box$

Now the characterization of consistent query answers with respect to the annotated theory is stated.

Proposition 3. Let DB be a database instance, IC a set of integrity constraints and $Q(\bar{x})$ a query in FO language Σ. Then, it holds that: $DB \models_c Q(\bar{t})$ iff $\mathcal{T}(\mathbf{DB}, \mathbf{IC}) \models_\Delta Q^{an}(\bar{t})$.

Proof: We first need the following lemma:

Lemma 8. For a minimal model $\mathcal{M}$ of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$ and formula $\varphi(\bar{x})$ in our first order language, we have $\mathcal{M} \models (\neg\varphi)^{an}(\bar{t})$ iff $\mathcal{M} \models \neg\varphi^{an}(\bar{t})$.

Proof: By induction on φ.

Initial step: $\varphi(\bar{t}) = p(\bar{t})$. Trivial, by the fact that every model of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$ annotates atoms either with $\mathbf{t}$, $\mathbf{f}$, $\mathbf{t_a}$ or $\mathbf{f_a}$.

Inductive step:

- $\varphi(\bar{t}) = \neg\alpha(\bar{t})$. $\mathcal{M} \models (\neg\neg\alpha)^{an}(\bar{t})$ iff $\mathcal{M} \models (\alpha)^{an}(\bar{t})$ iff $\mathcal{M} \not\models \neg(\alpha)^{an}(\bar{t})$ iff $\mathcal{M} \not\models (\neg\alpha)^{an}(\bar{t})$ (by induction hypothesis) iff $\mathcal{M} \models \neg(\neg\alpha)^{an}(\bar{t})$.
- $\varphi(\bar{t}) = \alpha(\bar{t_1}) \vee \beta(\bar{t_2}) = (\alpha \vee \beta)(\bar{t})$, where $\bar{t_1}$ is the restriction of $\bar{t}$ to α (the same for $\bar{t_2}$ and β). $\mathcal{M} \models (\neg(\alpha \vee \beta))^{an}(\bar{t})$ iff $\mathcal{M} \models (\neg\alpha)^{an}(\bar{t_1})$ and $\mathcal{M} \models (\neg\beta)^{an}(\bar{t_2})$ iff $\mathcal{M} \models \neg(\alpha)^{an}(\bar{t_1})$ and $\mathcal{M} \models \neg(\beta)^{an}(\bar{t_2})$ (by induction hypothesis) iff $\mathcal{M} \models \neg(\alpha \vee\ \beta)^{an}(\bar{t})$.
- $\varphi(\bar{t}) = \exists x\alpha(\bar{t})$. $\mathcal{M} \models (\neg\exists x\alpha)^{an}(\bar{t})$ iff $\mathcal{M} \models (\forall x\neg\alpha)^{an}(\bar{t})$ iff $\mathcal{M} \models \forall x(\neg\alpha)^{an}(\bar{t})$ iff $\mathcal{M} \models \neg\exists x(\alpha)^{an}(\bar{t})$ (by induction hypothesis) iff $\mathcal{M} \models \neg(\exists x\alpha)^{an}(\bar{t})$.

□

We will prove proposition 3 by induction on φ.

Initial step: $\varphi(\bar{x}) = p(\bar{x})$. $DB \models_c p(\bar{t})$ iff for every repair DB' of DB, $DB' \models_\Sigma p(\bar{t})$ iff for every minimal model $\mathcal{M}$ of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$, such that $DB_{\mathcal{M}}$ is finite, $\mathcal{M} \models p(\bar{t})$:$\mathbf{t} \vee p(\bar{t})$:$\mathbf{t_a}$ iff $\mathcal{T}(\mathbf{DB}, \mathbf{IC}) \models_\Delta p(\bar{t})$:$\mathbf{t} \vee p(\bar{t})$:$\mathbf{t_a}$.

Inductive step:

- $\varphi(\bar{x}) = \neg\alpha(\bar{x})$. $DB \models_c \neg\alpha(\bar{t})$ iff for every repair DB' of DB we have that $DB' \not\models_\Sigma \alpha(\bar{t})$ iff for every minimal model $\mathcal{M}$ of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$, $\mathcal{M} \not\models \alpha^{an}(\bar{t})$ (by induction hypothesis) iff for every minimal model $\mathcal{M}$ of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$, such that $DB_\mathcal{M}$ is finite, $\mathcal{M} \models \neg\alpha^{an}(\bar{t})$ iff $\mathcal{M} \models (\neg\alpha)^{an}(\bar{t})$ (by Lemma 8).

- $\varphi(\bar{x}) = \alpha(\bar{x_1}) \vee \beta(\bar{x_2}) = (\alpha \vee \beta)(\bar{x})$. $DB \models_c (\alpha \vee \beta)(\bar{t})$ iff for every repair DB' of DB it is true that $DB' \models_\Sigma \alpha(\bar{t_1})$ or $DB' \models_\Sigma \beta(\bar{t_2})$, where $\bar{t_i}$ is the restriction of substitution $\bar{t}$ to the variables $\bar{x_i}$, iff for every minimal model $\mathcal{M}$ of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$, such that $DB_\mathcal{M}$ is finite, $\mathcal{M} \models \alpha^{an}(\bar{t_1})$ or $\mathcal{M} \models \beta^{an}(\bar{t_2})$ (by induction hypothesis) iff $\mathcal{T}(\mathbf{DB}, \mathbf{IC}) \models_\Delta (\alpha^{an} \vee \beta^{an})(\bar{t})$ iff $\mathcal{T}(\mathbf{DB}, \mathbf{IC}) \models_\Delta (\alpha \vee \beta)^{an}(\bar{t})$.

- $\varphi(\bar{x}) = \exists y \alpha(\bar{x})$. $DB \models_c \exists y \alpha(\bar{t})$ iff for every repair DB' of DB there exists an element b in the domain such that it is true that $DB' \models_\Sigma \alpha(\bar{t})\{\frac{b}{y}\}$, where $\{\frac{b}{y}\}$ is the substitution of y for b, iff for every minimal model $\mathcal{M}$ of $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$, such that $DB_\mathcal{M}$ is finite, there exists element b in the domain such that $\mathcal{M} \models (\alpha(\bar{t})\{\frac{b}{y}\})^{an}$ (by induction hypothesis) iff $\mathcal{T}(\mathbf{DB}, \mathbf{IC}) \models_\Delta (\exists y \alpha(\bar{t}))^{an}$.

□

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

According to this proposition, in order to consistently answer queries, the problem of evaluating minimal entailment with respect to the annotated theory must be faced. In (Arenas et al. 2000a) some limited FO queries were evaluated, but no annotated queries were generated. The original query was answered using ad hoc algorithms that were extracted from theory $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$. No advantage was taken from a characterization of consistent answers in terms of minimal entailment from $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$. In the next chapter it will be addressed 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 $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$. The query to be posed to the logic program will be built from Q^{an}.

V. SPECIFICATION OF REPAIRS WITH LOGIC PROGRAMS

The idea of this chapter is to present, based on the generated annotated theory, disjunctive first order programs, in such a way that some identified models of these correspond to the repairs of the original instance. Also, a methodology is developed for obtaining consistent answers from these models. Finally, an implementation in *DLV* of the programs and an extension to consider referential integrity constraints are presented.

5.1 Logic Programming Specification of Repairs

At first ICs of the form (2.1) will be considered, more precisely of the form

$$\bigvee_{i=1}^{n} \neg p_i(\bar{t}_i) \vee \bigvee_{j=1}^{m} q_j(\bar{s}_j) \vee \varphi, \tag{5.1}$$

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.

In order to generate a first order logic program that gives an account of annotations, for each predicate $p(\bar{x}) \in P$, a new predicate $p(\bar{x}, \cdot)$, with an extra argument for annotations, is introduced. 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:

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

   where $\mathbf{t}^\star, \mathbf{f}^\star$ are new, auxiliary elements in the domain of annotations. It is not necessary to specify how they interact with the previous annotations.

3. For every constraint of the form (5.1), $\Pi(DB, IC)$ 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. It is mandatory to make sure that the repairing process continues and stabilizes in a state where all the ICs hold[1]. This is the role of the clauses in 2. containing the new annotation $\mathbf{t}^\star$, that groups together those atoms annotated with $\mathbf{t_d}$ and $\mathbf{t_a}$, and the new annotation $\mathbf{f}^\star$, that does the same with $\mathbf{f_d}$ and $\mathbf{f_a}$, but with the help of Definition 12 below, since there are no atoms with annotation argument $\mathbf{f_d}$ in the program.

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.

Example 7. (example 1 cont.) The schema is extended with table

$$\mathit{Eurbook}(\mathit{author}, \mathit{name}, \mathit{publYear}),$$

for European books. Now, DB also contains the literal $\mathit{Eurbook}(\ \mathit{kafka}, \mathit{metamorph}, \mathit{1919})$. If in addition to the ICs stated before, the set inclusion dependency

$$\forall xyz\ (\mathit{Eurbook}(x, y, z) \rightarrow \mathit{Book}(x, y, z))$$

[1]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".

is included in IC, the following program $\Pi(DB, IC)$ is obtained:

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

2. $\mathit{Book}(x, y, z, \mathbf{t}^\star) \leftarrow \mathit{Book}(x, y, z, \mathbf{t_d})$, $\mathit{Book}(x, y, z, \mathbf{t}^\star) \leftarrow \mathit{Book}(x, y, z, \mathbf{t_a})$, $\mathit{Book}(x, y, z, \mathbf{f}^\star) \leftarrow \mathit{Book}(x, y, z, \mathbf{f_a})$, $\mathit{Eurbook}(x, y, z, \mathbf{t}^\star) \leftarrow \mathit{Eurbook}(x, y, z, \mathbf{t_d})$, $\mathit{Eurbook}(x, y, z, \mathbf{t}^\star) \leftarrow \mathit{Eurbook}(x, y, z, \mathbf{t_a})$, $\mathit{Eurbook}(x, y, z, \mathbf{f}^\star) \leftarrow \mathit{Eurbook}(x, y, z, \mathbf{f_a})$.

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

A semantics for the repair programs is required. First, it is necessary to define the models of the program. Since the negative information in a database instance is only implicitly available and it is wished to avoid representing it, it is required to specify when negative information of the form $p(\bar{t}, \mathbf{f}^\star)$ is true of a model.

Definition 12. (a) Let I be a Herbrand structure 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 $p(\bar{a}, \mathbf{f}^\star)$: $I \models_\star p(\bar{a}, \mathbf{f}^\star)$ iff $p(\bar{a}, \mathbf{f}^\star) \in I$ or $p(\bar{a}, \mathbf{t_d}) \notin I$, holds.
(b) A Herbrand structure $\mathcal{M}$ 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 13. (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 $p(\bar{a}, \mathbf{f_a})$. □

The interest will be centered only in the Herbrand $\star$-models of the program that are minimal with respect to set inclusion[2] and plausible and coherent. Notice that in a coherent model both atoms $p(\bar{a}, \mathbf{t}^\star)$ and $p(\bar{a}, \mathbf{f}^\star)$ could still be

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

found. Notice also that a plausible atom may belong to a supported disjunctive head (Lobo et al. 1998), without the other disjuncts being forced to be false.

It is easy to see from the definition of a $\star$-model of a program that the classical notion of satisfaction could be kept 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 that is usually kept implicit via the closed world assumption. Moreover, a normal disjunctive program would be obtained, for which a stable model semantics could be used (Lifschitz 1996), as it will be seen later.

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

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

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

$\square$

Notice, that in contrast to the minimal models of the annotated theory $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$, 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 it should be possible to "read"database repairs. Every $\star$-model of the logic program has to be interpreted.

Definition 14. 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:

- If $p(\bar{a}, \mathbf{f_a})$ belongs to $\mathcal{M}$, then $p(\bar{a}, \mathbf{f}^{\star\star})$ belongs to $i(\mathcal{M})$.
- 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 $i(\mathcal{M})$.
- 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 $i(\mathcal{M})$.
- 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 $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$. A similar role plays the other new annotation with respect to the false annotations. These new annotations will simplify the expression of the queries to be posed to the program (see section 5.5). Without them, instead of simply asking $p(\bar{x}, \mathbf{t}^{\star\star})$ (for the tuples in a repair), it would be required 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:

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

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

□

The interpreted models could be easily obtained by adding new rules to the program $\Pi(DB, IC)$. This will be shown in section 5.5. From an interpreted model of the program a database instance can be obtained:

Definition 15. $DB^{\Pi}_{i(\mathcal{M})} = \{p(\bar{a}) \mid i(\mathcal{M}) \models p(\bar{a}, \mathbf{t}^{\star\star})\}$. □

Example 10. (example 9 cont.) The following database instances obtained from definition 15 are the repairs of DB:

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

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

Definition 16. From two database instances DB_1 and DB_2 over the same domain, $\mathcal{M}^\star(DB_1, DB_2)$ is the Herbrand structure $< D, I_P, I_B >$, where D is the domain of the database [3] and I_P, I_B are the interpretations for the database predicates (extended with annotation arguments) and the built-ins, respectively. I_P is defined as follows:

- If $p(\bar{a}) \in DB_1$ and $p(\bar{a}) \in DB_2$, then $p(\bar{a}, \mathbf{t_d})$ and $p(\bar{a}, \mathbf{t}^\star) \in I_P$.
- If $p(\bar{a}) \in DB_1$ and $p(\bar{a}) \notin DB_2$, then $p(\bar{a}, \mathbf{t_d})$, $p(\bar{a}, \mathbf{t}^\star)$, $p(\bar{a}, \mathbf{f_a})$ and $p(\bar{a}, \mathbf{f}^\star) \in I_P$.
- If $p(\bar{a}) \notin DB_1$ and $p(\bar{a}) \notin DB_2$, then for any annotation value v, the atom $p(\bar{a}, v) \notin I_P$.
- If $p(\bar{a}) \notin DB_1$ and $p(\bar{a}) \in DB_2$, then $p(\bar{a}, \mathbf{t_a})$ and $p(\bar{a}, \mathbf{t}^\star) \in I_P$.

The interpretation I_B is defined as expected: if q is a built-in, then $q(\bar{a}) \in I_B$ iff $q(\bar{a})$ is true in classical logic, and $q(\bar{a}) \notin I_B$ iff $q(\bar{a})$ is false. □

As with $\mathcal{M}(DB_1, DB_2)$ (definition 9), the intent here is to use these structures as the basis for construction of database repairs. In fact, when DB_1 is inconsistent and DB_2 is a repair, $\mathcal{M}^\star(DB_1, DB_2)$ contains the facts that are necessary for satisfying the program. Notice that the definition of $\mathcal{M}^\star(DB_1, DB_2)$ is not symmetric in the arguments.

It can be shown (lemmas 12 and 13) that there is a one-to-one correspondence between the coherent plausible $\star$-models of the program $\Pi(DB, IC)$ and

[3]Strictly speaking, the domain D now also contains the annotations values.

some distinguished instances of the database schema, called *pseudo-repairs* of the original instance (definition 20). As the process of repairing an inconsistent database instance considers interactuation between constraints and is not always solvable in one step, it can be seen as a finite chain of different instances, where one step solves the constraints the previous instance did not satisfy. This is the intuitive idea behind the notion of *pseudo-repair* that will be defined next. The fact that the chain is finite is very important: since database instances are always finite, a repair can be obtained from its original instance in a finite number of steps of deletions and insertions of tuples.

Definition 17. From two database instances DB and DB', $DB \curvearrowright DB'$ holds iff:

- For all φ in IC, if $DB \not\models_\Sigma \varphi$ then $DB' \models_\Sigma \varphi$.
- For every atom $q(\bar{b})$ in $(DB' - DB)$, there is a ground instance of a constraint $\phi = \bigvee_{i=1}^{m} \neg p_i(\bar{a}_i) \vee \bigvee_{l=1}^{r} q_l(\bar{b}_l) \vee \varphi$ in IC such that $DB \not\models_\Sigma \phi$ and $q(\bar{b})$ is $q_l(\bar{b}_l)$, for some $1 \leq l \leq r$.
- For every atom $p(\bar{a})$ in $(DB - DB')$, there is a ground instance of a constraint $\phi = \bigvee_{i=1}^{m} \neg p_i(\bar{a}_i) \vee \bigvee_{l=1}^{r} q_l(\bar{b}_l) \vee \phi$ in IC such that $DB \not\models_\Sigma \phi$ and $p(\bar{a})$ is $p_i(\bar{a}_i)$, for some $1 \leq i \leq m$.

