Repairing Inconsistent Databases: A Model-Theoretic Approach and Abductive Reasoning

Ofer Arieli[1] Marc Denecker[2] Bert Van Nuffelen[2] Maurice Bruynooghe[2]

[1] Department of Computer Science, The Academic College of Tel-Aviv
Antokolski 4, Tel-Aviv 61161, Israel
oarieli@mta.ac.il

[2] Department of Computer Science, The Catholic University of Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
{marcd,bertv,maurice}@cs.kuleuven.ac.be

Abstract. In this paper we consider two points of views to the problem of coherent integration of distributed data. First we give a pure model-theoretic analysis of the possible ways to 'repair' a database. We do so by characterizing the possibilities to 'recover' consistent data from an inconsistent database in terms of those models of the database that exhibit as minimal inconsistent information as reasonably possible. Then we introduce an abductive application to restore the consistency of a given database. This application is based on an abductive solver ($\mathcal{A}$-system) that implements an SLDNFA-resolution procedure, and computes a list of data-facts that should be inserted to the database or retracted from it in order to keep the database consistent. The two approaches for coherent data integration are related by soundness and completeness results.

1 Introduction

Integration of data coming from different databases is a very common, nevertheless nontrivial, task. There are number of different phases involved in this process, the most important of which are the following:

1. Resolving the different ontologies and/or database scheme, setting a single unified schema, and translating the integrity constraints[3] of each database to the new ontology.
2. Resolving contradictions among the integrity constraints of different local databases.
3. Integrating distributed databases w.r.t. the unified set of integrity constraints, computed in the previous phase.

[3] I.e., the rules that represent intentional truths of a database domain.

Each one of the phases mentioned above has its own difficulties and challenges. For instance, we are not aware of any work that gives a complete and robust solution to the problem of the first phase. Most of the formalisms for database integration implicitly assume that all the databases to be integrated have the same ontology, so the first phase is not needed.

The reason for separating the remaining two phases is that integrity constraints represent truths that should be valid in all situations, while a database instance represents an existentional truth, i.e., an actual situation. Consequently, the policy of resolving contradictions among integrity constraints is often different than the one that is applied on database facts, and the former should be applied first.

Despite their different nature, both these phases are based on some formalisms that maintain contradictions and allow to draw plausible conclusions from inconsistent situations. Roughly, there are two approaches to handle this problem:

- *Paraconsistent* formalisms, in which the amalgamated data may remain inconsistent, but the set of conclusions implied by it is not explosive, i.e.: not every fact follows from an inconsistent database. Paraconsistent procedures for integrating data (e.g., [14, 41]) are often based on a paraconsistent reasoning process, such as LFI [13], annotated logics [30, 40], or other non-classical proof systems [5, 37].
- *Coherent* (consistency-base) methods, in which the amalgamated data is revised in order to restore consistency (see, e.g., [6, 8, 11, 25, 31]). In many cases the underlying formalism of these approaches are closely related to the theory of belief revision [1, 23]. In the context of database systems the idea is to construct consistent databases that are "as close as possible" to the original database. These "repaired" instances of the spoiled database correspond to plausible and compact ways of restoring consistency.

In this paper we follow the latter approach, and consider two points of views for the last phase of the process, namely: coherent methods of integrating distributed databases (with the same ontology) w.r.t. a consistent set of integrity constraints. The main difficulty in this process stems from the fact that even when each local database is consistent, the collective information of all the distributed databases may not be consistent anymore. In particular, facts that are specified in a particular database may violate some integrity constraints defined elsewhere, and so it might contradict some elements in the unified set of integrity constraints. Our goal is therefore to find ways to properly "repair" a combined database, and restore its consistency.

One way of viewing this problem is by a model-theoretic analysis that characterizes database repairs in terms of a certain set of models of the inconsistent database (those that, intuitively, minimize the amount of inconsistent information). The other approach is based on abductive reasoning. For this we use an abductive solver ($\mathcal{A}$-system, [27]) that implements SLDNFA-resolution [16, 17] for computing a list of data-facts that should be inserted to the database or retracted from it in order to keep the data consistent. A corresponding application was introduced and described in greater details in [7]. Here we review it in order to keep this paper self contained, and putting our results in the right context. We then show that the abductive process of coherent integration of databases is sound and complete w.r.t. the semantics that is induced by the model theoretic analysis. [4]

2 Coherent integration of databases

In this paper we assume that we have a first-order language L, based on a fixed database schema S, and a fixed domain D. Every element of D has a unique name. A database instance $\mathcal{D}$ consists of atoms in the language L that are instances of the schema S. As such, every instance $\mathcal{D}$ has a finite active domain, which is a subset of D. A *database* is a pair $(\mathcal{D}, \mathcal{IC})$, where $\mathcal{D}$ is a database instance, and $\mathcal{IC}$, the set of *integrity constraints*, is a finite set of formulae in L (assumed to be satisfied by $\mathcal{D}$).

Given a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$, we apply to it the closed word assumption, so only the facts that are explicitly mentioned in $\mathcal{D}$ are considered true. The underlying semantics corresponds, therefore, to minimal Herbrand interpretations.

Definition 1. The *minimal Herbrand model* $\mathcal{H}^{\mathcal{D}}$ of a database instance $\mathcal{D}$ is the model of $\mathcal{D}$ that assigns true to all the ground instances of atomic formulae in $\mathcal{D}$, and false to all the other atoms.

Definition 2. A formula ψ *follows* from a database instance $\mathcal{D}$ (notation: $\mathcal{D} \models \psi$) if the minimal Herbrand model of $\mathcal{D}$ is also a model of ψ.

Definition 3. A database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ is *consistent* if $\mathcal{IC}$ is a classically consistent set, and each formula of it follows from $\mathcal{D}$ (notation: $\mathcal{D} \models \mathcal{IC}$).

