Computational Methods for Database Repair by Signed Formulae [†]

Ofer Arieli (oarieli@mta.ac.il)
Department of Computer Science, The Academic College of Tel-Aviv, 4 Antokolski street, Tel-Aviv 61161, Israel.

Marc Denecker, Bert Van Nuffelen and Maurice Bruynooghe
({marcd,bertv,maurice}@cs.kuleuven.ac.be)
Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Heverlee, Belgium.

Abstract. We introduce a *simple* and *practical* method for repairing inconsistent databases. Given a possibly inconsistent database, the idea is to properly *represent* the underlying problem, i.e., to describe the possible ways of restoring its consistency. We do so by what we call *signed formulae*, and show how the 'signed theory' that is obtained can be used by a variety of off-the-shelf computational models in order to *compute* the corresponding solutions, i.e., consistent repairs of the database.

1. Introduction

Reasoning with inconsistent databases has been extensively studied in the last few years, especially in the context of integration of (possibly contradicting) independent data sources. The ability to synthesize distributed data sources into a single coherent set of information is a major challenge in the construction of knowledge systems for data sharing, and in many cases this property enables inference of information that cannot be drawn otherwise. If, for instance, one source 'knows' that either a or b must hold (but it doesn't know which one is true), and another source 'knows' $\neg a$ (i.e., that a cannot be true), then a mediator system may learn a new fact, b, that is not 'known' to either sources. There is another scenario, however, in which one of the sources also 'knows' $\neg b$. In this case, not only that the mediator system cannot consistently conclude b, but moreover, in order to maintain consistency it cannot accept the collective information of the sources! In particular, the consistency of each data source is not a sufficient condition for the consistency of their collective information, which again implies that maintaining consistency is a fundamental ability of database merging

[†] This paper is a revised and extended version of [9].

systems.[1]

The management of inconsistency in database systems requires dealing with many aspects. At the representation level, for instance, systems that keep their data consistent (in contrast to systems that are paraconsistent, that is: preserve the inconsistency and yet draw consistent conclusions out of it) should be able to *express* how to keep the data coherent. This, of course, carries on to the *reasoning* level and to the *implementation* level, where algorithms for consistency restoration should be developed and supported by corresponding computational models.

In this paper we introduce a novel approach to database repair that touches upon all the aspects mentioned above: we consider a uniform representation of repairs of inconsistent relational databases, that is, a general description of how to restore the consistency of database instances that do not satisfy a given set of integrity constraints. In our approach, a given repair problem is defined by a theory that consists of what we call *signed formulae*. This is a very simple but nevertheless general way of representing the underlying problem, which can be used by a variety of off-the-shelf computational systems. We show that out of the signed theories, these systems efficiently solve the problem by computing database *repairs*, i.e., new consistent database instances that differ from the original database instance by a minimal set of changes (with respect to set inclusion or set cardinality). Here we apply two types of tools for repairing a database:

- We show that the problem of finding repairs with minimal cardinality for a given database can be converted to the problem of finding minimal Herbrand models for the corresponding 'signed theory'. Thus, once the process for consistency restoration of the database has been represented by a signed theory (using a polynomial transformation), tools for minimal model computations (such as the Sicstus Prolog constraint solver [23], the satisfiability solver zChaff [50], and the answer set programming solver DLV [31]) can be used to efficiently find the required repairs.

- For finding repairs that are minimal with respect to set inclusion, satisfiability solvers of appropriate quantified Boolean formulae (QBF) can be utilized. Again, we provide a polynomial-time transformation to (signed) QBF theories, and show how QBF solvers (e.g., those of [12, 22, 30, 32, 35, 41, 54]) can be used to restore the database consistency.

[1] See., e.g., [4, 10, 11, 17, 18, 25, 27, 37, 36, 45] for more details on reasoning with inconsistent databases and further references to related works.

The rest of the paper is organized as follows: In Section 2 we discuss various *representation* issues that are related to database repair. We formally define the underlying problem in the context of propositional logic (Section 2.1), show how to represent it by signed formulae (Section 2.2), and then consider an extended framework based on first-order logic (Section 2.3). Section 3 is related to the corresponding *computational* and *reasoning* aspects. We show how constraint solvers for logic programs (Section 3.1) and quantified Boolean formulae solvers (Section 3.2) can be utilized for computing database repairs, based on the signed theories. At the end of this section we also give some relevant complexity results (Section 3.3). Section 4 is related to *implementation* issues. Some experimental results of several benchmarks are given and the suitability of the underlying computational models to the database repair problem is analyzed in light of the results. In Section 5 we link our approach to some related areas, such as belief revision and data merging, showing that some basic postulates of these areas are satisfied in our case as well. Finally, in Section 6 we conclude with some further remarks and observations.

2. Database repair and its representation

2.1. Preliminaries

In this section we set-up the framework and define the database repair problem with respect to this framework. To simplify the readings we start with the propositional case, leaving the first-order case to Section 2.3. This two-phase approach may also be justified by the fact that the main contribution of this paper can be expressed already at the propositional level.