□

Intuitively, $DB \curvearrowright DB'$ says that DB' repairs DB with respect to the set of constraints that DB did not satisfy. It is still possible $DB' \not\models_\Sigma IC$, but this is due to a different set of constraints than the previous one. Moreover, every element in $\Delta(DB, DB')$ must be justified in terms of the violation of a constraint by DB (but possibly not the same). Although, the next example shows that not every chain built step by step, where one step solves the constraints the previous instance did not satisfy, is finite:

Example 11. Consider the database instance $DB = \{p(\bar{a})\}$ and the set of constraints $IC = \{p(\bar{x}) \rightarrow q(\bar{x}),\ q(\bar{x}) \rightarrow p(\bar{x})\}$. For the chain of instances $\{p(\bar{a})\}, \{q(\bar{a})\}$,

$\{p(\bar{a})\}, \{q(\bar{a})\}, \ldots$ ad infinitum, it is true that every instance is in the relation $\curvearrowright$ with the previous one. □

The problem with the previous example is that atoms deleted (inserted) in one step of the chain are inserted (deleted) later. This is solved by the next definition. Moreover, definition 18 excludes as a possible *pseudo-repair* any finite subchain contained in the previous infinite one, if its length is $n > 2$.

Definition 18. A chain $DB_0, \cdots, DB_n$ $(n \geq 0)$ of database instances satisfies the plausibility relation iff either $n = 0$ or, when $n > 0$, the following conditions are satisfied:

- $DB_{j-1} \curvearrowright DB_j$, for $1 \leq j \leq n$.
- For all $1 \leq j \leq n$, if $p(a) \in (DB_j - DB_{j-1})$, then $p(a) \notin (DB_{k-1} - DB_k)$ for $j < k < n$.
- For all $1 \leq j \leq n$, if $p(a) \in (DB_{j-1} - DB_j)$, then $p(a) \notin (DB_k - DB_{k-1})$ for $j < k < n$.

□

This means that for a chain of instances to satisfy the plausibility relation it must be true that an element is not inserted in a step of the chain and then deleted in a successive step (or viceversa). This represents the fact that the models that are being considered are the coherent ones. The relation is called plausibility, as it is closely related to the notion of plausibility in the models of the program $\Pi(DB, IC)$ due to the last two conditions in definition 17. In the plausible models an atom is deleted or inserted from the original instance only when it participates in the violation of a constraint. Note from definition 18 that a single database instance (when $n = 0$) always satisfies the plausibility relation.

Definition 19. The database instances DB and DB' satisfy the pseudo-repair relation, written $\mathcal{PS}(DB, DB')$, iff there is a finite chain $DB = DB_0, \cdots, DB_n = DB'$ $(n \geq 0)$ that satisfies the plausibility relation. □

Definition 20. A database instance DB' is a pseudo-repair of a database instance DB with respect to IC, if $DB' \models_\Sigma IC$ and it is the case that $\mathcal{PS}(DB, DB')$. $\square$

Lemma 9. Consider a database instance DB that is consistent with respect to IC, *i.e.* $DB \models_\Sigma IC$. Then, DB is the only pseudo-repair of itself.

Proof: Consider a database instance DB', such that $DB \neq DB'$ and $DB \curvearrowright DB'$. Then, there is at least one atom $p(\bar{a}) \in \Delta(DB, DB')$. Let us suppose, without loss of generality, that $p(\bar{a}) \in (DB - DB')$. Then, there must be a ground instance of a constraint $\phi = \bigvee_{i=1}^{m} \neg p_i(\bar{a}_i) \vee \bigvee_{l=1}^{r} q_l(\bar{b}_l) \vee \phi$ in IC such that $DB \not\models_\Sigma \phi$ and $p(\bar{a})$ is $p_i(\bar{a}_i)$, for some $1 \leq i \leq m$. This is a contradiction, since DB is consistent with respect to IC. Hence, it is not possible that, if $DB'' \neq DB$, the pair (DB, DB'') satisfies the relation $\mathcal{PS}$.

Consider now $DB = DB'$. Since a single database instance always satisfies the plausibility relation, and given $DB = DB' \models_\Sigma IC$, it is the case that DB' is a pseudo-repair of DB with respect to IC. $\square$

Example 12. Consider the instance $DB = \{p(a)\}$ and the set inclusion dependency $\forall x\big(p(x) \rightarrow q(x)\big)$. Then, instance $DB' = \{q(a)\}$ is a pseudo-repair of DB with respect to IC, as the chain DB,DB' satisfy the plausibility relation and $DB' \models_\Sigma IC$. However, DB' is not a repair of DB with respect to IC because it is not minimal. The genuine repairs $DB'_1 = \emptyset$ and $DB'_2 = \{p(\bar{a}), q(\bar{a})\}$ are also pseudo-repairs, because the chains DB,DB'_1 and DB,DB'_2 satisfy the plausibility relation. $\square$

We can say that a pseudo-repair DB' of a database instance DB is an instance that satisfies the ICs, such that every insertion or deletion performed on DB to obtain DB' is made in order to satisfy a constraint. However, as the previous example shows, a pseudo-repair does not necessarily differ in a minimal way from the original instance. The next lemma shows that every repair of a database instance is also a pseudo-repair of the same instance.

Lemma 10. Consider two database instances DB and DB' and a set of integrity constraints IC. If DB' is a repair of DB with respect to IC, then it is also a pseudo-repair of DB with respect to IC.

Proof: First of all, $DB' \models_\Sigma IC$. If $DB = DB'$, the chain DB satisfies the plausibility relation and DB' is pseudo-repair. If $DB \neq DB'$, consider the chain $DB_0, \cdots, DB_n$, where $DB_0 = DB$ and, for $1 \leq j \leq n$, $DB_j = (DB_{j-1} \cup \{q(\bar{b}) \in (DB' - DB) \mid$ there is a ground instance of some constraint $\phi = \bigvee_{i=1}^m \neg p_i(\bar{a}_i) \vee \bigvee_{l=1}^r q_l(\bar{b}_l) \vee \varphi$ in IC, such that $DB_{j-1} \not\models_\Sigma \phi$ and $q(\bar{b})$ is $q_l(\bar{b}_l)$, for $1 \leq l \leq r\}) - \{p(\bar{a}) \in (DB - DB') \mid$ there is a ground instance of some constraint $\phi = \bigvee_{i=1}^m \neg p_i(\bar{a}_i) \vee \bigvee_{l=1}^r q_l(\bar{b}_l) \vee \varphi$ in IC, such that $DB_{j-1} \not\models_\Sigma \phi$ and $p(\bar{a})$ is $p_i(\bar{a}_i)$, for $1 \leq i \leq m\}$. The chain stops when, for some $i \geq 1$, $DB_{i+1} = DB_i$, that is when the least fixpoint of the operator is reached. Given DB' and DB are database instances, the construction of the chain here shown is always finite.

We will first show show that $DB_n = DB'$. Let us suppose first there is an atom $p(\bar{a}) \in DB_n$ and $p(\bar{a}) \notin DB'$. Therefore, two possibilities arise: either $p(\bar{a}) \in (DB' - DB)$, meaning $p(\bar{a}) \in DB'$, a contradiction, or $p(\bar{a}) \in DB$ and $p(\bar{a}) \notin (DB - DB')$, meaning again that $p(\bar{a}) \in DB'$, a contradiction. Then, $DB_n \subseteq DB'$. We will prove now that $DB_n \models_\Sigma IC$. If this was not true, there would be a ground instance of some constraint $\phi = \bigvee_{i=1}^m \neg p_i(\bar{a}_i) \vee \bigvee_{l=1}^r q_l(\bar{b}_l) \vee \varphi$ in IC, such that $DB_n \not\models_\Sigma \phi$. But $DB' \models_\Sigma \phi$. Then, four possibilities arise:

1. There is an element $p(\bar{a}) \in (DB - DB')$, such that $p(\bar{a})$ is $p_i(\bar{a}_i)$, for some $1 \leq i \leq m$. Hence, $DB_n \neq DB_{n+1}$, a contradiction.

2. There is an element $q(\bar{b}) \in (DB' - DB)$, such that $q(\bar{b})$ is $q_l(\bar{b}_l)$, for some $1 \leq l \leq r$. Hence, $DB_n \neq DB_{n+1}$, a contradiction.

3. There is an element $p(\bar{a})$, such that $p(\bar{a}) \notin DB$, $p(\bar{a}) \notin DB'$ and $p(\bar{a})$ is $p_i(\bar{a}_i)$, for some $1 \leq i \leq m$. Then, $p(\bar{a}) \notin DB_n$. Hence, $DB_n \models_\Sigma \phi$, a contradiction.

4. There is an element $q(\bar{b})$, such that $q(\bar{b}) \in DB$, $q(\bar{b}) \in DB'$ and $q(\bar{b})$ is $q_l(\bar{b}_l)$, for some $1 \leq l \leq r$. Then, $q(\bar{b}) \in DB_n$. Hence, $DB_n \models_\Sigma \phi$, a contradiction.

Then, if $DB' \not\subseteq DB_n$, it would be the case that $DB_n \subsetneqq DB'$ and $DB_n \models_\Sigma IC$. A contradiction, given that DB' is a repair of the original instance. Therefore, $DB_n = DB'$, since $DB' \subseteq DB_n$ and $DB_n \subseteq DB'$.

We will now prove that $DB_{j-1} \curvearrowright DB_j$, for $1 \leq j \leq n$. Let us suppose there is a ground instance ϕ of a constraint in IC, such that $DB_{j-1} \not\models_\Sigma \phi$ and $DB_j \not\models_\Sigma \phi$, for $1 \leq j \leq n$. Then, by the construction of the instances in the chain, ϕ is not going to be satisfied by any other instance in the chain moving upwards. In particular, DB_n does not satisfy φ, which is a contradiction, since $DB_n = DB'$ and DB' is a repair of the original instance. Of course, by the way the instances in the chain were constructed, every insertion or deletion of an atom in one step of the chain is justified in terms of the violation of some constraint by the first instance in that step.

Since an atom $p(\bar{a})$ cannot belong to both $(DB' - DB)$ and $(DB - DB')$, it is true that if it is inserted in a step of the chain it cannot be deleted later, and viceversa. Then, the chain $DB_0, \cdots, DB_n$ satisfies the plausibility relation and $DB_n = DB'$ is a pseudo-repair of the original instance. $\square$

The following results show the one to one correspondence between coherent minimal plausible $\star$-models of the program $\Pi(DB, IC)$ and the repairs of the original instance.

Lemma 11 shows that the possibilities in definition 14 are mutually exclusive when considering coherent and plausible models of $\Pi(DB, IC)$. This is, a plausible and coherent model of $\Pi(DB, IC)$ can be univocally interpreted.

Lemma 11. If $\mathcal{M}$ is a coherent plausible $\star$-model of $\Pi(DB, IC)$, then for a database predicate p and ground tuple $\overline{a}$, exactly one of the following cases holds:

- $p(\bar{a}, \mathbf{t_d})$ and $p(\bar{a}, \mathbf{t^\star})$ belong to $\mathcal{M}$, and no other $p(\bar{a}, v)$, for v an annotation value, belongs to $\mathcal{M}$.
- $p(\bar{a}, \mathbf{t_d})$, $p(\bar{a}, \mathbf{t^\star})$, $p(\bar{a}, \mathbf{f_a})$ and $p(\bar{a}, \mathbf{f^\star})$ belong to $\mathcal{M}$, and no other $p(\bar{a}, v)$, for v an annotation value, belongs to $\mathcal{M}$.
- $p(\bar{a}, \mathbf{t_a})$ and $p(\bar{a}, \mathbf{t^\star})$ belong to $\mathcal{M}$, and no other $p(\bar{a}, v)$, for v an annotation value, belongs to $\mathcal{M}$.
- for every annotation value v, $p(\overline{a}, v)$ does not belong to $\mathcal{M}$.

Proof: For an atom $p(\bar{a})$ we have two possibilities:

a. $p(\bar{a}, \mathbf{t_d}) \in \mathcal{M}$. Then, $p(\bar{a}, \mathbf{t}^\star) \in \mathcal{M}$. Two cases are possible now: $p(\bar{a}, \mathbf{f_a}) \in \mathcal{M}$ or $p(\bar{a}, \mathbf{f_a}) \notin \mathcal{M}$ (in this case $p(\bar{a}, \mathbf{t_a}) \notin \mathcal{M}$, since $\mathcal{M}$ is plausible). For the first one we also have $p(\bar{a}, \mathbf{f}^\star) \in \mathcal{M}$ and $p(\bar{a}, \mathbf{t_a}) \notin \mathcal{M}$ (because $\mathcal{M}$ is coherent). This covers the first two items in the lemma.

b. $p(\bar{a}, \mathbf{t_d}) \notin \mathcal{M}$. Since the model is plausible, we have two possibilities now: $p(\bar{a}, \mathbf{t_a}) \in \mathcal{M}$ or $p(\bar{a}, \mathbf{t_a}) \notin \mathcal{M}$ (in this case $p(\bar{a}, \mathbf{f_a}) \notin \mathcal{M}$, since $\mathcal{M}$ is plausible). For the first one we also have $p(\bar{a}, \mathbf{t}^\star) \in \mathcal{M}$ and $p(\bar{a}, \mathbf{f_a}) \notin \mathcal{M}$ (because $\mathcal{M}$ is coherent). This covers the last two items in the lemma.

$\square$

The next lemma shows that some coherent and plausible $\star$-models of the program $\Pi(DB, IC)$ have associated a database instance that satisfies the constraints.

Lemma 12. Let us suppose $\mathcal{M}$ is a coherent plausible $\star$-model of $\Pi(DB, IC)$ and $DB^{\Pi}_{i(\mathcal{M})}$ is finite[4], then $DB^{\Pi}_{i(\mathcal{M})} \models_\Sigma IC$.

Proof: We want to show $DB^{\Pi}_{i(\mathcal{M})} \models_\Sigma \bigvee_{i=1}^{n} \neg p_i(\bar{x_i}) \vee \bigvee_{j=1}^{m} q_j(\bar{y_j}) \vee \varphi$, for every constraint in IC. Since $\mathcal{M}$ is a $\star$-model of $\Pi(DB, IC)$, we have that $\mathcal{M} \models_\star \bigvee_{i=1}^{n} p_i(\bar{x_i}, \mathbf{f_a}) \vee \bigvee_{j=1}^{m} q_j(\bar{y_j}, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^{n} p_i(\bar{x_i}, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^{m} q_j(\bar{y_j}, \mathbf{f}^\star) \wedge \bar{\varphi}$. Then, at least one of the following cases are satisfied:

- $\mathcal{M} \models_\star p_i(\bar{a}, \mathbf{f_a})$. Then, $i(\mathcal{M}) \models p_i(\bar{a}, \mathbf{f}^{\star\star})$ and $p_i(\bar{a}) \notin DB^{\Pi}_{i(\mathcal{M})}$ (by Lemma 11). Hence, $DB^{\Pi}_{i(\mathcal{M})} \models_\Sigma \neg p_i(\bar{a})$. Since the analysis was done for an arbitrary value $\bar{a}$, $DB^{\Pi}_{i(\mathcal{M})} \models_\Sigma \bigvee_{i=1}^{n} \neg p_i(\bar{x_i}) \vee \bigvee_{j=1}^{m} q_j(\bar{y_j}) \vee \varphi$ holds.
- $\mathcal{M} \models_\star q_j(\bar{a}, \mathbf{t_a})$. Then, $i(\mathcal{M}) \models q_j(\bar{a}, \mathbf{t}^{\star\star})$ and $q_j(\bar{a}) \in DB^{\Pi}_{i(\mathcal{M})}$ (by Lemma 11). Hence, $DB^{\Pi}_{i(\mathcal{M})} \models_\Sigma q_j(\bar{a})$. Since the analysis was done for an arbitrary value $\bar{a}$, $DB^{\Pi}_{i(\mathcal{M})} \models_\Sigma \bigvee_{i=1}^{n} \neg p_i(\bar{x_i}) \vee \bigvee_{j=1}^{m} q_j(\bar{y_j}) \vee \varphi$ holds.

[4]Remember that this means that the extensions of the database predicates are finite.

- It is not true that $\mathcal{M} \models_\star \bar{\varphi}$. Then, $\mathcal{M} \models_\star \varphi$. Hence, φ is true, and $DB^{\Pi}_{i(\mathcal{M})} \models_\Sigma \bigvee_{i=1}^{n} \neg p_i(\bar{x_i}) \vee \bigvee_{j=1}^{m} q_j(\bar{y_j}) \vee \varphi$ holds.

- $\mathcal{M} \not\models_\star p_i(\bar{a}, \mathbf{t}^\star)$. As the model is coherent and plausible, just the last item in lemma 11 holds. This means $i(\mathcal{M}) \models p_i(\bar{a}, \mathbf{f}^{\star\star})$, $p_i(\bar{a}) \notin DB^{\Pi}_{i(\mathcal{M})}$ and $DB^{\Pi}_{i(\mathcal{M})} \models_\Sigma \neg p_i(\bar{a})$. Since the analysis was done for an arbitrary value $\bar{a}$, $DB^{\Pi}_{i(\mathcal{M})} \models_\Sigma \bigvee_{i=1}^{n} \neg p_i(\bar{x_i}) \vee \bigvee_{j=1}^{m} q_j(\bar{y_j}) \vee \varphi$ holds.