Our goal is to integrate n consistent databases, $\mathcal{DB}_i = (\mathcal{D}_i, \mathcal{IC}_i)$, $i = 1, \ldots n$, in such a way that the combined data will contain everything

[4] Due to a lack of space some proofs are reduced or omitted altogether. Full proofs will appear in an extended version of this paper.

that can be deduced from one source of information, without violating any integrity constraint of another source. The idea is to consider the union of the distributed data, and then to restore its consistency. A key notion in this respect is the following:

Definition 4. A *repair* of $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ is a pair $(\mathsf{Insert}, \mathsf{Retract})$ such that (1) $\mathsf{Insert} \cap \mathcal{D} = \emptyset$, (2) $\mathsf{Retract} \subseteq \mathcal{D}$,[5] and (3) $(\mathcal{D} \cup \mathsf{Insert} \setminus \mathsf{Retract},\ \mathcal{IC})$ is a consistent database.

Intuitively, Insert is a set of elements that should be inserted into $\mathcal{D}$ and $\mathsf{Retract}$ is a set of elements that should be removed from $\mathcal{D}$ in order to obtain a consistent database.

Definition 5. A *repaired database* of $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ is a consistent database $(\mathcal{D} \cup \mathsf{Insert} \setminus \mathsf{Retract}\,,\ \mathcal{IC})$, where $(\mathsf{Insert}, \mathsf{Retract})$ is a repair of $\mathcal{DB}$.

As there may be many ways to repair an inconsistent database,[6] it is often convenient to make preferences among the possible repairs, and consider only the most preferred ones. Below are two common preference criteria.

Definition 6. Let $(\mathsf{Insert}, \mathsf{Retract})$ and $(\mathsf{Insert}', \mathsf{Retract}')$ be two repairs.

- *set inclusion preference criterion* : $(\mathsf{Insert}', \mathsf{Retract}') \leq_i (\mathsf{Insert}, \mathsf{Retract})$, if $\mathsf{Insert} \subseteq \mathsf{Insert}'$ and $\mathsf{Retract} \subseteq \mathsf{Retract}'$.
- *cardinality preference criterion*: $(\mathsf{Insert}', \mathsf{Retract}') \leq_c (\mathsf{Insert}, \mathsf{Retract})$ if $|\mathsf{Insert}| + |\mathsf{Retract}| \leq |\mathsf{Insert}'| + |\mathsf{Retract}'|$. [7]

In what follows we assume that $\leq$ is a fixed pre-order that represents some preference criterion on the set of repairs.

Definition 7. A *$\leq$-preferred repair* of $\mathcal{DB}$ is a repair $(\mathsf{Insert}, \mathsf{Retract})$ of $\mathcal{DB}$, s.t. for every repair $(\mathsf{Insert}', \mathsf{Retract}')$ of $\mathcal{DB}$, if $(\mathsf{Insert}, \mathsf{Retract}) \leq (\mathsf{Insert}', \mathsf{Retract}')$ then $(\mathsf{Insert}', \mathsf{Retract}') \leq (\mathsf{Insert}, \mathsf{Retract})$. The set of all the $\leq$-preferred repairs of $\mathcal{DB}$ is denoted by $!(\mathcal{DB}, \leq)$.

Definition 8. A *$\leq$-repaired database* of $\mathcal{DB}$ is a repaired database of $\mathcal{DB}$, constructed from a $\leq$-preferred repair of $\mathcal{DB}$. The set of all the $\leq$-repaired databases of $\mathcal{DB}$ is denoted by

$$\mathcal{R}(\mathcal{DB}, \leq) = \{\, (\mathcal{D} \cup \mathsf{Insert} \setminus \mathsf{Retract}\,,\ \mathcal{IC}) \mid (\mathsf{Insert}, \mathsf{Retract}) \in (\mathcal{DB}, \leq)\,\}.$$

[5] Note that by conditions (1) and (2) it follows that $\mathsf{Insert} \cap \mathsf{Retract} = \emptyset$.

[6] Some of them may be trivial and/or useless. For instance, the inconsistency in $(\mathcal{D}, \mathcal{IC}) = (\{p, q, r\}, \{\neg p\})$ may be removed by deleting every element in $\mathcal{D}$, but this is certainly not the optimal way of restoring consistency in this case.

[7] Set inclusion is also considered in [3, 11, 14, 25]; cardinality is considered, e.g., in [31]

Note that if $\mathcal{DB}$ is consistent, and the preference criterion is a partial order that is monotonic in the total size of the repairs' components (as in Def. 6), then $\mathcal{R}(\mathcal{DB}, \leq) = \{\mathcal{DB}\}$, so there is nothing to repair, as expected.

It is usual to refer to the $\leq$-preferred databases of $\mathcal{DB}$ as the consistent databases that are 'as close as possible' to $\mathcal{DB}$ itself (see, e.g., [3, 14, 31]). Indeed, denote $Th(\mathcal{D}) = \{P(t) \mid \mathcal{D} \models P(t)\}$, where P is a relation name and t is a ground tuple, and let $\mathtt{dist}(\mathcal{D}_1, \mathcal{D}_2)$ be the following set:

$$\mathtt{dist}(\mathcal{D}_1, \mathcal{D}_2) = (Th(\mathcal{D}_1) \setminus Th(\mathcal{D}_2)) \cup (Th(\mathcal{D}_2) \setminus Th(\mathcal{D}_1))$$

It is easy to see that $\mathcal{DB}' = (\mathcal{D}', \mathcal{IC})$ is a $\leq_i$-repaired database of $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$, if the set $\mathtt{dist}(\mathcal{D}', \mathcal{D})$ is minimal (w.r.t. set inclusion) among all the sets of the form $\mathtt{dist}(\mathcal{D}'', \mathcal{D})$, where $\mathcal{D}'' \models \mathcal{IC}$. Similarly, if $\#(S)$ denotes the number of elements in S, then $\mathcal{DB}' = (\mathcal{D}', \mathcal{IC})$ is a $\leq_c$-repaired database of $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$, if $\#(\mathtt{dist}(\mathcal{D}', \mathcal{D})$ is minimal in $\{\#(\mathtt{dist}(\mathcal{D}'', \mathcal{D})) \mid \mathcal{D}'' \models \mathcal{IC}\}$.

Definition 9. For $\mathcal{DB}_i = (\mathcal{D}_i, \mathcal{IC}_i)$, $i = 1, \ldots n$, let $\mathcal{UDB} = (\mathcal{D}, \mathcal{IC})$, where $\mathcal{D} = \bigcup_{i=1}^{n} \mathcal{D}_i$ and $\mathcal{IC} = \bigcup_{i=1}^{n} \mathcal{IC}_i$.