- $\mathcal{M} \not\models_\star q_j(\bar{a}, \mathbf{f}^\star)$. Hence, $\mathcal{M} \models_\star q_j(\bar{a}, \mathbf{t_d})$, and given the model is coherent and plausible, just the first item in lemma 11 holds. Then, $i(\mathcal{M}) \models q_j(\bar{a}, \mathbf{t}^{\star\star})$, $q_j(\bar{a}) \in DB^{\Pi}_{i(\mathcal{M})}$ and $DB^{\Pi}_{i(\mathcal{M})} \models_\Sigma q_j(\bar{a})$. Since the analysis was done for an arbitrary value $\bar{a}$, $DB^{\Pi}_{i(\mathcal{M})} \models_\Sigma \bigvee_{i=1}^{n} \neg p_i(\bar{x_i}) \vee \bigvee_{j=1}^{m} q_j(\bar{y_j}) \vee \varphi$ holds.

□

Lemma 13 shows that every pseudo-repair of the original instance is associated with a $\star$-model of the program that is coherent and plausible.

Lemma 13. If DB' is a pseudo-repair (definition 20) of DB with respect to IC, then $\mathcal{M}^\star(DB, DB')$ is a coherent plausible $\star$-model of $\Pi(DB, IC)$.

Proof: We first need the following lemma:

Lemma 14. If DB and DB' are two different database instances such that DB' is a pseudo-repair of DB with respect to IC, then:

- For every $p(\bar{a})$ such that $p(\bar{a}) \in DB$ and $p(\bar{a}) \notin DB'$, there is a ground instance of a clause, $\bigvee_{i=1}^{m} p_i(\bar{a_i}, \mathbf{f_a}) \vee \bigvee_{j=1}^{l} q_j(\bar{b_j}, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^{m} p_i(\bar{a_i}, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^{l} q_j(\bar{b_j}, \mathbf{f}^\star) \wedge \bar{\varphi}$ in $\Pi(DB, IC)$, such that $\mathcal{M}^\star(DB, DB') \models_\star \bigwedge_{i=1}^{m} p_i(\bar{a_i}, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^{l} q_j(\bar{b_j}, \mathbf{f}^\star) \wedge \bar{\varphi}$ and $p(\bar{a})$ is $p_i(\bar{a_i})$ for some $1 \leq i \leq m$.

- For every $q(\bar{b})$ such that $q(\bar{b}) \notin DB$ and $q(\bar{b}) \in DB'$, there is a ground instance of a clause, $\bigvee_{i=1}^{m} p_i(\bar{a_i}, \mathbf{f_a}) \vee \bigvee_{j=1}^{l} q_j(\bar{b_j}, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^{m} p_i(\bar{a_i}, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^{l} q_j(\bar{b_j}, \mathbf{f}^\star) \wedge \bar{\varphi}$ in $\Pi(DB, IC)$, such that $\mathcal{M}^\star(DB,\ DB') \models_\star \bigwedge_{i=1}^{m} p_i(\bar{a_i}, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^{l} q_j(\bar{b_j}, \mathbf{f}^\star) \wedge \bar{\varphi}$ and $q(\bar{b})$ is $q_j(\bar{b_j})$ for some $1 \leq j \leq l$.

Proof: Since DB' is a pseudo-repair of DB, and $DB \not\models_\Sigma IC$, we have DB' can be built from DB as a chain $DB = DB_0, \cdots, DB_n = DB'$ $(n \geq 1)$ satisfying the plausibility relation. Then, $p(\bar{a}) \in \Delta(DB_{j-1}, DB_j)$, for $1 \leq j \leq n$. For the first part of the lemma (the other part is symmetrical), this means that there is a constraint instance $\bigvee_{i=1}^m \neg p_i(\bar{a}_i) \vee \bigvee_{l=1}^r q_l(\bar{b}_l) \vee \varphi$ that it is not satisfied by DB_{j-1}, and $p_i(\bar{a}_i) = p(\bar{a})$ for $1 \leq i \leq m$. Then, for every $p_i(\bar{a}_i)$, we have either it is in DB or it belongs to $(DB_k - DB_{k-1})$, for some $k < j-1$. In the first case, $\mathcal{M}^\star(DB, DB')$ satisfies $p_i(\bar{a}_i, \mathbf{t}^\star)$. In the second case, given that for satisfying the plausibility relation an atom inserted in one step cannot be deleted in a posterior step, $p_i(\bar{a}_i) \in DB'$ and $\mathcal{M}^\star(DB, DB')$ satisfies $p_i(\bar{a}_i, \mathbf{t}^\star)$. We also have that, for every $q_l(\bar{b}_l)$, either it is not in the DB or it belongs to $(DB_{k-1} - DB_k)$, for some $k < j-1$. In the first case $\mathcal{M}^\star(DB, DB') \models_\star q_l(\bar{b}_l, \mathbf{f}^\star)$. In the second case, given that for satisfying the plausibility relation an atom deleted in one step cannot be inserted in a posterior step, we have that $\mathcal{M}^\star(DB, DB') \models_\star q_l(\bar{b}_l, \mathbf{f}^\star)$. Finally, we also have φ is false, then $\mathcal{M}^\star(DB,\ DB') \models_\star \bar{\varphi}$. $\square$

We now continue the proof of lemma 13. Since $DB' \models_\Sigma \bigvee_{i=1}^n \neg p_i(\bar{a}_i) \vee \bigvee_{j=1}^m q_j(\bar{b}_j) \vee \varphi$, we have three possibilities to anlyze with respect to the satisfaction of this clause. The first possibility is $DB' \models_\Sigma \neg p_i(\bar{a})$. Then, two cases arise

- $p_i(\bar{a}) \in DB$. Then, $p_i(\bar{a}, \mathbf{f}^\star)$, $p_i(\bar{a}, \mathbf{t_d})$, $p_i(\bar{a}, \mathbf{f_a})$ and $p_i(\bar{a}, \mathbf{t}^\star)$ belong to $\mathcal{M}^\star(DB,\ DB')$, and the program $\Pi(DB, IC)$ contains the following clauses: $p_i(\bar{a}, \mathbf{t_d}) \leftarrow$, $p_i(\bar{a}, \mathbf{t}^\star) \leftarrow p_i(\bar{a}, \mathbf{t_d})$, $p_i(\bar{a}, \mathbf{t}^\star) \leftarrow p_i(\bar{a}, \mathbf{t_a})$, $p_i(\bar{a}, \mathbf{f}^\star) \leftarrow p_i(\bar{a}, \mathbf{f_a})$. Then, all these formulas are satisfied by $\mathcal{M}^\star(DB, DB')$ (satisfaction with respect to $\models_\star$) and $p_i(\bar{a}, \mathbf{t_d})$, $p_i(\bar{a}, \mathbf{t}^\star)$ and $p_i(\bar{a}, \mathbf{f}^\star)$ are shown to be plausible. The program also contains the clause $\bigvee_{i=1}^n p_i(\bar{a}, \mathbf{f_a}) \vee \bigvee_{j=1}^m q_j(\bar{a}, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^n p_i(\bar{a}, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^m q_j(\bar{a}, \mathbf{f}^\star) \wedge \bar{\varphi}$, which is satisfied since $p_i(\bar{a}, \mathbf{f_a})$ belongs to $\mathcal{M}^\star(DB, DB')$. It rests to show that $p_i(\bar{a}, \mathbf{f_a})$ is plausible, but this is precisely what Lemma 14 states.

- $p_i(\bar{a}) \notin DB$. Then, for every annotation value v, $p_i(\overline{a_i}, v) \notin \mathcal{M}^\star(DB, DB')$. Then, $p_i(\bar{a}, \mathbf{t}^\star) \leftarrow p_i(\bar{a}, \mathbf{t_d})$, $p_i(\bar{a}, \mathbf{t}^\star) \leftarrow p_i(\bar{a}, \mathbf{t_a})$, $p_i(\bar{a}, \mathbf{f}^\star) \leftarrow p_i(\bar{a}, \mathbf{f_a})$ are in $\Pi(DB, IC)$. All these are trivially satisfied because the falsity of the body. The clause $\bigvee_{i=1}^n p_i(\bar{a}, \mathbf{f_a}) \vee \bigvee_{j=1}^m q_j(\bar{a}, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^n p_i(\bar{a}, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^m q_j(\bar{a}, \mathbf{f}^\star) \wedge \bar{\varphi}$ is also in $\Pi(DB, IC)$, and it is trivially satisfied since $p_i(\bar{a}, \mathbf{t}^\star) \notin \mathcal{M}^\star(DB, DB')$.

The second possibility is $DB' \models_\Sigma q_j(\bar{a})$. The following cases arise:

- $q_j(\bar{a}) \in DB$. Then, $\mathcal{M}^\star(DB, DB')$ contains $q_j(\bar{a}, \mathbf{t_d})$ and $q_j(\bar{a}, \mathbf{t}^\star)$, and the program $\Pi(DB, IC)$ contains the formulas $q_j(\bar{a}, \mathbf{t_d}) \leftarrow$, $q_j(\bar{a}, \mathbf{t}^\star) \leftarrow q_j(\bar{a}, \mathbf{t_d})$, $q_j(\bar{a}, \mathbf{t}^\star) \leftarrow q_j(\bar{a}, \mathbf{t_a})$, $q_j(\bar{a}, \mathbf{f}^\star) \leftarrow q_j(\bar{a}, \mathbf{f_a})$. $\mathcal{M}^\star(DB, DB')$ satisfies all these clauses (satisfaction with respect to $\models_\star$), and $q_j(\bar{a}, \mathbf{t_d})$ and $q_j(\bar{a}, \mathbf{t}^\star)$ are shown to be plausible. The clause $\bigvee_{i=1}^n p_i(\bar{a}, \mathbf{f_a}) \vee \bigvee_{j=1}^m q_j(\bar{a}, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^n p_i(\bar{a}, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^m q_j(\bar{a}, \mathbf{f}^\star) \wedge \bar{\varphi}$ is also in the program, and is satisfied trivially since it holds that $\mathcal{M}^\star(DB, DB') \not\models_\star q_j(\bar{a}, \mathbf{f}^\star)$.

- $q_j(\bar{a}) \notin DB$. Then, $q_j(\bar{a}, \mathbf{t_a})$ and $q_j(\bar{a}, \mathbf{t}^\star)$ are in $\mathcal{M}^\star(DB, DB')$, and the following formulas are in $\Pi(DB, IC)$: $q_j(\bar{a}, \mathbf{t}^\star) \leftarrow q_j(\bar{a}, \mathbf{t_d})$, $q_j(\bar{a}, \mathbf{t}^\star) \leftarrow q_j(\bar{a}, \mathbf{t_a})$, $q_j(\bar{a}, \mathbf{f}^\star) \leftarrow q_j(\bar{a}, \mathbf{f_a})$. These are satisfied by $\mathcal{M}^\star(DB, DB')$, and $q_j(\bar{a}, \mathbf{t}^\star)$ is plausible. The program also contains the clause $\bigvee_{i=1}^n p_i(\bar{a}, \mathbf{f_a}) \vee \bigvee_{j=1}^m q_j(\bar{a}, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^n p_i(\bar{a}, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^m q_j(\bar{a}, \mathbf{f}^\star) \wedge \bar{\varphi}$, which is satisfied as $q_j(\bar{a}, \mathbf{t_a})$ belongs to $\mathcal{M}^\star(DB, DB')$. It rests to show $q_j(\bar{a}, \mathbf{t_a})$ is plausible, but this is exactly what Lemma 14 states.

The third possibility is $DB' \models_\Sigma \varphi$. Then, φ is true. The program $\Pi(DB, IC)$ contains the clause $\bigvee_{i=1}^n p_i(\bar{a}, \mathbf{f_a}) \vee \bigvee_{j=1}^m q_j(\bar{a}, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^n p_i(\bar{a}, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^m q_j(\bar{a}, \mathbf{f}^\star) \wedge \bar{\varphi}$, which is satisfied since $\mathcal{M}^\star(DB, DB') \not\models_\star \bar{\varphi}$.

Since the analysis was done for an arbitrary value $\bar{a}$, then $\mathcal{M}^\star(DB, DB')$ is a plausible $\star$-model of $\Pi(DB, IC)$. Obviously, it is also coherent, as $\mathcal{M}^\star(DB, DB')$ was defined for not containting both $p(\bar{a}, \mathbf{f_a})$ and $p(\bar{a}, \mathbf{t_a})$. □

Theorem 3. If $\mathcal{M}$ is a coherent, minimal and 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, all the repairs are obtained in this way.

Proof: From Propositions 4 and 5 below. □

Propositions 4 and 5 differ from lemmas 12 and 13 in that only a subset of the pseudo-repairs (the repairs), and a subset of the coherent plausible $\star$-models (the minimal under set inclusion) are considered.

Proposition 4. If $\mathcal{M}$ is a coherent, minimal and 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.

Proof: From Lemma 12, we have that $DB^{\Pi}_{i(\mathcal{M})} \models_{\Sigma} IC$. We just have to show minimality. Let us suppose there is a database instance $DB^{\star}$, such that $DB^{\star} \models_{\Sigma} IC$ and $\Delta(DB, DB^{\star}) \subsetneqq \Delta(DB, DB^{\Pi}_{i(\mathcal{M})})$.

We will show $\mathcal{M}^{\star}(DB, DB^{\star})$ is a $\star$-model of $\Pi(DB, IC)$. Since $DB^{\star} \models_{\Sigma} \bigvee_{i=1}^{n} \neg p_i(\bar{a}_i) \vee \bigvee_{j=1}^{m} q_j(\bar{b_j}) \vee \varphi$, we have three possibilities to analyze with respect to the satisfaction of this clause. The first possibility is $DB' \models_{\Sigma} \neg p_i(\bar{a})$. Then, two cases arise

- $p_i(\bar{a}) \in DB$. Then, $p_i(\bar{a}, \mathbf{f}^{\star})$, $p_i(\bar{a}, \mathbf{t_d})$, $p_i(\bar{a}, \mathbf{f_a})$ and $p_i(\bar{a}, \mathbf{t}^{\star})$ belong to $\mathcal{M}^{\star}(DB, DB^{\star})$, and the program $\Pi(DB, IC)$ contains the following clauses: $p_i(\bar{a}, \mathbf{t_d}) \leftarrow$, $p_i(\bar{a}, \mathbf{t}^{\star}) \leftarrow p_i(\bar{a}, \mathbf{t_d})$, $p_i(\bar{a}, \mathbf{t}^{\star}) \leftarrow p_i(\bar{a}, \mathbf{t_a})$, $p_i(\bar{a}, \mathbf{f}^{\star}) \leftarrow p_i(\bar{a}, \mathbf{f_a})$. Then, all these formulas are satisfied by $\mathcal{M}^{\star}(DB, DB^{\star})$ (satisfaction with respect to $\models_{\star}$). The program also contains the clause $\bigvee_{i=1}^{n} p_i(\bar{a}, \mathbf{f_a}) \vee \bigvee_{j=1}^{m} q_j(\bar{a}, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^{n} p_i(\bar{a}, \mathbf{t}^{\star}) \wedge \bigwedge_{j=1}^{m} q_j(\bar{a}, \mathbf{f}^{\star}) \wedge \bar{\varphi}$, which is satisfied since $p_i(\bar{a}, \mathbf{f_a})$ belongs to $\mathcal{M}^{\star}(DB, DB^{\star})$.

- $p_i(\bar{a}) \notin DB$. Then, for every annotation value v, $p_i(\overline{a_i}, v) \notin \mathcal{M}^{\star}(DB, DB^{\star})$. Then, $p_i(\bar{a}, \mathbf{t}^{\star}) \leftarrow p_i(\bar{a}, \mathbf{t_d})$, $p_i(\bar{a}, \mathbf{t}^{\star}) \leftarrow p_i(\bar{a}, \mathbf{t_a})$, $p_i(\bar{a}, \mathbf{f}^{\star}) \leftarrow p_i(\bar{a}, \mathbf{f_a})$ are in $\Pi(DB, IC)$. All these are trivially satisfied because the falsity of the body. The clause $\bigvee_{i=1}^{n} p_i(\bar{a}, \mathbf{f_a}) \vee \bigvee_{j=1}^{m} q_j(\bar{a}, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^{n} p_i(\bar{a}, \mathbf{t}^{\star}) \wedge \bigwedge_{j=1}^{m} q_j(\bar{a}, \mathbf{f}^{\star}) \wedge \bar{\varphi}$ is also in $\Pi(DB, IC)$, and it is trivially satisfied since $p_i(\bar{a}, \mathbf{t}^{\star}) \notin \mathcal{M}^{\star}(DB, DB^{\star})$.