Given n distributed databases and a preference criterion $\leq$, our goal is to compute the set $\mathcal{R}(\mathcal{UDB}, \leq)$ of the $\leq$-repaired databases of $\mathcal{UDB}$ (or to be able to compute, in an efficient way, some elements in this set). Below are test-cases for such database integration. [8] [9]

Example 1. Consider a distributed database with a relation *teaches* of the following scheme: (`course_name`, `teacher_name`). Suppose also that each database contains a single integrity constraint, stating that the same course cannot be taught by two different teachers:

$$\mathcal{IC} = \{\, \forall X \forall Y \forall Z\, (teaches(X, Y) \wedge teaches(X, Z) \rightarrow Y = Z) \,\}.$$

Consider now the following two databases:

$$\mathcal{DB}_1 = (\,\{teaches(c_1, n_1),\ teaches(c_2, n_2)\},\ \mathcal{IC}\,),$$
$$\mathcal{DB}_2 = (\,\{teaches(c_2, n_3)\},\ \mathcal{IC})$$

Clearly, the unified database $\mathcal{DB}_1 \cup \mathcal{DB}_2$ is inconsistent. Its preferred repairs are $(\emptyset, \{teaches(c_2, n_2)\})$ and $(\emptyset, \{teaches(c_2, n_3)\})$. Hence, the two repaired databases are the following:

$$\mathcal{R}_1 = (\,\{teaches(c_1, n_1),\ teaches(c_2, n_2)\},\ \mathcal{IC}\,),$$
$$\mathcal{R}_2 = (\,\{teaches(c_1, n_1),\ teaches(c_2, n_3)\},\ \mathcal{IC}\,).$$

[8] See, e.g., [3, 11, 25] for more discussions on the examples below.

[9] In all the following examples we use set inclusion as the preference criterion. In what follows we shall fix a preference criterion for choosing the "best" repairs and omit its notation whenever possible.

Example 2. Let $\mathcal{D}_1 = \{p(a), p(b)\}, \mathcal{D}_2 = \{q(a), q(c)\}$, and $\mathcal{IC} = \{\forall X(p(X) \rightarrow q(X))\}$. Again, $(\mathcal{D}_1, \emptyset) \cup (\mathcal{D}_2, \mathcal{IC})$ is inconsistent. The corresponding preferred repairs are $(\{q(b)\}, \emptyset)$ and $(\emptyset, \{p(b)\})$. The repaired databases are $\mathcal{R}_1 = (\{p(a), p(b), q(a), q(b), q(c)\}, \mathcal{IC})$ and $\mathcal{R}_2 = (\{p(a), q(a), q(c)\}, \mathcal{IC})$.

3 Database repair – A model-theoretic point of view

In this section we characterize the repairs of a given database in terms of its models. First, we consider arbitrary repairs, and show that they can be represented either by *two-valued models* of the theory of integrity constraints, or by *three-valued models* of the set of integrity constraints and the set of literals, obtained by applying the closed world assumption on the database facts. Then we focus on the most preferred repairs, and show that a certain subset of the three-valued models considered above can be used for characterizing $\leq$-preferred repairs.

Definition 10. Given a valuation ν and a truth value x. Denote:

$$\nu^x = \{p \mid p \text{ is an atomic formula, and } \nu(p) = x\}.$$ [10]

The following two propositions characterize repairs in terms of two-valued structures.

Proposition 1. *Let $(\mathcal{D}, \mathcal{IC})$ be a database and let M be a two-valued model of $\mathcal{IC}$. Let* $\mathsf{Insert} = M^t \setminus \mathcal{D}$ *and* $\mathsf{Retract} = \mathcal{D} \setminus M^t$. *Then* $(\mathsf{Insert}, \mathsf{Retract})$ *is a repair of $(\mathcal{D}, \mathcal{IC})$.*

Proof: The definitions of Insert and Retract immediately imply that $\mathsf{Insert} \cap \mathcal{D} = \emptyset$ and $\mathsf{Retract} \subseteq \mathcal{D}$. For the the last condition in Definition 4, note that in our case $\mathcal{D} \cup \mathsf{Insert} \setminus \mathsf{Retract} = \mathcal{D} \cup (M^t \setminus \mathcal{D}) \setminus (\mathcal{D} \setminus M^t) = M^t$. It follows that M is the least Herbrand model of $\mathcal{D} \cup \mathsf{Insert} \setminus \mathsf{Retract}$ and it is also a model of $\mathcal{IC}$, therefore $\mathcal{D} \cup \mathsf{Insert} \setminus \mathsf{Retract} \models \mathcal{IC}$. □

Proposition 2. *Let* $(\mathsf{Insert}, \mathsf{Retract})$ *be a repair of a database $(\mathcal{D}, \mathcal{IC})$. Then there is a classical model M of $\mathcal{IC}$,*[11] *such that* $\mathsf{Insert} = M^t \setminus \mathcal{D}$ *and* $\mathsf{Retract} = \mathcal{D} \setminus M^t$.

Proof: Consider a valuation M, defined for every atom p as follows:

$$M(p) = \begin{cases} t & \text{if } p \in \mathcal{D} \cup \mathsf{Insert} \setminus \mathsf{Retract}, \\ f & \text{otherwise.} \end{cases}$$

[10] Note, in particular, that $(\mathcal{H}^{\mathcal{D}})^t = \mathcal{D}$.

[11] Recall that we assume that $\mathcal{IC}$ is classically consistent, thus it has classical models.

By its definition, M is a minimal Herbrand model of $\mathcal{D} \cup \mathsf{Insert} \setminus \mathsf{Retract}$. Now, since $(\mathsf{Insert}, \mathsf{Retract})$ is a repair of $(\mathcal{D}, \mathcal{IC})$, we have that $\mathcal{D} \cup \mathsf{Insert} \setminus \mathsf{Retract} \models \mathcal{IC}$, thus M is a (two-valued) model of $\mathcal{IC}$. Moreover, $\mathsf{Insert} \cap \mathcal{D} = \emptyset$ and $\mathsf{Retract} \subseteq \mathcal{D}$, hence we have the following:

- $M^t \setminus \mathcal{D} = (\mathcal{D} \cup \mathsf{Insert} \setminus \mathsf{Retract}) \setminus \mathcal{D} = \mathsf{Insert}$,
- $\mathcal{D} \setminus M^t = \mathcal{D} \setminus (\mathcal{D} \cup \mathsf{Insert} \setminus \mathsf{Retract}) = \mathsf{Retract}$. □

The above formalization in terms of two-valued models has the drawback that a unified database $\mathcal{UDB}$ in need of a repair is inconsistent. In order to avoid reasoning on inconsistent theories, and since classical logic can infer everything from an inconsistent theory, we develop another formalization, based on a three-valued semantics. The benefit of this is that, as we show below, *any* database has models w.r.t. appropriate three-valued semantics, from which it is possible to pinpoint the inconsistent information, and thus it is also possible to extract repairs for $\mathcal{UDB}$.

The underlying 3-valued semantics considered here is induced by the algebraic structure $\mathcal{THREE}$, shown in the double-Hasse diagram of Figure 1. Intuitively, the elements t and f in $\mathcal{THREE}$ correspond to the usual classical elements `true` and `false`, while the third element, $\top$, represents inconsistent information (or belief).

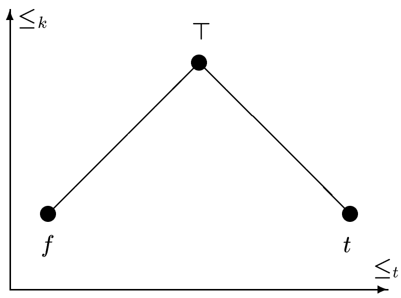

Fig. 1. The structure $\mathcal{THREE}$

Viewed horizontally, $\mathcal{THREE}$ is a complete lattice. We denote the meet, join, and the order reversing operation on the corresponding order relation (i.e., $\leq_t$) by $\wedge$, $\vee$, and $\neg$ (respectively). Viewed vertically, $\mathcal{THREE}$ is a semi-upper lattice. We denote by $\oplus$ the meet operation w.r.t. the corresponding order ($\leq_k$). We note that $\mathcal{THREE}$ is the algebraic structure that defines the semantics of several three-valued formalisms, such as LFI [13] and LP [36, 37].

The various semantic notions are defined on $\mathcal{THREE}$ as natural generalizations of similar classical ones: a *valuation* ν is a function that assigns a truth value in $\mathcal{THREE}$ to each atomic formula. Any valuation is extended to complex formulae in the obvious way. The set of the *designated* truth values in $\mathcal{THREE}$ (i.e., those elements in $\mathcal{THREE}$ that represent true assertions) consists of t and $\top$. A valuation ν *satisfies* a formula ψ iff $\nu(\psi)$ is designated. A valuation that assigns a designated value to every formula in a theory $\mathcal{T}$ is a (three-valued) *model* of $\mathcal{T}$.

Next we characterize the repairs of a database $\mathcal{DB}$ by its three-valued models:

Proposition 3. *Let* $(\mathcal{D}, \mathcal{IC})$ *be a database and let* M *be a two-valued model of* $\mathcal{IC}$*. Consider the three-valued valuation* N*, defined for every atom* p *by* $N(p) = \mathcal{H}^{\mathcal{D}}(p) \oplus M(p)$*, and let* $\mathsf{Insert} = N^{\top} \setminus \mathcal{D}$*,* $\mathsf{Retract} = N^{\top} \cap \mathcal{D}$*. Then* N *is a three-valued model of* $\mathcal{D} \cup \mathcal{IC}$*, and* $(\mathsf{Insert}, \mathsf{Retract})$ *is a repair of* $(\mathcal{D}, \mathcal{IC})$*.*

Proof: For the first claim, note that for three-valued valuations ν and μ, if for every atom p, $\nu(p) \geq_k \mu(p)$, then for every formula ψ, $\nu(\psi) \geq_k \mu(\psi)$ (the proof is by an easy induction on the structure of ψ). We denote this fact by $\nu \geq_k \mu$. Note also, that if $\nu \geq_k \mu$ and μ is a model of some theory $\mathcal{T}$, then ν is also a model of $\mathcal{T}$. Now, since by the definition of N, $N \geq_k \mathcal{H}^{\mathcal{D}}$, and since $\mathcal{H}^{\mathcal{D}}$ is a model of $\mathcal{D}$, N is a model $\mathcal{D}$. Similarly, $N \geq_k M$, and M is a model of $\mathcal{IC}$, thus N is also a model of $\mathcal{IC}$.

For the second part one has to show that the three conditions of Definition 4 are satisfied. Indeed, the first two conditions obviously hold. For the last condition, note that $\mathcal{D} \cup \mathsf{Insert} \setminus \mathsf{Retract} = \mathcal{D} \cup (N^{\top} \setminus \mathcal{D}) \setminus (N^{\top} \cap \mathcal{D}) = \mathcal{D} \cup (M^t \setminus \mathcal{D}) \setminus (M^f \cap \mathcal{D}) = \mathcal{D} \cup (M^t \setminus \mathcal{D}) \setminus (\mathcal{D} \setminus M^t) = M^t$. It follows that M is the minimal Herbrand model of $\mathcal{D} \cup \mathsf{Insert} \setminus \mathsf{Retract}$ and it is also a model of $\mathcal{IC}$, therefore $\mathcal{D} \cup \mathsf{Insert} \setminus \mathsf{Retract} \models \mathcal{IC}$. □

Again, it is possible to show that the converse is also true:

Proposition 4. *Let* $(\mathsf{Insert}, \mathsf{Retract})$ *be a repair of a database* $(\mathcal{D}, \mathcal{IC})$*. Then there is a three-valued model* N *of* $\mathcal{D} \cup \mathcal{IC}$*, such that* $\mathsf{Insert} = N^{\top} \setminus \mathcal{D}$ *and* $\mathsf{Retract} = N^{\top} \cap \mathcal{D}$*.*

Outline of proof: Consider a valuation N, defined as follows:

$$N(p) = \begin{cases} \top & \text{if } p \in \mathsf{Insert} \cup \mathsf{Retract}, \\ t & \text{if } p \notin \mathsf{Insert} \cup \mathsf{Retract} \text{ but } p \in \mathcal{D}, \\ f & \text{otherwise.} \end{cases}$$

Clearly, N is a (three-valued) model of $\mathcal{D}$ and $\mathcal{IC}$, and $N^{\top} \setminus \mathcal{D} = (\mathsf{Insert} \cup \mathsf{Retract}) \setminus \mathcal{D} = \mathsf{Insert}$, $N^{\top} \cap \mathcal{D} = (\mathsf{Insert} \cup \mathsf{Retract}) \cap \mathcal{D} = \mathsf{Retract}$. □

The last two propositions characterize the repairs of $\mathcal{UDB}$ in terms of pairs that are associated with three-valued models of $\mathcal{D} \cup \mathcal{IC}$. We shall denote the elements of these pairs as follows:

Definition 11. Let N be a three-valued model and let $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ be a knowledge-base. Denote: $\mathsf{Insert}^N = N^{\top} \setminus \mathcal{D}$ and $\mathsf{Retract}^N = N^{\top} \cap \mathcal{D}$.