The second possibility is $DB' \models_{\Sigma} q_j(\bar{a})$. The following cases arise:

- $q_j(\bar{a}) \in DB$. Then, $\mathcal{M}^{\star}(DB, DB^{\star})$ contains $q_j(\bar{a}, \mathbf{t_d})$ and $q_j(\bar{a}, \mathbf{t}^{\star})$, and the program $\Pi(DB, IC)$ contains the formulas $q_j(\bar{a}, \mathbf{t_d}) \leftarrow$, $q_j(\bar{a}, \mathbf{t}^{\star}) \leftarrow q_j(\bar{a}, \mathbf{t_d})$, $q_j(\bar{a}, \mathbf{t}^{\star}) \leftarrow q_j(\bar{a}, \mathbf{t_a})$, $q_j(\bar{a}, \mathbf{f}^{\star}) \leftarrow q_j(\bar{a}, \mathbf{f_a})$. The structure $\mathcal{M}^{\star}(DB, DB^{\star})$ satisfies all these clauses (satisfaction with respect to $\models_{\star}$). The clause $\bigvee_{i=1}^{n} p_i(\bar{a}, \mathbf{f_a}) \vee \bigvee_{j=1}^{m} q_j(\bar{a}, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^{n} p_i(\bar{a}, \mathbf{t}^{\star}) \wedge \bigwedge_{j=1}^{m} q_j(\bar{a}, \mathbf{f}^{\star}) \wedge \bar{\varphi}$ is also in the program, and is satisfied trivially since it holds that $\mathcal{M}^{\star}(DB, DB^{\star}) \not\models_{\star} q_j(\bar{a}, \mathbf{f}^{\star})$.

- $q_j(\bar{a}) \notin DB$. Then, $q_j(\bar{a}, \mathbf{t_a})$ and $q_j(\bar{a}, \mathbf{t}^\star)$ are in $\mathcal{M}^\star(DB, DB^\star)$, and the following formulas are in $\Pi(DB, IC)$: $q_j(\bar{a}, \mathbf{t}^\star) \leftarrow q_j(\bar{a}, \mathbf{t_d})$, $q_j(\bar{a}, \mathbf{t}^\star) \leftarrow q_j(\bar{a}, \mathbf{t_a})$, $q_j(\bar{a}, \mathbf{f}^\star) \leftarrow q_j(\bar{a}, \mathbf{f_a})$. These are satisfied by $\mathcal{M}^\star(DB, DB^\star)$. Also in the program is the clause $\bigvee_{i=1}^n p_i(\bar{a}, \mathbf{f_a}) \vee \bigvee_{j=1}^m q_j(\bar{a}, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^n p_i(\bar{a}, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^m q_j(\bar{a}, \mathbf{f}^\star) \wedge \bar{\varphi}$, which is satisfied since $q_j(\bar{a}, \mathbf{t_a})$ belongs to $\mathcal{M}^\star(DB, DB^\star)$.

The third possibility is $DB' \models_\Sigma \varphi$. Then, φ is true. The program $\Pi(DB, IC)$ contains the clause $\bigvee_{i=1}^n p_i(\bar{a}, \mathbf{f_a}) \vee \bigvee_{j=1}^m q_j(\bar{a}, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^n p_i(\bar{a}, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^m q_j(\bar{a}, \mathbf{f}^\star) \wedge \bar{\varphi}$, which is satisfied since $\mathcal{M}^\star(DB, DB^\star) \not\models_\star \bar{\varphi}$.

The previous part of the proof shows that $\mathcal{M}^\star(DB, DB^\star)$ is a model of $\Pi(DB, IC)$. We are going to prove now it is strictly contained in $\mathcal{M}$. By lemma 11, for an atom $p(\bar{a})$ just four cases are possible in a coherent and plausible $\star$-model of $\Pi(DB, IC)$. This fact will be used in the rest of the proof.

We will first show $\mathcal{M}^\star(DB, DB^\star) \subseteq \mathcal{M}$. Let us suppose just $p(\bar{a}, \mathbf{t}^\star)$ and $p(\bar{a}, \mathbf{t_d})$ belong to $\mathcal{M}^\star(DB, DB^\star)$. Then $p(\bar{a}) \in DB$ and $p(\bar{a}) \in DB^\star$. Since, $p(\bar{a}) \notin \Delta(DB, DB^\star)$, we have two possibilities. The first one is $p(\bar{a}) \notin \Delta(DB, DB^\Pi_{i(\mathcal{M})})$. Then, $p(\bar{a}, \mathbf{t}^\star)$ and $p(\bar{a}, \mathbf{t_d})$ also belong to $\mathcal{M}$. The second possibility is that $p(\bar{a}) \in \Delta(DB, DB^\Pi_{i(\mathcal{M})})$. Again, $p(\bar{a}, \mathbf{t}^\star)$ and $p(\bar{a}, \mathbf{t_d})$ belong to $\mathcal{M}$.

Let us suppose now that, for every annotation value v, $p(\overline{a}, v)$ does not belong to $\mathcal{M}^\star(DB, DB^\star)$. Then, as the empty set is a subset of every other set, we have our fact trivially satisfied.

Consider now just $p(\bar{a}, \mathbf{t}^\star)$, $p(\bar{a}, \mathbf{t_d})$, $p(\bar{a}, \mathbf{f_a})$ and $p(\bar{a}, \mathbf{f}^\star)$ are in the Herbrand structure $\mathcal{M}^\star(DB, DB^\star)$. Then, $p(\bar{a}) \in DB$ and $p(\bar{a}) \notin DB^\star$. Hence, $p(\bar{a}) \in \Delta(DB, DB^\star)$, and due to our assumption $p(\bar{a}) \in \Delta(DB, DB^\Pi_{i(\mathcal{M})})$. Therefore, $p(\bar{a}, \mathbf{t}^\star)$, $p(\bar{a}, \mathbf{t_d})$, $p(\bar{a}, \mathbf{f_a})$ and $p(\bar{a}, \mathbf{f}^\star)$ belong to $\mathcal{M}$.

Finally, we will suppose just $p(\bar{a}, \mathbf{t_a})$ and $p(\bar{a}, \mathbf{t}^\star)$ are in $\mathcal{M}^\star(DB, DB^\star)$. Then, $p(\bar{a}) \notin DB$ and $p(\bar{a}) \in DB^\star$. Hence, $p(\bar{a}) \in \Delta(DB, DB^\star)$, and due to our assumption $p(\bar{a}) \in \Delta(DB, DB^\Pi_{i(\mathcal{M})})$. Therefore, $p(\bar{a}, \mathbf{t}^\star)$ and $p(\bar{a}, \mathbf{t_a})$ belong to $\mathcal{M}$.

We will now show $\mathcal{M}^\star(DB, DB^\star) \subsetneqq \mathcal{M}$. We have assumed there is an element of $\Delta(DB, DB^\Pi_{i(\mathcal{M})})$ that is not an element of $\Delta(DB, DB^\star)$. Thus, for some element $p(\bar{a})$, either $p(\bar{a}) \in DB$, $p(\bar{a}) \in DB^\star$ and $p(\bar{a}) \notin DB^{\Pi}_{i(\mathcal{M})}$, or $p(\bar{a}) \notin DB$, $p(\bar{a}) \notin DB^\star$ and $p(\bar{a}) \in DB^{\Pi}_{i(\mathcal{M})}$. For the first one we have $\mathcal{M}^\star(DB, DB^\star)$ satisfies $p(\bar{a}, \mathbf{t_d})$ and $p(\bar{a}, \mathbf{t}^\star)$, and $\mathcal{M}$ satisfies $p(\bar{a}, \mathbf{t_d})$ and $p(\bar{a}, \mathbf{t}^\star)$, but it also satisfies $p(\bar{a}, \mathbf{f_a})$ and $p(\bar{a}, \mathbf{f}^\star)$. In the second one, $\mathcal{M}^\star(DB, DB^\star)$ does not satisfy any fact related to $p(\bar{a})$ and $\mathcal{M}$ satisfies $p(\bar{a}, \mathbf{t_a})$ and $p(\bar{a}, \mathbf{t}^\star)$.

Then, $\mathcal{M}$ is not a minimal model; a contradiction. □

Proposition 5. If DB' is a repair of DB with respect to IC, then $\mathcal{M}^\star(DB, DB')$ is a coherent, minimal and plausible $\star$-model of $\Pi(DB, IC)$.

Proof: By lemmas 10 and 13 we have $\mathcal{M}^\star(DB, DB')$ is a coherent plausible $\star$-model of $\Pi(DB, IC)$. We just have to show it is minimal. Let us suppose first there exists a $\star$-model $\mathcal{M}$ of $\Pi(DB, IC)$ such that $\mathcal{M} \subsetneqq \mathcal{M}^\star(DB, DB')$ and $\mathcal{M}$ is plausible (it is also coherent since it is contained in $\mathcal{M}^\star(DB, DB')$). We will prove DB' is not minimal.

Let us suppose $p(\bar{a}) \in \Delta(DB, DB^{\Pi}_{i(\mathcal{M})})$. Then, either $p(\bar{a}) \in DB$ and $p(\bar{a}) \notin DB^{\Pi}_{i(\mathcal{M})}$ or $p(\bar{a}) \notin DB$ and $p(\bar{a}) \in DB^{\Pi}_{i(\mathcal{M})}$. In the first case, $p(\bar{a}, \mathbf{t_d})$, $p(\bar{a}, \mathbf{t}^\star)$, $p(\bar{a}, \mathbf{f_a})$ and $p(\bar{a}, \mathbf{f}^\star)$ are in $\mathcal{M}$ (by lemma 11, since $\mathcal{M}$ is coherent and plausible). By our assumption these are also in $\mathcal{M}^\star(DB, DB')$. Hence, $p(\bar{a}) \in \Delta(DB, DB')$. In the second case, $p(\bar{a}, \mathbf{t_a})$ and $p(\bar{a}, \mathbf{t}^\star)$ are in $\mathcal{M}$ (by lemma 11, since $\mathcal{M}$ is coherent and plausible). By our assumption these are also in $\mathcal{M}^\star(DB, DB')$. Hence, $p(\bar{a}) \in \Delta(DB, DB')$ and it has been proved that $\Delta(DB, DB^{\Pi}_{i(\mathcal{M})}) \subseteq \Delta(DB, DB')$.

We will now prove $\Delta(DB, DB^{\Pi}_{i(\mathcal{M})}) \subsetneqq \Delta(DB, DB')$. We know for some fact $p(\bar{a})$ there is an element related to it which is in $\mathcal{M}^\star(DB, DB')$ and which is not in $\mathcal{M}$. One possible case is $p(\bar{a}, \mathbf{f_a})$ and $p(\bar{a}, \mathbf{f}^\star)$ are in $\mathcal{M}^\star(DB, DB')$ and not in $\mathcal{M}$. Then, $p(\bar{a}) \in \Delta(DB, DB')$, but $p(\bar{a}) \notin \Delta(DB, DB^{\Pi}_{i(\mathcal{M})})$. The other possible case $p(\bar{a}, \mathbf{t_a})$ and $p(\bar{a}, \mathbf{t}^\star)$ are in $\mathcal{M}^\star(DB, DB')$ and not in $\mathcal{M}$. Then, $p(\bar{a}) \in \Delta(DB, DB')$, but $p(\bar{a}) \notin \Delta(DB, DB^{\Pi}_{i(\mathcal{M})})$.

By Lemma 12, we have $DB^{\Pi}_{i(\mathcal{M})} \models_\Sigma IC$ (since $\mathcal{M}$ is coherent and plausible). Also, $DB^{\Pi}_{i(\mathcal{M})}$ is finite. This contradicts our fact that DB' is a repair.

Let us suppose now there is a $\star$-model $\mathcal{M}$ of $\Pi(DB, IC)$, such that $\mathcal{M} \subsetneqq \mathcal{M}^\star(DB, DB')$, but $\mathcal{M}$ is not plausible (it must be coherent anyway). If we delete every atom in $\mathcal{M}$ which is not plausible, a model $\mathcal{M}'$ of $\Pi(DB, IC)$ is obtained which is plausible and coherent. Then, the argument above can be used to obtain a contradiction. $\square$

5.2 The Interpretation Program

The interpretation of the models specified in definition 14 could be obtained by adding to the program, for every predicate p, the following interpretation clauses:

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

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

Now, the facts with annotations other than $\mathbf{t^{\star\star}}$ and $\mathbf{f^{\star\star}}$ are not deleted from the model, but this is not a problem if the attention is concentrated in the double-starred annotations. It can be seen that the atoms in a model $\mathcal{M}$ with annotation value $\mathbf{t^{\star\star}}$, when the previous clauses are inserted to the program, are exactly those with annotation value $\mathbf{t^{\star\star}}$ in the intrepretation $i(\mathcal{M})$ of that model. Then, if for any model $\mathcal{M}$ of a program $\Pi(DB, IC)$ extended with the previous clauses it is defined that:

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

the following re-statement of theorem 3 can be given:

Theorem 4. If $\mathcal{M}$ is a coherent minimal plausible $\star$-model of $\Pi(DB, IC)$ extended with interpretation clauses, and $DB^{\Pi}_{\mathcal{M}}$ is finite, then $DB^{\Pi}_{\mathcal{M}}$ is a repair of DB with respect to IC. Furthermore, all the repairs are obtained in this way.

The program, now extended with interpretation clauses, will continue being denoted by $\Pi(DB, IC)$.

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 12). In order to compute from the program using a stable model semantics for disjunctive programs, a new program $\Pi^{\neg}(DB, IC)$ is presented, obtained from the original one by adding the clause $p(\bar{x}, \mathbf{f^\star}) \leftarrow \ \mathit{not}\ p(\bar{x}, \mathbf{t_d})$ that gives an account of the closed world assumption.

From now on, $\mathcal{M}^\star(DB, DB')$ will be redefined with the following statements:

- If $p(\bar{a}) \notin DB$ and $p(\bar{a}) \notin DB'$, $p(\bar{a}, \mathbf{f^\star}) \in I_P$.
- If $p(\bar{a}) \notin DB$ and $p(\bar{a}) \in DB'$, then $p(\bar{a}, \mathbf{f^\star})$, $p(\bar{a}, \mathbf{t_a})$ and $p(\bar{a}, \mathbf{t^\star}) \in I_P$.

The other two cases in definition 16 do not change.

The following results will show there is a one to one correspondence between some stable models of $\Pi^{\neg}(DB, IC)$ and the repairs of the original instance.

Definition 21. A model $\mathcal{M}$ of a disjunctive program P is stable iff $\mathcal{M}$ is a minimal model of $P^{\mathcal{M}}$, where $P^{\mathcal{M}}$ is defined as $\{A_1, \ldots, A_k \leftarrow B_1, \ldots, B_n \mid A_1, \ldots, A_k \leftarrow B_1, \ldots, B_n, notC_1, \ldots, notC_m$ is a ground instance of a clause in P and, for every $1 \leq i \leq m$, $\mathcal{M} \not\models C_i\}$ □

The next lemma shows that the possibilities in definition 14 are mutually exclusive when $\mathcal{M}$ is a coherent and plausible model of $(\Pi^{\neg}(DB, IC))^{\mathcal{M}}$.

Lemma 15. If $\mathcal{M}$ is a coherent plausible model of $(\Pi^{\neg}(DB, IC))^{\mathcal{M}}$, then exactly one of the following cases holds:

- $p(\bar{a}, \mathbf{t_d})$ and $p(\bar{a}, \mathbf{t^\star})$ belong to $\mathcal{M}$, and no other $p(\bar{a}, v)$, for v an annotation value, belongs to $\mathcal{M}$.
- $p(\bar{a}, \mathbf{t_d})$, $p(\bar{a}, \mathbf{t^\star})$, $p(\bar{a}, \mathbf{f_a})$ and $p(\bar{a}, \mathbf{f^\star})$ belong to $\mathcal{M}$, and no other $p(\bar{a}, v)$, for v an annotation value, belongs to $\mathcal{M}$.

- $p(\bar{a}, \mathbf{t_a})$, $p(\bar{a}, \mathbf{f}^\star)$ and $p(\bar{a}, \mathbf{t}^\star)$ belong to $\mathcal{M}$, and no other $p(\bar{a}, v)$, for v an annotation value, belongs to $\mathcal{M}$.
- $p(\bar{a}, \mathbf{f}^\star)$ belongs to $\mathcal{M}$, and no other $p(\bar{a}, v)$, for v an annotation value, belongs to $\mathcal{M}$.