We conclude this model-theoretic analysis by characterizing the set of the $\leq$-preferred repairs, where $\leq$ is one of the preference criteria, considered in Definition 6 (i.e., set inclusion or differences in cardinality).

Definition 12. Given a knowledge-base $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$, denote:
$\mathcal{M}^{\mathcal{DB}} = \{N \mid N \geq_k \mathcal{H}^{\mathcal{D}} \oplus M,\ M \text{ is a classical model of } \mathcal{IC}\}$. [12]

Example 3. In what follows we shall write $M = \{p_i : x_i\}$ for $M(p_i) = x_i$ ($x_i \in \{t, f, \top\}$, $i = 1, \ldots, n$). Let $\mathcal{DB} = (\{p, r\},\ \{p \to q\})$. We have that $\mathcal{H}^{\mathcal{D}} = \{p{:}t,\ q{:}f,\ r{:}t\}$, and so $\mathcal{M}^{\mathcal{DB}} = \{N \mid N(p) \geq_k t,\ N(q) = \top,\ N(r) \geq_k t\} \cup \{N \mid N(p) = \top, N(q) \geq_k f,\ N(r) \geq_k t\}$.

Definition 13. Let $\mathcal{S}$ be a set of three-valued valuations, and $N_1, N_2 \in \mathcal{S}$.

- N_1 is $\leq_i$-*more consistent* than N_2, if $N_1^{\top} \subset N_2^{\top}$.
- N_1 is $\leq_c$-*more consistent* than N_2, if $\#(N_1^{\top}) < \#(N_2^{\top})$. [13]
- $N \in \mathcal{S}$ is $\leq_i$-*maximally consistent* in $\mathcal{S}$ (respectively, N is $\leq_c$-*maximally consistent* in $\mathcal{S}$), if there is no $N' \in \mathcal{S}$ that is $\leq_i$-more consistent than N (respectively, no $N' \in \mathcal{S}$ is $\leq_c$-more consistent than N).

Proposition 5. *If N is a $\leq_i$-maximally consistent element in $\mathcal{M}^{\mathcal{DB}}$, then* ($\mathsf{Insert}^N$, $\mathsf{Retract}^N$) *is a $\leq_i$-preferred repair of $\mathcal{DB}$.*

Proposition 6. *Suppose that* (Insert, $\mathsf{Retract}$) *is a $\leq_i$-preferred repair of $\mathcal{DB}$. Then there is a $\leq_i$-maximally consistent element N in $\mathcal{M}^{\mathcal{DB}}$ s.t.* $\mathsf{Insert} = \mathsf{Insert}^N$ *and* $\mathsf{Retract} = \mathsf{Retract}^N$.

Note 1. Propositions 5 and 6 hold also when $\leq_i$ is replaced by $\leq_c$.

[12] Note that N is a *three-valued* valuation and M is a *two-valued* model of $\mathcal{IC}$.
[13] Recall that $\#(S)$ denotes the size of S.

Example 4. Consider again Example 2. We have that:

$$\mathcal{UDB} = (\mathcal{D}, \mathcal{IC}) = (\,\{p(a),\ p(b),\ q(a),\ q(c)\},\ \{\forall X\,(p(X)\rightarrow q(X))\}\,).$$

Thus $\mathcal{H}^{\mathcal{D}} = \{p(a):t,\ p(b):t,\ p(c):f,\ q(a):t,\ q(b):f,\ q(c):t\}$, and the classical models of $\mathcal{IC}$ are those in which either $p(y)$ is false or $q(y)$ is true for every $y \in \{a,b,c\}$. Now, since in $\mathcal{H}^{\mathcal{D}}$ neither $p(b)$ is false nor $q(b)$ is true, it follows that *every* element in $\mathcal{M}^{\mathcal{UDB}}$ must assign $\top$ either to $p(b)$ or to $q(b)$. Hence, the $\leq_i$-maximally consistent elements in $\mathcal{M}^{\mathcal{UDB}}$ (which in this case are also the $\leq_c$-maximally consistent elements in $\mathcal{M}^{\mathcal{UDB}}$) are the following:

$$M_1 = \{\,p(a){:}t,\ p(b){:}\top,\ p(c){:}f,\ q(a){:}t,\ \ q(b){:}f,\ q(c){:}t\,\}$$

$$M_2 = \{\,p(a){:}t,\ p(b){:}t,\ p(c){:}f,\ q(a){:}t,\ q(b){:}\top,\ q(c){:}t\,\}$$

By Propositions 5 and 6, then, the $\leq_i$-preferred repairs of $\mathcal{UDB}$ (which are also its $\leq_c$-preferred repairs) are $(\mathsf{Insert}^{M_1},\ \mathsf{Retract}^{M_1}) = (\emptyset,\ \{p(b)\})$ and $(\mathsf{Insert}^{M_2},\ \mathsf{Retract}^{M_2}) = (\{q(b)\},\ \emptyset)$ (cf. Example 2).

Similarly, the $\leq_i$-maximally consistent (and the $\leq_c$-maximally consistent) elements in $\mathcal{M}^{\mathcal{DB}}$, where $\mathcal{DB}$ is the database of Example 3, are $N_1 = \{\,p{:}t,\ q{:}\top,\ r{:}t\,\}$ and $N_2 = \{\,p{:}\top, q{:}f,\ r{:}t\,\}$. It follows that the preferred repairs in this case are $(\{q\},\ \emptyset)$ and $(\emptyset,\ \{p\})$.