Proof: Similar to that of lemma 11. Just the last two cases are different, but this is due to the inclusion of the negative information of the database explicitly. $\square$

The next lemma shows that if $\mathcal{M}$ is a coherent plausible model of the program $(\Pi^\neg(DB, IC))^\mathcal{M}$, such that $\mathcal{M}$ represents a finite database instance, then that instance satisfies the constraints.

Lemma 16. Let us suppose $\mathcal{M}$ is a coherent plausible model of $(\Pi^\neg(DB, IC))^\mathcal{M}$ and $DB^\Pi_{i(\mathcal{M})}$ is finite, then $DB^\Pi_{i(\mathcal{M})} \models_\Sigma IC$.

Proof: We want to show $DB^\Pi_{i(\mathcal{M})} \models_\Sigma \bigvee_{i=1}^n \neg p_i(\bar{x}_i) \vee \bigvee_{j=1}^m q_j(\bar{y}_j) \vee \varphi$, for every constraint in IC. Since $\mathcal{M}$ is a model of $(\Pi^\neg(DB, IC))^\mathcal{M}$, we have that $\mathcal{M} \models \bigvee_{i=1}^n p_i(\bar{x}_i, \mathbf{f_a}) \vee \bigvee_{j=1}^m q_j(\bar{y}_j, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^n p_i(\bar{x}_i, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^m q_j(\bar{y}_j, \mathbf{f}^\star) \wedge \bar{\varphi}$. Then, at least one of the following cases is satisfied:

- $\mathcal{M} \models p_i(\bar{a}, \mathbf{f_a})$. Then, $i(\mathcal{M}) \models p_i(\bar{a}, \mathbf{f}^{\star\star})$ and $p(\bar{a}) \notin DB^\Pi_{i(\mathcal{M})}$ (by lemma 15). Hence, $DB^\Pi_{i(\mathcal{M})} \models_\Sigma \neg p_i(\bar{a})$. Since the analysis was done for an arbitrary value $\bar{a}$, $DB^\Pi_{i(\mathcal{M})} \models_\Sigma \bigvee_{i=1}^n \neg p_i(\bar{x}_i) \vee \bigvee_{j=1}^m q_j(\bar{y}_j) \vee \varphi$ holds.
- $\mathcal{M} \models q_j(\bar{a}, \mathbf{t_a})$. It is symmetrical to the previous one.
- It is not true that $\mathcal{M} \models \bar{\varphi}$. Then $\mathcal{M} \models \varphi$. Hence, φ is true, and $DB^\Pi_{i(\mathcal{M})} \models_\Sigma \bigvee_{i=1}^n \neg p_i(\bar{x}_i) \vee \bigvee_{j=1}^m q_j(\bar{y}_j) \vee \varphi$ holds.
- $\mathcal{M} \not\models p_i(\bar{a}, \mathbf{t}^\star)$. As the model is coherent and plausible, just the last item in lemma 15 holds. This means $i(\mathcal{M}) \models p_i(\bar{a}, \mathbf{f}^{\star\star})$, $p_i(\bar{a}) \notin DB^\Pi_{i(\mathcal{M})}$ and $DB^\Pi_{i(\mathcal{M})} \models_\Sigma \neg p_i(\bar{a})$. Since the analysis was done for an arbitrary value $\bar{a}$, $DB^\Pi_{i(\mathcal{M})} \models_\Sigma \bigvee_{i=1}^n \neg p_i(\bar{x}_i) \vee \bigvee_{j=1}^m q_j(\bar{y}_j) \vee \varphi$ holds.

- $\mathcal{M} \not\models q_j(\bar{a}, \mathbf{f}^\star)$. Given the model is coherent and plausible, just the first item in lemma 15 holds. Then, $i(\mathcal{M}) \models q_j(\bar{a}, \mathbf{t}^{\star\star})$, $q_j(\bar{a}) \in DB^{\Pi}_{i(\mathcal{M})}$ and $DB^{\Pi}_{i(\mathcal{M})} \models_\Sigma q_j(\bar{a})$. Since the analysis was done for an arbitrary value $\bar{a}$, $DB^{\Pi}_{i(\mathcal{M})} \models_\Sigma \bigvee_{i=1}^{n} \neg p_i(\bar{x_i}) \vee \bigvee_{j=1}^{m} q_j(\bar{y_j}) \vee \varphi$ holds.

$\square$

The next lemma shows that every pseudo-repair of the original instance has associated a structure $\mathcal{M}$, such that $\mathcal{M}$ is a coherent and plausible model of $(\Pi^{\neg}(DB, IC))^{\mathcal{M}}$.

Lemma 17. If DB' is a pseudo-repair (definition 20) of DB with respect to IC, then $\mathcal{M}^\star(DB, DB')$ is a coherent plausible model of $(\Pi^{\neg}(DB, IC))^{\mathcal{M}^\star(DB,DB')}$.

Proof: We first need the following lemma:

Lemma 18. If DB and DB' are two different database instances such that DB' is a pseudo-repair of DB with respect to IC, then:

- For every $p(\bar{a})$ such that $p(\bar{a}) \in DB$ and $p(\bar{a}) \notin DB'$, there is a ground instance of a clause, $\bigvee_{i=1}^{m} p_i(\bar{a_i}, \mathbf{f_a}) \vee \bigvee_{j=1}^{l} q_j(\bar{b_j}, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^{m} p_i(\bar{a_i}, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^{l} q_j(\bar{b_j}, \mathbf{f}^\star) \wedge \bar{\varphi}$ in $(\Pi^{\neg}(DB, IC))^{\mathcal{M}^\star(DB,DB')}$, such that $\mathcal{M}^\star(DB, DB') \models \bigwedge_{i=1}^{m} p_i(\bar{a_i}, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^{l} q_j(\bar{b_j}, \mathbf{f}^\star) \wedge \bar{\varphi}$ and $p(\bar{a})$ is $p_i(\bar{a_i})$ for some $1 \leq i \leq m$.
- For every $q(\bar{b})$ such that $q(\bar{b}) \notin DB$ and $q(\bar{b}) \in DB'$, there is a ground instance of a clause, $\bigvee_{i=1}^{m} p_i(\bar{a_i}, \mathbf{f_a}) \vee \bigvee_{j=1}^{l} q_j(\bar{b_j}, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^{m} p_i(\bar{a_i}, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^{l} q_j(\bar{b_j}, \mathbf{f}^\star) \wedge \bar{\varphi}$ in $(\Pi^{\neg}(DB, IC))^{\mathcal{M}^\star(DB,DB')}$, such that $\mathcal{M}^\star(DB, DB') \models \bigwedge_{i=1}^{m} p_i(\bar{a_i}, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^{l} q_j(\bar{b_j}, \mathbf{f}^\star) \wedge \bar{\varphi}$and $q(\bar{b})$ is $q_j(\bar{b_j})$ for some $1 \leq j \leq l$.

Proof: Exactly that of lemma 10, since the programs $\Pi(DB, IC)$ and $\Pi^{\neg}(DB, IC)$ have exactly the same rules for representing the constraints. $\square$

We now continue the proof of lemma 17. We wll prove first $\mathcal{M}(DB, DB')$ is a coherent plausible model of $(\Pi^{\neg}(DB, IC))^{\mathcal{M}^\star(DB,DB')}$. Since $DB' \models_\Sigma \bigvee_{i=1}^{n} \neg p_i(\bar{a_i})$

$\vee \bigvee_{j=1}^{m} q_j(\bar{b_j}) \vee \varphi$, we have three possibilities to analyze with respect to the satisfaction of this clause. The first possibility is $DB' \models_\Sigma \neg p_i(\bar{a})$. Then, two cases arise

- $p_i(\bar{a}) \in DB$. Then, the $p_i(\bar{a}, \mathbf{f}^\star)$, $p_i(\bar{a}, \mathbf{t_d})$, $p_i(\bar{a}, \mathbf{f_a})$ and $p_i(\bar{a}, \mathbf{t}^\star)$ belong to $\mathcal{M}^\star(DB, DB')$, and the program $(\Pi^\neg(DB, IC))^{\mathcal{M}^\star(DB,DB')}$ contains the following: $p_i(\bar{a}, \mathbf{t_d}) \leftarrow, p_i(\bar{a}, \mathbf{t}^\star) \leftarrow p_i(\bar{a}, \mathbf{t_d}), p_i(\bar{a}, \mathbf{t}^\star) \leftarrow p_i(\bar{a}, \mathbf{t_a}), p_i(\bar{a}, \mathbf{f}^\star) \leftarrow p_i(\bar{a}, \mathbf{f_a})$. Then, all these formulas are satisfied by $\mathcal{M}^\star(DB, DB')$ and $p_i(\bar{a}, \mathbf{t_d})$, $p_i(\bar{a}, \mathbf{t}^\star)$ and $p_i(\bar{a}, \mathbf{f}^\star)$ are shown to be plausible. The program also contains the clause $\bigvee_{i=1}^{n} p_i(\bar{a}, \mathbf{f_a}) \vee \bigvee_{j=1}^{m} q_j(\bar{a}, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^{n} p_i(\bar{a}, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^{m} q_j(\bar{a}, \mathbf{f}^\star) \wedge \bar{\varphi}$, which is satisfied since $p_i(\bar{a}, \mathbf{f_a})$ belongs to $\mathcal{M}^\star(DB, DB')$. It rests to show that $p_i(\bar{a}, \mathbf{f_a})$ is plausible. But this is precisely what Lemma 18 states.

- $p_i(\bar{a}) \notin DB$. Just $p(\bar{a}, \mathbf{f}^\star)$ belongs to $\mathcal{M}^\star(DB, DB')$. Then, $p_i(\bar{a}, \mathbf{t}^\star) \leftarrow p_i(\bar{a}, \mathbf{t_d})$, $p_i(\bar{a}, \mathbf{t}^\star) \leftarrow p_i(\bar{a}, \mathbf{t_a})$, $p_i(\bar{a}, \mathbf{f}^\star) \leftarrow p_i(\bar{a}, \mathbf{f_a})$ and $p(\bar{a}, \mathbf{f}^\star) \leftarrow$ are in the program $(\Pi^\neg(DB, IC))^{\mathcal{M}^\star(DB,DB')}$. All these are satisfied and $p(\bar{a}, \mathbf{f}^\star)$ is shown to be plausible. The program also contains the clause $\bigvee_{i=1}^{n} p_i(\bar{a}, \mathbf{f_a}) \vee \bigvee_{j=1}^{m} q_j(\bar{a}, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^{n} p_i(\bar{a}, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^{m} q_j(\bar{a}, \mathbf{f}^\star) \wedge \bar{\varphi}$, which is trivially satisfied since $p_i(\bar{a}, \mathbf{t}^\star) \notin \mathcal{M}^\star(DB, DB')$.

The second possibility is $DB' \models_\Sigma q_j(\bar{a})$. The following cases arise:

- $q_j(\bar{a}) \in DB$. Then, $\mathcal{M}^\star(DB, DB')$ contains $q_j(\bar{a}, \mathbf{t_d})$ and $q_j(\bar{a}, \mathbf{t}^\star)$, and the program $(\Pi^\neg(DB, IC))^{\mathcal{M}^\star(DB,DB')}$ contains the formulas $q_j(\bar{a}, \mathbf{t_d}) \leftarrow$, $q_j(\bar{a}, \mathbf{t}^\star) \leftarrow q_j(\bar{a}, \mathbf{t_d})$, $q_j(\bar{a}, \mathbf{t}^\star) \leftarrow q_j(\bar{a}, \mathbf{t_a})$, $q_j(\bar{a}, \mathbf{f}^\star) \leftarrow q_j(\bar{a}, \mathbf{f_a})$. All these are satisfied by $\mathcal{M}^\star(DB, DB')$, and $q_j(\bar{a}, \mathbf{t_d})$ and $q_j(\bar{a}, \mathbf{t}^\star)$ are shown to be plausible. The clause $\bigvee_{i=1}^{n} p_i(\bar{a}, \mathbf{f_a}) \vee \bigvee_{j=1}^{m} q_j(\bar{a}, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^{n} p_i(\bar{a}, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^{m} q_j(\bar{a}, \mathbf{f}^\star) \wedge \bar{\varphi}$ is also in the program, and is satisfied trivially since $\mathcal{M}^\star(DB, DB') \not\models q_j(\bar{a}, \mathbf{f}^\star)$.

- $q_j(\bar{a}) \notin DB$. Then, $q_j(\bar{a}, \mathbf{f}^\star)$, $q_j(\bar{a}, \mathbf{t_a})$ and $q_j(\bar{a}, \mathbf{t}^\star)$ are in $\mathcal{M}^\star(DB, DB')$, and the following formulas are in $(\Pi^\neg(DB, IC))^{\mathcal{M}^\star(DB,DB')}$: $q_j(\bar{a}, \mathbf{f}^\star) \leftarrow$, $q_j(\bar{a}, \mathbf{t}^\star) \leftarrow q_j(\bar{a}, \mathbf{t_d})$, $q_j(\bar{a}, \mathbf{t}^\star) \leftarrow q_j(\bar{a}, \mathbf{t_a})$ and $q_j(\bar{a}, \mathbf{f}^\star) \leftarrow q_j(\bar{a}, \mathbf{f_a})$. These are satisfied by $\mathcal{M}^\star(DB, DB')$, and $q_j(\bar{a}, \mathbf{t}^\star)$ and $q_j(\bar{a}, \mathbf{f}^\star)$ are plausible. The program also contains the clause $\bigvee_{i=1}^{n} p_i(\bar{a}, \mathbf{f_a}) \vee \bigvee_{j=1}^{m} q_j(\bar{a}, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^{n} p_i(\bar{a}, \mathbf{t}^\star) \wedge \bigwedge_{j=1}^{m} q_j(\bar{a}, \mathbf{f}^\star) \wedge \bar{\varphi}$, which is satisfied as $q_j(\bar{a}, \mathbf{t_a})$ belongs to $\mathcal{M}^\star(DB, DB')$. It rests to show $q_j(\bar{a}, \mathbf{t_a})$ is plausible, but this is exactly what Lemma 18 states.

The third possibility is $DB' \models_\Sigma \varphi$. Then, φ is true. The clause $\bigvee_{i=1}^n p_i(\bar{a}, \mathbf{f_a}) \vee \bigvee_{j=1}^m q_j(\bar{a}, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^n p_i(\bar{a}, \mathbf{t^\star}) \wedge \bigwedge_{j=1}^m q_j(\bar{a}, \mathbf{f^\star}) \wedge \bar{\varphi}$ is in $(\Pi^\neg(DB, IC))^{\mathcal{M}^\star(DB,DB')}$. This clause is satisfied since $\mathcal{M}^\star(DB, DB') \not\models \bar{\varphi}$.

As the analysis was done for an arbitrary value $\bar{a}$, it holds that the Herbrand structure $\mathcal{M}^\star(DB, DB')$ is a plausible model of $(\Pi^\neg(DB, IC))^{\mathcal{M}^\star(DB,DB')}$. Obviously, it is also coherent, since $\mathcal{M}^\star(DB, DB')$ was defined for not containig both $p(\bar{a}, \mathbf{t_a})$ and $p(\bar{a}, \mathbf{f_a})$ at the same time. $\square$

Note that in the following theorem, the notion of plausibility is no longer treated. However, it can be proved (lemma 19) that the stable models of a disjunctive program are exactly those models that are plausible and minimal.

Theorem 5. If $\mathcal{M}$ is a coherent stable model of $\Pi^\neg(DB, IC)$, and $DB_{\mathcal{M}}$ is finite, then $DB_{\mathcal{M}}$ is a repair of DB with respect to IC. Furthermore, the repairs obtained in this way are all the repairs of DB.

Proof: We will first show that the stable models of the program are the models that are plausible and minimal. The theorem then follows from propositions 6 and 7 below. $\square$

Lemma 19. Consider a database instance DB and a set IC of constraints. A model $\mathcal{M}$ of the program $\Pi^\neg(DB, IC)$ is stable iff it is plausible and minimal.

Proof: A model $\mathcal{M}$ of $\Pi^\neg(DB, IC)$ is stable iff it is a minimal model of $\Pi^\neg(DB, IC)^{\mathcal{M}}$ iff it is a minimal model of every clause in the program $\Pi^\neg(DB, IC)$ which is not of the form $p(\bar{x}, \mathbf{f^\star}) \leftarrow$ *not* $p(\bar{x}, \mathbf{t_d})$ and it is a model of the clause $p(\bar{a}, \mathbf{f^\star}) \leftarrow$, for every $p(\bar{a}) \notin DB$. This is equivalent to be a minimal and plausible model of the program $\Pi^\neg(DB, IC)$. $\square$

Proposition 6. If $\mathcal{M}$ is a coherent, minimal and plausible model of $(\Pi^\neg(DB, IC))^{\mathcal{M}}$, and $DB^\Pi_{i(\mathcal{M})}$ is finite, then $DB^\Pi_{i(\mathcal{M})}$ is a repair of DB with respect to IC.

Proof: From Lemma 16, we have $DB^{\Pi}_{i(\mathcal{M})} \models_{\Sigma} IC$ We just have to show minimality. Let us suppose there is a database instance $DB^{\star}$, such that $DB^{\star} \models_{\Sigma} IC$ and $\Delta(DB, DB^{\star}) \subsetneqq \Delta(DB, DB^{\Pi}_{i(\mathcal{M})})$. The Herbrand structure $\mathcal{M}^{\star}(DB, DB^{\star})$ is a model of $(\Pi^{\neg}(DB, IC))^{\mathcal{M}^{\star}(DB,DB^{\star})}$ by the same reason given in the proof of proposition 4. Then, it is also a coherent plausible model of $(\Pi^{\neg}(DB, IC))^{\mathcal{M}}$. We will first show $\mathcal{M}^{\star}(DB, DB^{\star}) \subseteq \mathcal{M}$. By lemma 15, for an atom $p(\bar{a})$ just four cases are possible in a coherent and plausible model of $\Pi^{\neg}(DB, IC)$. This fact will be used in the rest of the proof.

Let us suppose only $p(\bar{a}, \mathbf{t}^{\star})$ and $p(\bar{a}, \mathbf{t_d})$ belong to $\mathcal{M}^{\star}(DB, DB^{\star})$. Then $p(\bar{a}) \in DB$ and $p(\bar{a}) \in DB^{\star}$. As, $p(\bar{a}) \notin \Delta(DB, DB^{\star})$, we have two possibilities. The first one saying $p(\bar{a}) \notin \Delta(DB, DB^{\Pi}_{i(\mathcal{M})})$. Then, $p(\bar{a}, \mathbf{t}^{\star})$ and $p(\bar{a}, \mathbf{t_d})$ also belong to $\mathcal{M}$. The second one saying $p(\bar{a}) \in \Delta(DB, DB^{\Pi}_{i(\mathcal{M})})$. Again, $p(\bar{a}, \mathbf{t}^{\star})$ and $p(\bar{a}, \mathbf{t_d})$ belong to $\mathcal{M}$.

Let us suppose now, just $p(\bar{a}, \mathbf{f}^{\star})$ belongs to $\mathcal{M}^{\star}(DB, DB^{\star})$. Again, we have two possibilities. The first one says $p(\bar{a}) \notin \Delta(DB, DB^{\Pi}_{i(\mathcal{M})})$. Then, $p(\bar{a}, \mathbf{f}^{\star})$ also belongs to $\mathcal{M}$. The second one says $p(\bar{a}) \in \Delta(DB, DB^{\Pi}_{i(\mathcal{M})})$. Again, $p(\bar{a}, \mathbf{f}^{\star})$ belongs to $\mathcal{M}$.

Let us suppose $p(\bar{a}, \mathbf{t}^{\star})$, $p(\bar{a}, \mathbf{t_d})$, $p(\bar{a}, \mathbf{f_a})$ and $p(\bar{a}, \mathbf{f}^{\star})$ belong to the model $\mathcal{M}^{\star}(DB, DB^{\star})$. Then, $p(\bar{a}) \in DB$ and $p(\bar{a}) \notin DB^{\star}$. Hence, $p(\bar{a}) \in \Delta(DB, DB^{\star})$, and due to our assumption $p(\bar{a}) \in \Delta(DB, DB^{\Pi}_{i(\mathcal{M})})$. Therefore, $p(\bar{a}, \mathbf{t}^{\star})$, $p(\bar{a}, \mathbf{t_d})$, $p(\bar{a}, \mathbf{f_a})$ and $p(\bar{a}, \mathbf{f}^{\star})$ belong to $\mathcal{M}$.

Finally, we will suppose $p(\bar{a}, \mathbf{f}^{\star})$, $p(\bar{a}, \mathbf{t_a})$ and $p(\bar{a}, \mathbf{t}^{\star})$ belong to the model $\mathcal{M}^{\star}(DB, DB^{\star})$. Then, $p(\bar{a}) \notin DB$ and $p(\bar{a}) \in DB^{\star}$. Hence, $p(\bar{a}) \in \Delta(DB, DB^{\star})$, and due to our assumption $p(\bar{a}) \in \Delta(DB, DB^{\Pi}_{i(\mathcal{M})})$. Therefore, $p(\bar{a}, \mathbf{f}^{\star})$, $p(\bar{a}, \mathbf{t}^{\star})$ and $p(\bar{a}, \mathbf{t_a})$ belong to $\mathcal{M}$.

We will now show $\mathcal{M}^{\star}(DB, DB^{\star}) \subsetneqq \mathcal{M}$. We have assumed there is an element of $\Delta(DB, DB^{\Pi}_{i(\mathcal{M})})$ that is not an element of $\Delta(DB, DB^{\star})$. Thus, for some element $p(\bar{a})$, either $p(\bar{a}) \in DB$, $p(\bar{a}) \in DB^{\star}$ and $p(\bar{a}) \notin DB^{\Pi}_{i(\mathcal{M})}$, or $p(\bar{a}) \notin DB$, $p(\bar{a}) \notin DB^{\star}$ and $p(\bar{a}) \in DB^{\Pi}_{i(\mathcal{M})}$. For the first one we have $\mathcal{M}^{\star}(DB, DB^{\star})$ satisfies $p(\bar{a}, \mathbf{t_d})$ and $p(\bar{a}, \mathbf{t}^{\star})$, and $\mathcal{M}$ satisfies $p(\bar{a}, \mathbf{t_d})$ and $p(\bar{a}, \mathbf{t}^{\star})$, but also satisfies $p(\bar{a}, \mathbf{f_a})$

and $p(\bar{a}, \mathbf{f^\star})$. In the second one, $\mathcal{M}^\star(DB, DB^\star)$ satisfies $p(\bar{a}, \mathbf{f^\star})$ and $\mathcal{M}$ satisfies $p(\bar{a}, \mathbf{f^\star})$, but also $p(\bar{a}, \mathbf{t_a})$ and $p(\bar{a}, \mathbf{t^\star})$.

Then, $\mathcal{M}$ is not a minimal model; a contradiction. $\square$

Proposition 7. If DB' is a repair of DB with respect to IC, then $\mathcal{M}^\star(DB, DB')$ is a coherent minimal and plausible model of $(\Pi^\neg(DB, IC))^{\mathcal{M}^\star(DB,DB')}$.

Proof: By Lemma 17 we have $\mathcal{M}^\star(DB, DB')$ is a coherent plausible model of $\Pi^\neg(DB, IC)^{\mathcal{M}^\star(DB,DB')}$. We just have to show it is minimal. Let us suppose first there exists a model $\mathcal{M}$ of $(\Pi^\neg(DB, IC))^{\mathcal{M}^\star(DB,DB')}$ such that it is the case that $\mathcal{M} \subsetneqq \mathcal{M}^\star(DB, DB')$ and $\mathcal{M}$ is plausible (it is also coherent since it is contained in $\mathcal{M}^\star(DB, DB')$). We will prove $\Delta(DB, DB^\Pi_{i(\mathcal{M})}) \subsetneqq \Delta(DB, DB')$. We will begin proving $\Delta(DB, DB^\Pi_{i(\mathcal{M})}) \subseteq \Delta(DB, DB')$.

Let us suppose $p(\bar{a}) \in \Delta(DB, DB^\Pi_{i(\mathcal{M})})$. Then, either $p(\bar{a}) \in DB$ and $p(\bar{a}) \notin DB^\Pi_{i(\mathcal{M})}$ or $p(\bar{a}) \notin DB$ and $p(\bar{a}) \in DB^\Pi_{i(\mathcal{M})}$. In the first case, $p(\bar{a}, \mathbf{t_d})$, $p(\bar{a}, \mathbf{t^\star})$, $p(\bar{a}, \mathbf{f_a})$ and $p(\bar{a}, \mathbf{f^\star})$ are in $\mathcal{M}$. By our assumption these are also in $\mathcal{M}^\star(DB, DB')$. Hence, $p(\bar{a}) \in \Delta(DB, DB')$. In the second case, $p(\bar{a}, \mathbf{f^\star})$, $p(\bar{a}, \mathbf{t_a})$ and $p(\bar{a}, \mathbf{t^\star})$ are in $\mathcal{M}$. By our assumption these are also in $\mathcal{M}^\star(DB, DB')$. Hence, $p(\bar{a}) \in \Delta(DB, DB')$.

We will now prove $\Delta(DB, DB^\Pi_{i(\mathcal{M})}) \subsetneqq \Delta(DB, DB')$. We know for some fact $p(\bar{a})$ there is an element related to it which is in $\mathcal{M}^\star(DB, DB')$ and which is not in $\mathcal{M}$. One possible case is $p(\bar{a}, \mathbf{f_a})$ and $p(\bar{a}, \mathbf{f^\star})$ are in $\mathcal{M}^\star(DB, DB')$ and not in $\mathcal{M}$. Then, $p(\bar{a}) \in \Delta(DB, DB')$, but $p(\bar{a}) \notin \Delta(DB, DB^\Pi_{i(\mathcal{M})})$. The other possible case $p(\bar{a}, \mathbf{t_a})$ and $p(\bar{a}, \mathbf{t^\star})$ are in $\mathcal{M}^\star(DB, DB')$ and not in $\mathcal{M}$. Then, $p(\bar{a}) \in \Delta(DB, DB')$, but $p(\bar{a}) \notin \Delta(DB, DB^\Pi_{i(\mathcal{M})})$.

By Lemma 16, we have $DB^\Pi_{i(\mathcal{M})} \models_\Sigma IC$. Also, $DB^\Pi_{i(\mathcal{M})}$ is finite. This contradicts our fact that DB' is a repair.

Let us suppose now there is a model $\mathcal{M}$ of $(\Pi^\neg(DB, IC))^{\mathcal{M}^\star(DB,DB')}$, such that $\mathcal{M} \subsetneqq \mathcal{M}^\star(DB, DB')$, but $\mathcal{M}$ is not plausible (it must be coherent anyway). If every not plausible atom in $\mathcal{M}$ is deleted, a model $\mathcal{M}'$ of $(\Pi^\neg(DB, IC))^{\mathcal{M}^\star(DB,DB')}$ is obtained which is plausible and coherent. Then, the argument above can be used to obtain a contradiction. $\square$

From theorem 5 it can be seen there is a one to one correspondence between the coherent stable models of $\Pi^{\neg}(DB, IC)$ and the repairs of the original instance. Notice that a classical notion of satisfaction is now being considered in the models of the program $\Pi^{\neg}(DB, IC)$ that produce the repairs of the original instance. This is not true of program $\Pi(DB, IC)$ when a non-classical notion of satisfaction ($\models_{\star}$) is considered.

In consequence, the database repairs could be computed using an implementation of the disjunctive stable models semantics with denial constraints, like *DLV* (Eiter et al. 2000). In the actual case, it is also possible to prune out the models that do not satisfy $\leftarrow p(\bar{x}, \mathbf{t_a}), p(\bar{x}, \mathbf{f_a})$ (the non-coherent ones). Implementation is the main reason why a first order logic representation of the theory through the program was chosen. Most of the applications that compute stable models work with first order logic. *XSB* works with annotated programs, but there is no reason to think that an annotated program would have an easier implementation than this one.

Example 13. Consider the database instance $\{p(a)\}$ that is inconsistent with respect to the set inclusion dependency $\forall x\ (p(x) \rightarrow q(x))$. The program $\Pi^{\neg}(DB, IC)$ contains the following clauses:

a. The following rules do not depend on ICs

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

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

b. A single rule capturing the IC

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

c. Database contents

$$p(a, \mathbf{t_d}) \leftarrow$$

d. The new rules for the closed world assumption

$$p(x, \mathbf{f}^{\star}) \leftarrow \ \mathit{not}\ p(x, \mathbf{t_d}), \qquad q(x, \mathbf{f}^{\star}) \leftarrow \ \mathit{not}\ q(x, \mathbf{t_d})$$

e. Denial constraints for coherence

$$\leftarrow p(\bar{x}, \mathbf{t_a}), p(\bar{x}, \mathbf{f_a}), \qquad \leftarrow q(\bar{x}, \mathbf{t_a}), q(\bar{x}, \mathbf{f_a})$$

f. Rules for interpreting the models

$$p(x, \mathbf{t}^{\star\star}) \leftarrow p(x, \mathbf{t_a}), \qquad p(x, \mathbf{t}^{\star\star}) \leftarrow p(x, \mathbf{t_d}),\ \mathit{not}\ p(x, \mathbf{f_a})$$

$$p(x, \mathbf{f}^{\star\star}) \leftarrow p(x, \mathbf{f_a}), \qquad p(x, \mathbf{f}^{\star\star}) \leftarrow\ \mathit{not}\ p(x, \mathbf{t_d}),\ \mathit{not}\ p(x, \mathbf{t_a})$$

$$q(x, \mathbf{t}^{\star\star}) \leftarrow q(x, \mathbf{t_a}), \qquad q(x, \mathbf{t}^{\star\star}) \leftarrow q(x, \mathbf{t_d}),\ \mathit{not}\ q(x, \mathbf{f_a})$$

$$q(x, \mathbf{f}^{\star\star}) \leftarrow q(x, \mathbf{f_a}), \qquad q(x, \mathbf{f}^{\star\star}) \leftarrow\ \mathit{not}\ q(x, \mathbf{t_d}),\ \mathit{not}\ q(x, \mathbf{t_a}).$$

□

The programs $\Pi^{\neg}(DB, IC)$ with their coherence denials can be run with *DLV*.

Example 14. (example 13 continued) If the stable models of the program in example 13 are computed using DLV, by adding the extensional predicates for representing the domain and the facts of the database:

$$\mathit{domd}(a). \qquad d(a, \mathbf{t_d}).$$

and the clause for representing the database facts as intensional predicates:

$$p(x, \mathbf{t_d}) \leftarrow d(x, \mathbf{t_d}), \mathit{domd}(x).$$

the following is obtained:

$$M_1 = \{\mathit{domd}(a), d(a, \mathbf{t_d}), p(a, \mathbf{t_d}), p(a, \mathbf{t}^\star), q(a, \mathbf{f}^\star), q(a, \mathbf{t_a}), p(a, \mathbf{t}^{\star\star}), q(a, \mathbf{t}^\star), q(a, \mathbf{t}^{\star\star})\}$$

$$M_2 = \{\mathit{domd}(a), d(a, \mathbf{t_d}), p(a, \mathbf{t_d}), p(a, \mathbf{t}^\star), p(a, \mathbf{f}^\star), q(a, \mathbf{f}^\star), p(a, \mathbf{f}^{\star\star}), q(a, \mathbf{f}^{\star\star}), p(a, \mathbf{f_a})\}$$

The first model says, through its atom $q(a, \mathbf{t}^{\star\star})$, that $q(a)$ has to be inserted in the database. The second one, through its atom $p(a, \mathbf{f}^{\star\star})$, that $p(a)$ has to be deleted.