4 Database repair – An abductive approach

In [7] we have presented an abductive approach to the problem of combining inconsistent databases. In this section we give an outline of this method. For more detailed description the reader is referred to [7]; the application itself is available at `http://www.cs.kuleuven.ac.be/~dtai/kt`.

A high level description of the integration problem under consideration is given in ID-logic [15], which is a framework for declarative knowledge representation that extends classical logic with inductive definitions. This logic incorporates two types of knowledge: definitional and assertional. Assertional knowledge is a set of first-order statements, representing a general truth about the domain of discourse. Definitional knowledge is a set of rules of the form $p \leftarrow \mathcal{B}$, in which the head p is a predicate and the body $\mathcal{B}$ is a first order formula. A predicate that appears in a head of a rule is called *defined*; a predicate that does not occur in any head is called *open*, or *abducible*.

A *theory* $\mathcal{T}$ in ID-logic is therefore a pair (*Def*, *Fol*), where *Def* (the definitional knowledge) is a set of rules as described above, and *Fol* (the assertional knowledge) is a set of first order statements. The meaning of $\mathcal{T}$ is defined by the *extended well-founded semantics* [35] as follows: let M

be an arbitrary two-valued interpretation for the open predicates in *Def*. Once M is determined, *Def* becomes a standard logic program, with a unique well-founded model [42]. This model is then a model of the whole theory $\mathcal{T}$ if it is also a model of *Fol*.

ID-logic is a generalization of the notion of abductive logic programs (ALP) [18]. For instance, the open predicates of a theory in ID-logic correspond to the abducibles in an abductive logic program. Consequently, solutions of abductive logic programs that are computed by an abductive solver are also models of the corresponding ID-logic theory. Here we use such a solver, called the $\mathcal{A}$-system [7, 27] for computing solutions. The main idea of this solver is to reduce a high level specification into a lower level constraint store, which is managed by a constraint solver. The solver combines the refutation procedures SLDNFA [17] and ACLP [29], and uses an improved control strategy. In our case, solutions are repairs of a database, and in order to compute *preferred* solutions (i.e., preferred repairs for the integrated database), the $\mathcal{A}$-system has been extended with a simple branch and bound component, called *optimizer* (see [7]). This is actually a "filter" on the solutions space that speeds-up execution and makes sure that only the desired solutions will be obtained.

The elements of the distributed databases are uniformly represented by the unary predicate `db`, and the elements of a repaired database are represented by the unary predicate `fact`. In order to compute these elements, two open predicates are used: `retract` and `insert`. These predicates represent, respectively, the facts that may be removed and those that may be introduced for restoring the consistency of the unified database. The rules for computing the elements of a repaired database are then defined as follows:

```
fact(X) :- db(X), not retract(X).
fact(X) :- insert(X).
```

In addition, the following integrity constraints are specified: [14]

- It is inconsistent to have a retracted element that does not belong to some database:
  `ic :- retract(X), not db(X).`
- It is inconsistent to have an inserted element that belongs to a database:
  `ic :- insert(X), db(X).`

To make sure that all the integrity constraints will hold w.r.t. the combined data, every occurrence of a database fact `R(x)` in some integrity constraint is replaced by `fact`$(R(x))$.

[14] In what follows we use the notation "`ic :- B`" to denote the denial "`false` $\leftarrow$ `B`".

Below is a code for implementing Example 1: [15]

```
defined(fact(_)). defined(db(_)). open(insert(_)). open(retract(_)).

fact(X) :- db(X), not(retract(X)).
fact(X) :- insert(X).
ic :- insert(X), db(X).
ic :- retract(X), not db(X).

db(teaches(1,1)).  db(teaches(2,2)).                    % D1
db(teaches(2,3)).                                       % D2
ic :- fact(teaches(X,Y)), fact(teaches(X,Z)), Y\=Z.     % IC
```

We have executed this code as well as other examples from the literature in our system. The soundness and completeness theorems given in the next section guarantee that the output in each case is indeed the set of the most preferred solutions of the corresponding problem.

5 Soundness and Completeness

In this section we relate the two approaches of the previous sections through soundness and completeness theorems. For that we first recall some related results from [7] (Propositions 7 – 10 below). In what follows we denote by $\mathcal{T}$ an abductive theory, constructed as described in Section 4 for defining a composition problem of n databases $\mathcal{DB}_1, \ldots, \mathcal{DB}_n$.

Proposition 7. *Every abductive solution that is obtained by the $\mathcal{A}$-system for $\mathcal{T}$ is a repair of $\mathcal{UDB}$.*

Proposition 8. *Suppose that the query '$\leftarrow$ `true`' has a finite SLDNFA-tree w.r.t. $\mathcal{T}$. Then every repair of $\mathcal{UDB}$ is obtained by running $\mathcal{T}$ in the $\mathcal{A}$-system .*

Proposition 9. *Every output that is obtained by running $\mathcal{T}$ in the $\mathcal{A}$-system together with an $\leq_i$-optimizer [respectively, together with a $\leq_c$-optimizer] is an $\leq_i$-preferred repair [respectively, a $\leq_c$-preferred repair] of $\mathcal{UDB}$.*

Proposition 10. *Suppose that the query '$\leftarrow$ `true`' has a finite SLDNFA-tree w.r.t. $\mathcal{T}$. Then every $\leq_i$-preferred repair [respectively, every $\leq_c$-preferred repair] of $\mathcal{UDB}$ is obtained by running $\mathcal{T}$ in the $\mathcal{A}$-system together with an $\leq_i$-optimizer [respectively, together with a $\leq_c$-optimizer].*

By the propositions above and those of Section 3, we have:

[15] The code for Example 2 is similar.

Corollary 1. *Suppose that the query '$\leftarrow$ `true`' has a finite SLDNFA refutation tree w.r.t. $\mathcal{T}$. Then:*

1. *for every output* $(\mathsf{Insert}, \mathsf{Retract})$ *of the $\mathcal{A}$-system for $\mathcal{T}$, there is a classical model M of $\mathcal{IC}$ s.t.* $\mathsf{Insert} = M^t \setminus \mathcal{D}$ *and* $\mathsf{Retract} = \mathcal{D} \setminus M^t$.
2. *for every two-valued model M of $\mathcal{IC}$ there is an output* $(\mathsf{Insert}, \mathsf{Retract})$ *of the $\mathcal{A}$-system for $\mathcal{T}$, s.t.* $\mathsf{Insert} = M^t \setminus \mathcal{D}$ *and* $\mathsf{Retract} = \mathcal{D} \setminus M^t$.