The reason for considering the predicate $domd(x)$ is that it is recommendable for every clause in the program to include it in its body, as this implies the program will be instantiated just in the domain values and not in the annotation values. $\Box$

Example 15. In example 13 the coherence denial constraints did not play any role, because the only models, without those constraints, are exactly the same obtained with them. Since there is only one IC, to repair the database only one repair step is needed, in consequence there is no way to get an incoherent stable model. Let us add to that example a new inclusion dependency, $\forall x\ (q(x) \rightarrow r(x))$, with the same instance. One repair is obtained by inserting $q(a)$, which causes the insertion of $r(a)$. Now the program is as before, but with the additional rules:

$$r(x, \mathbf{f}^\star) \leftarrow r(x, \mathbf{f_a}), \qquad r(x, \mathbf{t}^\star) \leftarrow r(x, \mathbf{t_a}),$$

$$r(x, \mathbf{t}^\star) \leftarrow r(x, \mathbf{t_d}) \qquad r(x, \mathbf{f}^\star) \leftarrow \ \mathit{not}\ r(x, \mathbf{t_d})$$

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

$$r(x, \mathbf{t}^{\star\star}) \leftarrow r(x, \mathbf{t_a}), \qquad r(x, \mathbf{t}^{\star\star}) \leftarrow r(x, \mathbf{t_d}),\ \mathit{not}\ r(x, \mathbf{f_a})$$

$$r(x, \mathbf{f}^{\star\star}) \leftarrow r(x, \mathbf{f_a}), \qquad r(x, \mathbf{f}^{\star\star}) \leftarrow \ \mathit{not}\ r(x, \mathbf{t_d}),\ \mathit{not}\ r(x, \mathbf{t_a})$$

$$\leftarrow r(x, \mathbf{t_a}), r(x, \mathbf{f_a}).$$

After running the program the following models are computed

$$\begin{aligned} M_1 = \{&domd(a), d(a, \mathbf{t_d}), p(a, \mathbf{t_d}), p(a, \mathbf{t}^\star), p(a, \mathbf{f}^\star), q(a, \mathbf{f}^\star), r(a, \mathbf{f}^\star), p(a, \mathbf{f}^{\star\star}),\\ &r(a, \mathbf{f}^{\star\star}), q(a, \mathbf{f}^{\star\star}), p(a, \mathbf{f_a})\} \end{aligned}$$

$$\begin{aligned} M_2 = \{&domd(a), d(a, \mathbf{t_d}), p(a, \mathbf{t_d}), p(a, \mathbf{t}^\star), q(a, \mathbf{f}^\star), r(a, \mathbf{f}^\star), r(a, \mathbf{t_a}), q(a, \mathbf{t_a}),\\ &p(a, \mathbf{t}^{\star\star}), q(a, \mathbf{t}^\star), q(a, \mathbf{t}^{\star\star}), r(a, \mathbf{t}^\star), r(a, \mathbf{t}^{\star\star})\} \end{aligned}$$

Nevertheless, if the coherence denial constraints are omitted, more precisely the one for the table q, a third model is obtained, which is incoherent:

$$M_3 = \{domd(a), d(a, \mathbf{t_d}), p(a, \mathbf{t_d}), p(a, \mathbf{t}^\star), q(a, \mathbf{f}^\star), r(a, \mathbf{f}^\star), r(a, \mathbf{f}^{\star\star}), q(a, \mathbf{f}^{\star\star}), q(a, \mathbf{t_a}), p(a, \mathbf{t}^{\star\star}), q(a, \mathbf{t}^\star), q(a, \mathbf{t}^{\star\star}), q(a, \mathbf{f_a})\}$$

□

It is known (Dantsin et al. 1997) that the complexity of computing the stable models of a disjunctive logic program is Π_2^P-complete in the size of the ground program. For some specific cases, computing consistent answers for non-existential conjunctive queries can be done more efficiently using the well-founded interpretation, that is computed in polynomial time (Arenas et al. 2000b).

As mentioned before, consistent answers can be obtained running a query program together 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.

5.4 Head Cycle Free Programs

From (Ben-Eliyahu et al. 1994) and (Dantsin et al. 1997) it is known that the *Head Cycle Free* (HCF) property of a disjunctive logic program is a sufficient condition for this one to have exactly the same stable models than its *related definite program*. An advantage of this transformation is the fact that the complexity of computing the stable models of a normal definite program is co-NP-complete, *i.e.* probably lower than the complexity of computing stable models for a disjunctive logic program. The related definite program of a disjunctive program is constructed by replacing every clause of the form [5]:

$$\bigvee_{i=1}^{n} b_i \leftarrow \bigwedge_{j=1}^{m} a_j$$

[5]Considering, as in the case of the programs presented here, just positive literals belong to the head of a clause.

by the set of n definite clauses of the form:

$$b_k \leftarrow \bigwedge_{j=1}^{m} a_j, \bigwedge_{i=1}^{n} not\ b_i, \quad 1 \leq k \leq n,\ i \neq k$$

With every program Π a directed graph $\mathcal{DG}_\Pi =< \mathcal{N}, E >$ is associated, such that:

1. every ground predicate of Π is a node in $\mathcal{N}$.
2. there is an arc in E directed from node a to node b iff there is a ground clause r in Π such that a and b are the predicates of a positive literal appearing in the body and the head of that clause, respectively.

A program Π is HCF iff there is no ground clause r in Π such that two predicates occuring in the head of r are in the same cycle of $\mathcal{DG}_\Pi$ (Leone et al. 1997).

Example 16. The program in example 13 satisfies the property of being HCF property, as it can be easily shown by drawing the relevant part of the graph (figure 5.1) (there is no cycle involving $p(x, \mathbf{f_a})$ and $q(x, \mathbf{t_a})$). The definite version of the program in example 13 is exactly the same for every definite clause in that program. The only disjunctive clause present there is the one for representing the IC. Then, it can be transformed in two definite clauses:

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

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

If the stable models of this new program are computed using DLV, exactly those of example 14 are obtained. This coincides with the fact that, for a HCF disjunctive logic program, its stable models can be computed by computing the stable models of its related definite program. □

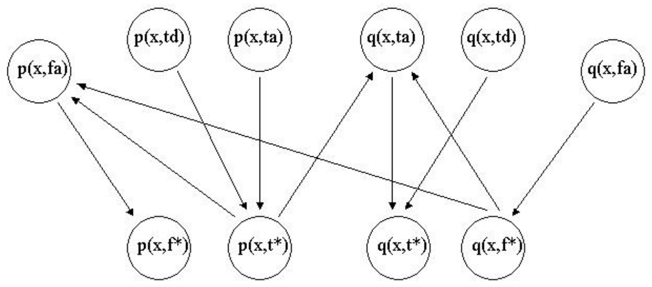

Figure 5.1 Directed graph of program in example 16.

The question if every repair program considered in this work is HCF or has the same stable models of its related definite instance is answered by the following example.

Example 17. Consider the instance $DB = \{p(a), r(a)\}$ and the following set of ICs:

$$\{\neg p(x) \vee \neg r(x) \vee q(x),\ \neg p(x) \vee \neg q(x) \vee \neg r(x),\ \neg r(x) \vee q(x) \vee p(x)\}$$

represented as clauses:

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

$$r(x, \mathbf{f_a}) \vee p(x, \mathbf{f_a}) \vee q(x, \mathbf{f_a}) \leftarrow p(x, \mathbf{t}^\star), q(x, \mathbf{t}^\star), r(x, \mathbf{t}^\star)$$

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

The instance is inconsistent with respect to its constraints. As the program $\Pi^\neg(DB, IC)$ also contains the following clauses:

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

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

part of the directed graph is seen as in figure 5.2.

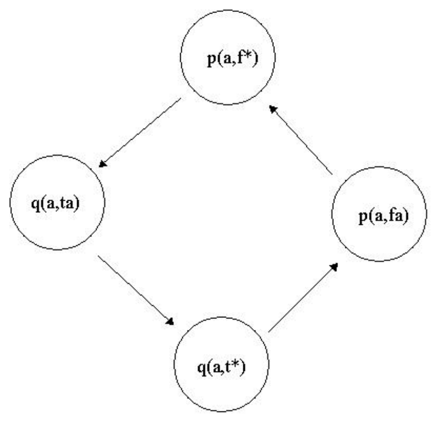

Figure 5.2 Directed graph of program in example 17.

Given $p(x, \mathbf{f_a})$ and $q(x, \mathbf{t_a})$ are in the same cycle and they are both in the head of the first clause, the program under consideration is not HCF. Anyway, since the HCF property is not a necessary condition for both the disjunctive program and its related definite program to have the same stable models, it could still be possible that this happens.

Using DLV for computing the stable models of this program, the following is obtained:

$$M_1 = \{domd(a), d(a, \mathbf{t_d}), e(a, \mathbf{t_d}), p(a, \mathbf{t_d}), p(a, \mathbf{t}^\star), r(a, \mathbf{t_d}), r(a, \mathbf{t}^\star), q(a, \mathbf{f}^\star),$$
$$r(a, \mathbf{f}^\star), r(a, \mathbf{f}^{\star\star}), q(a, \mathbf{f}^{\star\star}), p(a, \mathbf{t}^{\star\star}), r(a, \mathbf{f_a})\}$$

$$M_2 = \{domd(a), d(a, \mathbf{t_d}), e(a, \mathbf{t_d}), p(a, \mathbf{t_d}), p(a, \mathbf{t}^\star), r(a, \mathbf{t_d}), r(a, \mathbf{t}^\star), p(a, \mathbf{f}^\star),$$
$$q(a, \mathbf{f}^\star), p(a, \mathbf{f}^{\star\star}), q(a, \mathbf{t_a}), p(a, \mathbf{f_a}), r(a, \mathbf{t}^{\star\star}), q(a, \mathbf{t}^\star), q(a, \mathbf{t}^{\star\star})\}$$

The related definite program from $\Pi^\neg(DB, IC)$ can be obtained by replacing every disjunctive clause above by the following set of definite clauses:

$$r(x, \mathbf{f_a}) \leftarrow p(x, \mathbf{t}^\star), q(x, \mathbf{f}^\star), r(x, \mathbf{t}^\star),\ not\ p(x, \mathbf{f_a}),\ not\ q(x, \mathbf{t_a})$$

$$p(x, \mathbf{f_a}) \leftarrow p(x, \mathbf{t}^\star), q(x, \mathbf{f}^\star), r(x, \mathbf{t}^\star),\ not\ r(x, \mathbf{f_a}),\ not\ q(x, \mathbf{t_a})$$

$$q(x,\mathbf{t_a}) \leftarrow p(x,\mathbf{t}^\star), q(x,\mathbf{f}^\star), r(x,\mathbf{t}^\star),\ \mathit{not}\ r(x,\mathbf{f_a}),\ \mathit{not}\ q(x,\mathbf{f_a})$$

$$r(x,\mathbf{f_a}) \leftarrow p(x,\mathbf{t}^\star), q(x,\mathbf{t}^\star), r(x,\mathbf{t}^\star),\ \mathit{not}\ p(x,\mathbf{f_a}),\ \mathit{not}\ q(x,\mathbf{f_a})$$

$$p(x,\mathbf{f_a}) \leftarrow p(x,\mathbf{t}^\star), q(x,\mathbf{t}^\star), r(x,\mathbf{t}^\star),\ \mathit{not}\ r(x,\mathbf{f_a}),\ \mathit{not}\ q(x,\mathbf{f_a})$$

$$q(x,\mathbf{f_a}) \leftarrow p(x,\mathbf{t}^\star), q(x,\mathbf{t}^\star), r(x,\mathbf{t}^\star),\ \mathit{not}\ r(x,\mathbf{f_a}),\ \mathit{not}\ p(x,\mathbf{f_a})$$

$$r(x,\mathbf{f_a}) \leftarrow p(x,\mathbf{f}^\star), q(x,\mathbf{f}^\star), r(x,\mathbf{t}^\star),\ \mathit{not}\ p(x,\mathbf{t_a}),\ \mathit{not}\ q(x,\mathbf{t_a})$$

$$p(x,\mathbf{t_a}) \leftarrow p(x,\mathbf{f}^\star), q(x,\mathbf{f}^\star), r(x,\mathbf{t}^\star),\ \mathit{not}\ r(x,\mathbf{f_a}),\ \mathit{not}\ q(x,\mathbf{t_a})$$

$$q(x,\mathbf{t_a}) \leftarrow p(x,\mathbf{f}^\star), q(x,\mathbf{f}^\star), r(x,\mathbf{t}^\star),\ \mathit{not}\ r(x,\mathbf{f_a}),\ \mathit{not}\ p(x,\mathbf{t_a})$$

This program just have one stable model. This is:

$$\begin{aligned}M_1 = \{&\mathit{domd}(a), d(a,\mathbf{t_d}), e(a,\mathbf{t_d}), p(a,\mathbf{t_d}), p(a,\mathbf{t}^\star), r(a,\mathbf{t_d}), r(a,\mathbf{t}^\star), q(a,\mathbf{f}^\star),\\ &r(a,\mathbf{f}^\star), r(a,\mathbf{f}^{\star\star}), q(a,\mathbf{f}^{\star\star}), p(a,\mathbf{t}^{\star\star}), r(a,\mathbf{f_a})\}\end{aligned} \qquad \square$$

It has been shown that not every program considered here can be transformed to its related definite program for obtaining its stable models. Therefore the complexity of the problem cannot be reduced to be co-NP-complete in this manner.

5.5 The Query Program

Given a first order query Q, the consistent answers from DB are looked for. In consequence, those atoms that are simultaneously true in every interpreted coherent stable model of the program $\Pi^\neg(DB, IC)$ are required. 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).

Example 18. (example 7 cont.) The program $\Pi^\neg(DB, IC)$ is the program $\Pi(DB, IC)$ of example 7, but extended with the interpretation program and rules for dealing with negative information:

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

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

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

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

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

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

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

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

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

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

For the query

$$Q(y) : \exists z Book(kafka,y,z),$$

is generated

$$Q^{\star\star}(y) : \exists z Book(kafka,y,z,\mathbf{t}^{\star\star}),$$

that is transformed into the query program

$$\Pi(Q^{\star\star}) : Answer(y) \leftarrow Book(kafka,y,z,\mathbf{t}^{\star\star}).$$

The coherent stable models of $\Pi^{\neg}(DB, IC) \cup \Pi(Q^{\star\star})$ are:

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

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

It can be seen that $y = \mathit{metamorph}$ is a consistent answer to the query. □

VI. PROGRAMS WITH REFERENTIAL ICS

The repair program of section 5.1 was stated for universal ICs. Referential ICs of the form $\forall \bar{x}\big(p(\bar{x}) \rightarrow \exists y(q(\bar{x}', y))\big)$, where $\bar{x}' \subseteq \bar{x}$, are now going to be considered. It is assumed that the variables range over an underlying database domain D, that does not include the value *null*, nevertheless, a RIC can be repaired by insertion of the null value, say $q(\bar{a}, \mathit{null})$, or by elimination in cascade. If the repair is by introduction of *null*, it is assumed that this change will not propagate to other ICs, e.g. a set inclusion dependency like $\forall \bar{x}(q(\bar{x}', y) \rightarrow r(\bar{x}', y))$. The program will not detect such inconsistency.

The program $\Pi(DB, IC)$ is then extended with the following formulas:

$$aux(\bar{x}) \leftarrow q(\bar{x}', y, \mathbf{t_d}),\ \mathit{not}\ q(\bar{x}', y, \mathbf{f_a}). \tag{6.1}$$

$$aux(\bar{x}) \leftarrow q(\bar{x}', y, \mathbf{t_a}). \tag{6.2}$$

$$p(\bar{x}, \mathbf{f_a}) \vee q(\bar{x}', \mathit{null}, \mathbf{t_a}) \leftarrow p(\bar{x}, \mathbf{t}^\star),\ \mathit{not}\ aux(\bar{x}),\ \mathit{not}\ q(\bar{x}', \mathit{null}, \mathbf{t_d}). \tag{6.3}$$

Intuitively, clauses (6.1) and (6.2) detect if the formula $\exists y(q(\bar{a}', y){:}\mathbf{t} \vee q(\bar{a}', y){:}\mathbf{t_a}))$ is satisfied by the model. If this is not the case, and $p(\bar{a}, \mathbf{t}^\star)$ belongs to the model, and $q(\bar{a}', \mathit{null})$ is not in the original instance, i.e. there is a violation of the RIC, then, according to rule (6.3), the repair is done either by deleting $p(\bar{a})$ or inserting $q(\bar{a}', \mathit{null})$.

Notice that here the definition of repair given in section 2 has not been followed, in the sense that repairs are obtained by deletion of tuples or insertion of null values, the usual ways to maintain referential integrity constraints. For example, if the instance is $\{p(\bar{a})\}$ and IC contains only $\forall \bar{x}\big(p(\bar{x}) \rightarrow \exists y q(\bar{x}, y)\big)$, then $\{p(\bar{a}), q(\bar{a}, b)\}$, with $b \in D$, is not a repair, although it would be a repair according to the original definition. In particular, this "repair" will not be captured by the program.

If we insist in keeping the original definition of repair, i.e. allowing $\{p(\bar{a}), q(\bar{a}, b)\}$ to be a repair for every element $b \in D$, clause (6.3) could be replaced by:

$$p(\bar{x}, \mathbf{f_a}) \vee q(\bar{x}', y, \mathbf{t_a}) \leftarrow p(\bar{x}, \mathbf{t^\star}),\ \mathit{not}\ \mathit{aux}(\bar{x}'),\ \mathit{not}\ q(\bar{x}', \mathit{null}, \mathbf{t_d}), \mathit{choice}(\bar{x}', y). \quad (6.4)$$

where $\mathit{choice}(\bar{X}, \bar{Y})$ is a static non-deterministic operator that selects one value for attribute tuple $\bar{Y}$ for each value of the attribute tuple $\bar{X}$ (Giannotti et al. 1997). In equation (6.4), $\mathit{choice}(\bar{x}', y)$ selects one value from the domain. Then, this rule forces the one to one correspondence between stable models and repairs.