Corollary 2. *Under the same assumption as that of Corollary 1,*

1. *for every output* $(\mathsf{Insert}, \mathsf{Retract})$ *of the $\mathcal{A}$-system for $\mathcal{T}$ there is a 3-valued model N of $\mathcal{D} \cup \mathcal{IC}$, s.t.* $\mathsf{Insert}^N = \mathsf{Insert}$ *and* $\mathsf{Retract}^N = \mathsf{Retract}$.
2. *for every 3-valued model N of $\mathcal{D} \cup \mathcal{IC}$ there is an output* $(\mathsf{Insert}, \mathsf{Retract})$ *of the $\mathcal{A}$-system for $\mathcal{T}$, s.t.* $\mathsf{Insert} = \mathsf{Insert}^N$ *and* $\mathsf{Retract} = \mathsf{Retract}^N$.

Corollary 3. *In the notations of Corollary 1 and under its assumption,*

1. *for every output* $(\mathsf{Insert}, \mathsf{Retract})$ *that is obtained by running $\mathcal{T}$ as an input to the $\mathcal{A}$-system together with an $\leq_i$-optimizer [respectively, together with a $\leq_c$-optimizer], there is an $\leq_i$-maximally consistent element [respectively, a $\leq_c$-maximally consistent element] N in $\mathcal{M}^{\mathcal{UDB}}$ s.t.* $\mathsf{Insert}^N = \mathsf{Insert}$ *and* $\mathsf{Retract}^N = \mathsf{Retract}$.
2. *for every $\leq_i$-maximally consistent element [respectively, $\leq_c$-maximally consistent element] N in $\mathcal{M}^{\mathcal{UDB}}$ there is a solution* $(\mathsf{Insert}, \mathsf{Retract})$ *that is obtained by running $\mathcal{T}$ in the $\mathcal{A}$-system together with an $\leq_i$-optimizer [respectively, together with a $\leq_c$-optimizer] s.t.* $\mathsf{Insert} = \mathsf{Insert}^N$ *and* $\mathsf{Retract} = \mathsf{Retract}^N$.

6 Related works

Coherent integration and proper representation of amalgamated data is extensively studied in the literature (see, e.g., [8, 12, 22, 24, 25, 31–34, 38, 41]). Common approaches for dealing with this task are based on techniques of belief revision [31], methods of resolving contradictions by quantitative considerations (such as "majority vote" [32]) or qualitative ones (e.g., defining priorities on different sources of information or preferring certain data over another [4, 9]), and approaches that are based on rewriting rules for representing the information in a specific form [25]. As in our case, abduction is used for database updating in [28] and an extended form of abduction is used in [26, 39] to explain modifications in a theory.

The use of three-valued logics is also a well-known technique for maintaining incomplete or inconsistent information; such logics are often used for defining fixpoint semantics of incomplete logic programs [19, 42], and so in principle they can be applied on integrity constraints in an (extended) clause form [15]. Three-valued formalisms such as LFI [13] are also the basis of paraconsistent methods to construct database repairs [14] and are useful in general for pinpointing inconsistencies [37]. As noted above, this is also the role of the three-valued semantics in our case.

Other approaches are based on semantics with arbitrarily many truth values, which allow to decode within the language itself some "meta-information" such as confidence factors, amount of belief for or against a specific assertion, etc. These approaches combine corresponding formalisms of knowledge representation (such as annotated logic programs [40, 41] or bilattice-based logics [5, 21, 33]) together with non-classical refutation procedures [20, 30, 40] that allow to detect inconsistent parts of a database and maintain them.

A closely related topic is the problem of giving consistent query answers in inconsistent database [3, 10, 25]. The idea is to answer database queries in a consistent way *without* computing the repairs of the database.

There are some other applications for integrating possibly conflicting information and updating databases (e.g., LUPS [2], BReLS [31], RI [30], Subrahmanian's mediator of annotated databases [41], and the system of Franconi et al. [22]). In comparison with such systems, we note that the main advantages of the present application are its expressive power (to the best of our knowledge, our approach is more expressive than any other available application for coherent data integration), the fact that no syntactical embedding of first-order formulae into other languages nor any extensions of two-valued semantics are necessary (our approach is a pure generalization of classical refutation procedures), and the encapsulation of the way that the underlying data is kept coherent (no input from the reasoner nor any other external policy for making preferences among conflicting sources is compulsory in order to resolve contradictions).

7 Future work

We conclude by sketching some issues for future work. First, as we have already noted, two more phases, which have not been considered here, might be needed for a complete data integration: (a) translation of difference concepts to a unified ontology, and (b) resolving contradictions among different integrity constraints. Another issue for future work is to allow definitions of *concepts* (and not only integrity constraints) in the databases (see [15] for a sketch on how this may be done). This data may be further combined with (possibly inconsistent) temporal information, (partial) transactions, and (contradictory) update information. Finally, since different databases may have different information about the same predicate, it is reasonable to use some weakened version of the closed word assumption as part of the integration process (for instance, an assumption that something is false unless it is in the database, or some other database has some information about it). An alternative approach may be to replace the closed word assumption with partial valuations (in case that databases may contain negative facts and not only positive ones).

References

1. C.E.Alchouron, P.Gradenfors, D.Makinson. On the logic of theory change: Partial meet contraction and revision function. *J. Symbolic Logic* 50, pp.510–530, 1985.
2. J.J.Alferes, J.A.Leite, L.M.Pereira, P.Quaresma. Planning as abductive updating. *Proc. AISB'00*, pp.1–8, 2000.
3. M.Arenas, L.Bertossi, J.Chomicki. Consistent query answers in inconsistent databases. *Proc. PODS'99*, pp.68–79, 1999.
4. O.Arieli. Four-valued logics for reasoning with uncertainty in prioritized data. In: *Information, Uncertainty, Fusion*, B.Bouchon-Meunier, R.R.Yager, L.Zadeh, editors, pp.263–309, Kluwer, 1999.
5. O.Arieli, A.Avron. Reasoning with logical bilattices. *J. Logic, Language, and Information* 5(1), pp.25–63, 1996.
6. O.Arieli, A.Avron. A model theoretic approach to recover consistent data from inconsistent knowledge-bases. *J. Automated Reasoning* 22(3), pp.263–309, 1999.
7. O.Arieli, B.Van Nuffelen, M.Denecker, M.Bruynooghe. Coherent composition of distributed knowledge-bases through abduction. *Proc. LPAR'01*, LNCS 2250, Springer, pp.620–635, 2001.
8. C.Baral, S.Kraus, J Minker. Combining Multiple Knowledge Bases. *IEEE Trans. on Knowledge and Data Enginnering* 3(2), pp.208–220, 1991.
9. S.Benferhat, C.Cayrol, D.Dubois, J.Lang, H.Prade. Inconsistency management and prioritized syntax-based entailment. *Proc. IJCAI'93*, pp.640–645, 1993.
10. S.Benferhat, D.Dubois, H.Prade. How to infer from inconsistent beliefs without revising? *Proc. IJCAI'95*, pp.1449–1455, 1995.
11. L.Bertossi, C.Schwind. Analytic tableau and database repairs:Foundations. *Proc. FoIKS'02*, LNCS 2284, Springer, pp.32–48, 2002.
12. F.Bry. Query Answering in Information Systems with Integrity Constraints. *Proc. IICIS'97*, pp.113–130, 1997.
13. W.Carnielli, J.Marcos. Tableau systems for logics of formal inconsistency. *Proc. IC-AI'01*, Vol.II, CSREA Press, pp.848–852, 2001.
14. S.de Amo, W.Carnielli, J.Marcos. A logical framework for integrating inconsistent information in multiple databases. *Proc. FoIKS'02*, LNCS 2284, pp.67–84, 2002.
15. M.Denecker. Extending classical logic with inductive definitions. *Proc. CL'2000*, LNAI 1861, Springer, pp.703–717, 2000.
16. M.Denecker, D.De Schreye. SLDNFA an abductive procedure for normal abductive programs. *Proc. Int. Joint Conf. and Symp. on Logic Programming*, pp.686–700, MIT Press, 1992.

17. M.Denecker, D.De Schreye. SLDNFA an abductive procedure for abductive logic programs. *J. Logic Programming* 34(2), pp.111–167, 1998.
18. M. Denecker, A.C. Kakas. Abductive Logic Programming. *J. Logic Programming (Special Issue on Abduction)* 44 (1-3), 2000.
19. M.Fitting, Kripke-Kleene semantics for logic programs. *J. Logic Programming* 2, pp.295–312, 1985.
20. M.Fitting. Negation as refutation. *Proc. LICS'89*, IEEE Press, pp.63–70, 1989.
21. M.Fitting. Bilattices and the semantics of logic programming. *J. Logic Programming* 11(2), pp.91–116, 1991.
22. E.Franconi, A.L.Palma, N.Leone, S.Perri, F.Scarcello. Census data repair: A challenging application og disjunctive logic programming. *Proc. LPAR'01*, LNCS 2250, Springer, pp.561–578, 2001.
23. P.Gaerdenfors, H.Rott. Belief revision. In: D.M.Gabbay, J.Hogger, J.A.Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming* Vol.4, pp.35–132, Oxford University Press, 1995.
24. M.Gertz, U.W.Lipeck. An extensible framework for repairing constraint violations. *Proc. IICIS'97*, pp.89–111, 1997.
25. S.Greco, E.Zumpano. Querying inconsistent databases. *Proc. LPAR'2000*, LNAI 1955, pp.308–325, Springer, 2000.
26. K.Inoue, C.Sakama. Abductive framework for nonmonotonic theory change. *Proc. IJCAI'95*, pp.204-210, 1995.
27. T.Kakas, B.Van Nuffelen, M.Denecker. $\mathcal{A}$-System: Problem solving through abduction. *Proc. IJCAI'01*, 2001.
28. T.Kakas, P.Mancarella. Database updates through abduction. *Proc. VLDB'90*, pp.650–661, 1990.
29. T.Kakas, A.Michael, C.Mourlas. ACLP: Abductive constraint logic programming. *J. Logic Programming* 44(1–3), pp.129–177, 2000.
30. M.Kifer, E.L.Lozinskii. A logic for reasoning with inconsistency. *J. Automated Reasoning* 9(2), pp.179–215, 1992.
31. P.Liberatore, M.Schaerf. BReLS: a system for the integration of knowledge bases. *Proc. KR'2000*, pp.145–152, 2000.
32. J.Lin, A.O.Mendelzon. Merging databases under constraints. *J. Cooperative Information Systems* 7(1), 55–76, 1998.
33. B.Messing. Combining knowledge with many-valued logics. *J. Data and Knowledge Engineering* 23, pp.297–315, 1997.
34. A.Olivé. Integrity checking in deductive databases. *Proc. VLDB'91*, pp.513–523, 1991.
35. L.M. Pereira, J.N. Aparicio, J.J. Alferes. Hypothetical Reasoning with Well Founded Semantics. *Proc. 3rd Scandinavian Conf. on AI*, pp.289-300, 1991.
36. G.Priest. Reasoning about truth. *Artificial Intelligence* 39, pp.231–244, 1989.
37. G.Priest. Minimally Inconsistent LP. *Studia Logica* 50, pp.321–331, 1991.
38. P.Z.Revesz. On the semantics of theory change: Arbitration between old and new information. *Proc. PODS'93*, pp.71–82, 1993.
39. C.Sakama, K.Inoue. Updating extended logic programs through abduction. *Proc. LPNMR'99*, pp.147–161, 1999.
40. V.S.Subrahmanian. Mechanical proof procedures for many valued lattice-based logic programming. *J. Non-Classical Logic* 7, pp.7–41, 1990.
41. V.S.Subrahmanian. Amalgamating knowledge-bases. *ACM Trans. on Database Systems* 19(2), pp.291–331, 1994.
42. A.Van Gelder, K.A. Ross, J.S. Schlipf. The Well-Founded Semantics for General Logic Programs, *J. of the ACM*, 38(3), pp.620–650, 1991.