Example 19. Consider the database instance $\{p(\bar{a})\}$ and the following set of ICs:

$$\forall x \big(p(x) \rightarrow \exists y q(x, y)\big), \quad \forall x \forall y \big(q(x, y) \rightarrow r(x, y)\big).$$

The program $\Pi^{\neg}(DB, IC)$ is written in DLV as:

Database contents

$$\mathit{domd}(a). \qquad d(a, \mathbf{t_d}). \qquad p(x, \mathbf{t_d}) \leftarrow d(x, td), \mathit{domd}(x).$$

Rules not depending on ICs

$$p(x, \mathbf{f^\star}) \leftarrow\ \mathit{not}\ p(x, \mathbf{t_d}), \mathit{domd}(x).$$

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

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

$$p(x, \mathbf{t^\star}) \leftarrow p(x, \mathbf{t_d}), \mathit{domd}(x).$$

$$q(x, y, \mathbf{f^\star}) \leftarrow\ \mathit{not}\ q(x, y, \mathbf{t_d}), \mathit{domd}(x), \mathit{domd}(y).$$

$$q(x, y, \mathbf{f^\star}) \leftarrow q(x, y, \mathbf{f_a}), \mathit{domd}(x), \mathit{domd}(y).$$

$$q(x, y, \mathbf{t^\star}) \leftarrow q(x, y, \mathbf{t_a}), \mathit{domd}(x), \mathit{domd}(y).$$

$$q(x, y, \mathbf{t^\star}) \leftarrow q(x, y, \mathbf{t_d}), \mathit{domd}(x), \mathit{domd}(y).$$

$$r(x, y, \mathbf{f^\star}) \leftarrow \; not\; r(x, y, \mathbf{t_d}), domd(x), domd(y).$$

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

$$r(x, y, \mathbf{t^\star}) \leftarrow r(x, y, \mathbf{t_a}), domd(x), domd(y).$$

$$r(x, y, \mathbf{t^\star}) \leftarrow r(x, y, \mathbf{t_d}), domd(x), domd(y).$$

Rules for the ICs

$$aux(x) \leftarrow q(x, y, \mathbf{t_d}),\; not\; q(x, y, \mathbf{f_a}), domd(x), domd(y).$$

$$aux(x) \leftarrow q(x, y, \mathbf{t_a}), domd(x), domd(y).$$

$$p(x, \mathbf{f_a}) \vee q(x, null, ta) \leftarrow p(x, \mathbf{t^\star}),\; not\; aux(x),\; not\; q(x, null, \mathbf{t_d}), domd(x).$$

$$q(x, y, \mathbf{f_a}) \vee r(x, y, \mathbf{t_a}) \leftarrow q(x, y, \mathbf{t^\star}), r(x, y, \mathbf{f^\star}), domd(x), domd(y).$$

Rules for interpreting the models

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

$$p(x, \mathbf{t^{\star\star}}) \leftarrow p(x, \mathbf{t_d}),\; not\; p(x, \mathbf{f_a}), domd(x).$$

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

$$p(x, \mathbf{f^{\star\star}}) \leftarrow \; not\; p(x, \mathbf{t_d}),\; not\; p(x, \mathbf{t_a}), domd(x).$$

$$q(x, \mathbf{t^{\star\star}}) \leftarrow q(x, \mathbf{t_a}), domd(x).$$

$$q(x, \mathbf{t^{\star\star}}) \leftarrow q(x, \mathbf{t_d}),\; not\; q(x, \mathbf{f_a}), domd(x).$$

$$q(x, \mathbf{f^{\star\star}}) \leftarrow q(x, \mathbf{f_a}), domd(x).$$

$$q(x, \mathbf{f^{\star\star}}) \leftarrow \; not\; q(x, \mathbf{t_d}),\; not\; q(x, \mathbf{t_a}), domd(x).$$

$$r(x, \mathbf{t^{\star\star}}) \leftarrow r(x, \mathbf{t_a}), domd(x).$$

$$r(x, \mathbf{t^{\star\star}}) \leftarrow r(x, \mathbf{t_d}),\; not\; r(x, \mathbf{f_a}), domd(x).$$

$$r(x, \mathbf{f}^{\star\star}) \leftarrow r(x, \mathbf{f_a}), domd(x).$$

$$r(x, \mathbf{f}^{\star\star}) \leftarrow \ not\ r(x, \mathbf{t_d}),\ not\ r(x, \mathbf{t_a}), domd(x).$$

Rules for interpreting the null values

$$q(x, null, \mathbf{t}^{\star\star}) \leftarrow q(x, null, \mathbf{t_a}).$$

$$q(x, null, \mathbf{t}^{\star\star}) \leftarrow q(x, null, \mathbf{t_d}),\ not\ q(x, null, \mathbf{f_a}).$$

$$r(x, null, \mathbf{t}^{\star\star}) \leftarrow r(x, null, \mathbf{t_a}).$$

$$r(x, null, \mathbf{t}^{\star\star}) \leftarrow r(x, null, \mathbf{t_d}),\ not\ r(x, null, \mathbf{f_a}).$$

Denial constraints

$$\leftarrow p(x, \mathbf{t_a}), p(x, \mathbf{f_a}). \quad \leftarrow q(x, y, \mathbf{t_a}), q(x, y, \mathbf{f_a}). \quad \leftarrow r(x, y, \mathbf{t_a}), r(x, y, \mathbf{f_a}).$$

The models obtained are:

$$M_1 = \{domd(a), d(a, \mathbf{t_d}), p(a, \mathbf{t_d}), p(a, \mathbf{t}^\star), p(a, \mathbf{f}^\star), p(a, \mathbf{f}^{\star\star}), p(a, \mathbf{f_a}), q(a, a, \mathbf{f}^\star), r(a, a, \mathbf{f}^\star), q(a, a, \mathbf{f}^{\star\star}), r(a, a, \mathbf{f}^{\star\star})\}$$

$$M_2 = \{domd(a), d(a, \mathbf{t_d}), p(a, \mathbf{t_d}), p(a, \mathbf{t}^\star), p(a, \mathbf{t}^{\star\star}), q(a, null, \mathbf{t_a}), q(a, a, \mathbf{f}^\star), r(a, a, \mathbf{f}^\star), q(a, a, \mathbf{f}^{\star\star}), r(a, a, \mathbf{f}^{\star\star}), q(a, null, \mathbf{t}^{\star\star})\}$$

corresponding to the database instances $\emptyset$ and $\{p(a),\ q(a, null)\}$. Notice the implementation does not consider the inclusion dependency $\forall x \forall y \big(q(x, y) \rightarrow r(x, y)\big)$ to be violated by the insertion of the tuple $q(a, null)$.

What if the fact $q(a, null)$ is added to the instance? Then, the following clauses are in the program:

$$e(a, null, \mathbf{t_d}). \qquad q(x, null, \mathbf{t_d}) \leftarrow e(x, null, \mathbf{t_d}), domd(x).$$

The only model obtained is

$$M_1 = \{domd(a), d(a, \mathbf{t_d}), e(a, null, \mathbf{t_d}), p(a, \mathbf{t_d}), p(a, \mathbf{t}^\star), q(a, null, \mathbf{t_d}), p(a, \mathbf{t}^{\star\star}), q(a, a, \mathbf{f}^{\star\star}), r(a, a, \mathbf{f}^\star), q(a, a, \mathbf{f}^{\star\star}), r(a, a, \mathbf{f}^{\star\star}), q(a, null, \mathbf{t}^{\star\star})\}$$

That is, the program considers the instance $\{p(a),\ q(a, null)\}$ does not violate the RIC, as its only repair is itself.

$\square$

Let us suppose finally, that the policy of repairing the violation of a RIC just deleting tuples and never adding information to the instance is desirable. Then, equation (6.3) should be written as

$$p(\bar{x}, \mathbf{f_a}) \leftarrow p(\bar{x}, \mathbf{t}^{\star}),\ \ not\ aux(\bar{x}),\ \ not\ q(\bar{x}', null, \mathbf{t_d}).$$

That is, if the RIC is violated, detected by the body of the clause, the fact $p(\bar{a})$ that produces such violation must be eliminated (stated in the head).

VII. RELATED WORK

Closest related works are (Greco et al. 2001) and (Arenas et al. 2000b), that present a general methodology to specify database repairs for universal ICs by means of logic programs with stable model semantics. The programs presented here are much simpler and shorter than those due to the simplicity of its stabilizing clauses. In particular, in (Arenas et al. 2000b) and (Greco et al. 2001) an integrity constraint involving n database predicates has n disjunctive clauses associated. The approach in (Greco et al. 2001) concentrates mainly in producing the set of changes, rather than the repaired instances directly. This implies the program cannot be used directly to obtain consistent answers.

In (Blair et al. 1989; Kifer et al. 1992b; Leach et al. 1996) paraconsistent and annotated logic programs are introduced. 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 work have a completely classical semantics. However, in (Damasio et al. 2000) more classical semantics are proposed for general annotated logic programs (Kifer et al. 1992b), and in (Damasio et al. 1999) some transformation methodologies for paraconsistent logic programs (Blair et al. 1989) are shown that allow assigning to them extensions of classical semantics.

Another approach to database repairs based on logic programming semantics consists of the *revision programs* (Marek et al. 1998). The rules in those programs explicitly declare how to enforce the satisfaction of an integrity constraint, rather than explicitly stating the ICs, e.g.:

$$in(a) \leftarrow in(a_1), \ldots, in(a_k), out(b_1), \ldots, out(b_m)$$

has the intended procedural meaning of inserting the database atom a whenever $a_1, \ldots, a_k$ are in the database, but not $b_1, \ldots, b_m$. Also a declarative, stable model semantics is given to revision programs. Preferences for certain kinds of repair actions can be captured by declaring the corresponding rules in program and omitting rules that could lead to other forms of repairs.

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

In general, the work done with inconsistent databases is very close to the one for incomplete information, *i.e.* databases with null or missing values. A *certain answer* to a query in a probably incomplete database, is an answer that holds in every "model" of the instance, where a "model" could be understood as a notion equivalent to the notion of repair in the context of incompleteness (Lipski 1979). Under this framework it has also been proposed (Reiter 1978) to return *disjunctive answers.* A disjunctive answer to the query $Q(\bar{x})$ is a set of tuples $\{a_1, \ldots, a_m\}$ such that $Q(a_1) \vee \cdots \vee Q(a_m)$ is a certain answer and no proper subset of that set also satisfies this condition. It seems interesting to study this possibility under the context of inconsistency.

VIII. CONCLUSIONS

In this work a general treatment of consistent query answering for first order queries and universal ICs was presented. It was shown, additionally, how to annotate referential ICs, obtaining a correspondence between the minimal models of the annotated first order specification and the database repairs. Then, the problem of consistent query answering as a problem of non-monotonic entailment from the annotated theory was formulated. Classical disjunctive logic programs, where annotations are now arguments, that specify the database repairs in the case of universal ICs were also formulated. In consequence, consistent query answers can be obtained by "running"the program. The semantics of the programs is essentially stable model semantics for disjunctive programs. Finally, an implementation of those programs in DLV and an extension of the programs to consider referential integrity constraints were presented.

Among the relevant advantages of having presented a first order program such that its stable model semantics specifies the repairs of the original instance are:

- For the annotated theory $\mathcal{T}(\mathbf{DB}, \mathbf{IC})$ it is necessary to select its minimal models to obtain the repairs. Although, every coherent stable model of the program $\Pi^{\neg}(DB, IC)$ has associated a repair.
- The theory does not have a deductive system as it is needed to reason with a subset of its models. It would be required a non-monotonic logic, and is usually hard to work with non-monotonic logics. For a logic program there is a clear way to deduce things. Maybe not very efficient but clear.
- After inserting and deleting tuples it is easy to change the program to obtain the new repairs. This can be done just varying the facts the program possess.
- The stable models of a first order disjunctive logic program can be computed with an application like DLV. It is not known any application to compute disjunctive annotated programs.

An important problem that requires much more research by the logic programming and database communities has to do with developing mechanisms for query evaluation from disjunctive logic programs that are guided by the query, hopefully for open queries containing free variables and asking for answers that are sets of tuples. The current alternative is based on finding those ground query atoms that belong to all the stable models once they have been computed via a complete ground instantiation of the original program; obviously not a very appealing mechanism.

Ongoing work considers, among others:

- To find a natural representation of user preferences for choosing some repairs. A consistent preferred answer would be an answer that holds in every preferred repair. Also, it would be interesting to model the preferences in the style of repair policies, *e.g.* the preference for deleting tuples before inserting tuples.
- To study the class of the programs presented here that are HCF. The computation of the semantics of these programs has (probably) a lower complexity than that of disjunctive programs. Also, an implementation in XSB can be done for this kind of programs and its *Oracle* or *ODBC* interface can become useful.
- To study how previously computed repairs can be used for more efficiently computing new repairs, once some information has been added or deleted from the instance or once the integrity constraints have been changed.
- To characterize with a modal logic the notion of consistent query answering. In this work that notion has been specified in the meta-language, as those answers that holds in every coherent stable model of the program. Anyway, if it was wanted to define it in the object language of a modal logic, it would be useful to think in possible worlds as repairs, and to define a modal operator C for representing being true in every possible world.

BIBLIOGRAPHY

[Abiteboul et al. 1995] ABITEBOUL, S.; HULL, R. and VIANU, V. (1995) *Foundations of Databases.* Addison-Wesley, 1995.

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

[Arenas et al. 2000a] ARENAS, M.; BERTOSSI, L. and KIFER, M. (2000) 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. (2000) 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, pp. 27–41, 2000.

[Ben-Eliyahu et al. 1994] BEN-ELIYAHU, R.; and DECHTER, R. (1994) Propositional Semantics for Disjunctive Logic Programs. *Annals of Mathematics in Artificial Intelligence*, 12:53-87, 1994.

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

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

[Damasio et al. 1999] DAMASIO, C. V.; PEREIRA, L.M. and SWIFT, T. (1999) Coherent Well-founded Annotated Logic Programs. In *Proc. LPNMR'99.* Springer Lecture Notes in AI 1730, pp. 262–276, 1999.

[Damasio et al. 2000] DAMASIO, C. V. and PEREIRA, L.M. (1998) A Survey on Paraconsistent Semantics for Extended Logic Programas. In *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, Vol. 2, D.M. Gabbay and Ph. Smets (eds.), Kluwer Academic Publishers, pp. 241–320, 2000.

[Dantsin et al. 1997] DANTSIN, E.; EITER, T.; GOTTLOB, G.; and VORONKOV A. (1997) Complexity and Expressive Power of Logic Programming. In *Proc. Twelfth Annual IEEE Conference on Computational Complexity*, CCC'97, Ulm, Germany, pages 82-101, 1997.

[Eiter et al. 2000] EITER, T.; FABER, W.; LEONE, N.; PFEIFER, G. (2000) Declarative Problem-Solving in DLV. In *Logic-Based Artificial Intelligence*, J. Minker (ed.), Kluwer, pp. 79–103, 2000.

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

[Gelfond et al. 1988] GELFOND, M.; and LIFSCHITZ, V. (1988) 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, pp. 1070–1080, 1988.

[Giannotti et al. 1997] GIANNOTTI, F.; GRECO, S.; SACCÁ, D.; and ZANIOLO C. (1997) Programming with Non-Determinism in Deductive Databases. *Annals of Mathematics and Artificial Intelligence* 19: 97-125, 1997.

[Greco et al. 2001] GRECO, G.; GRECO, S.; and ZUMPANO, E. (2001) 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, pp. 348–364, 2001.

[Kifer et al. 1992a] KIFER, M. and LOZINSKII, E.L. (1992) 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. (1992) Theory of Generalized Annotated Logic Programming and its Applications. *Journal of Logic Programming*, 12(4):335-368, April 1992.

[Leone et al. 1997] LEONE, N.; RULLO, P.; and SCARCELLO, F. (1997) Disjunctive Stable Models: Unfounded Sets, Fixpoint Semantics, and Computation. *Information and Computation*, 135(2):69-112, 1997.

[Lifschitz 1996] LIFSCHITZ, V. (1996) Foundations of Logic Programming. In *Principles of Knowledge Representation*, G. Brewka (ed.), CSLI Publications, pp. 69–127, 1996.

[Lloyd 1987] LLOYD, J.W. (1987) *Foundations of Logic Programming.* Springer Verlag, 1987.

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

[Lipski 1979] LIPSKI, W. (1979) On Semantics Issues Connected with Incomplete Information Databases. *ACM Transactions on Database Systems*, 4(3):262-296, September 1979.

[Lobo et al. 1998] LOBO, J.; MINKER, J. and RAJASEKAR, A. (1998) 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.

[Marek et al. 1998] MAREK, V.W. and TRUSZCZYNSKI, M. (1998) Revision Programming. *Theoretical Computer Science*, 190(2):241–277, 1998.

[Reiter 1978] REITER, R. (1978) Deductive Question Answering on Relational Databases. In *Logic and Data Bases*, H. Gallaire et al. (eds.), pp. 149-178. Plenum, New York, 1978.

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