Let $\mathcal{L}$ be a propositional language with $\mathcal{P}$ its underlying set of atomic propositions. A (propositional) *database instance* $\mathcal{D}$ is a finite subset of $\mathcal{P}$. The semantics of a database instance is given by the conjunction of the atoms in $\mathcal{D}$, augmented with the *Closed World Assumption* [53] ($\mathsf{CWA}(\mathcal{D})$), stating that each atom in $\mathcal{P}$ that does not appear in $\mathcal{D}$ is false. We shall denote the (unique) model of $\mathcal{D}$ and $\mathsf{CWA}(\mathcal{D})$ by $\mathcal{H}^{\mathcal{D}}$. Now, a formula ψ *follows from* $\mathcal{D}$ (or *is satisfied in* $\mathcal{D}$; notation: $\mathcal{D} \models \psi$) if $\mathcal{H}^{\mathcal{D}}$ satisfies ψ. Otherwise we say that ψ is *violated in* $\mathcal{D}$.

DEFINITION 2.1. 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 and consistent set of formulae in $\mathcal{L}$. A database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ is *consistent* if every formula in $\mathcal{IC}$ follows from $\mathcal{D}$ (notation: $\mathcal{D} \models \mathcal{IC}$), that is, there is no integrity constraint that is violated in $\mathcal{D}$.

Given an inconsistent database, our goal is to restore its consistency, i.e., to 'repair' the database:

DEFINITION 2.2. An *update* of a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ is a pair $(\mathsf{Insert}, \mathsf{Retract})$, where $\mathsf{Insert}, \mathsf{Retract} \subseteq \mathcal{P}$ are sets of atoms such that $\mathsf{Insert} \cap \mathcal{D} = \emptyset$ and $\mathsf{Retract} \subseteq \mathcal{D}$.[2] A *repair* of a database $\mathcal{DB}$ is an update $(\mathsf{Insert}, \mathsf{Retract})$ of $\mathcal{DB}$, for which $((\mathcal{D} \cup \mathsf{Insert}) \setminus \mathsf{Retract},\ \mathcal{IC})$ is a consistent database.

DEFINITION 2.3. The database $((\mathcal{D} \cup \mathsf{Insert}) \setminus \mathsf{Retract},\ \mathcal{IC})$ is called *the updated database* of $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ with update $(\mathsf{Insert}, \mathsf{Retract})$.

Intuitively, a database is updated by inserting the elements of Insert and removing the elements of $\mathsf{Retract}$. An update is a repair when its updated database is consistent. Note that if $\mathcal{DB}$ is consistent, then $(\emptyset, \emptyset)$ is a repair of $\mathcal{DB}$.

Definition 2.2 can easily be generalized by allowing repairs only to insert atoms belonging to some set $\mathcal{E}^{\mathsf{I}}$, and similarly to delete only atoms of a set $\mathcal{E}^{\mathsf{R}}$. Thus, for instance, it would be possible to forbid deletions by letting $\mathcal{E}^{\mathsf{R}} = \emptyset$. In the sequel, however, we shall always assume that any element in $\mathcal{P}$ may be inserted or deleted. This assumption can easily be lifted (see also footnote 3 below).

EXAMPLE 2.4. Let $\mathcal{P} = \{p, q\}$ and $\mathcal{DB} = (\{p\}, \{p \rightarrow q\})$. Clearly, this database is not consistent. It has three repairs: $\mathcal{R}_1 = (\{\}, \{p\})$, $\mathcal{R}_2 = (\{q\}, \{\})$, and $\mathcal{R}_3 = (\{q\}, \{p\})$. These repairs correspond, respectively, to removing p from the database, inserting q to the database, and performing both actions simultaneously.

As the example above shows, there are usually many ways to repair a given database, some of them may not be very natural or sensible. It is common, therefore, to specify some preference criterion on the possible repairs, and to apply only those repairs that are *(most) preferred* with respect to the underlying criterion. The most common criteria for preferring a repair $(\mathsf{Insert}, \mathsf{Retract})$ over a repair $(\mathsf{Insert}', \mathsf{Retract}')$ are *set inclusion* [4, 5, 10, 11, 17, 18, 27, 37, 36], i.e.,

$$\begin{aligned}&(\mathsf{Insert}, \mathsf{Retract}) \leq_i (\mathsf{Insert}', \mathsf{Retract}')\\ &\text{if } \mathsf{Insert} \cup \mathsf{Retract} \subseteq \mathsf{Insert}' \cup \mathsf{Retract}',\end{aligned}$$

[2] Note that these conditions imply that Insert and $\mathsf{Retract}$ must be disjoint.

or *minimal cardinality* [10, 11, 25, 45], i.e.,

$$\begin{aligned}&(\mathsf{Insert}, \mathsf{Retract}) \leq_c (\mathsf{Insert}', \mathsf{Retract}')\\ &\text{if } |\mathsf{Insert}| + |\mathsf{Retract}| \leq |\mathsf{Insert}'| + |\mathsf{Retract}'|\\ &(\text{where } |S| \text{ denotes the cardinality of the set } S).\end{aligned}$$

Both criteria above reflect the intuitive feeling that a ‘natural’ way to repair an inconsistent database should require a minimal change, therefore the repaired database is kept ‘as close as possible’ to the original one. According to this view, for instance, each one of the repairs $\mathcal{R}_1$ and $\mathcal{R}_2$ in Example 2.4 is strictly better than $\mathcal{R}_3$. Note also that $(\emptyset, \emptyset)$ is the *only* $\leq_i$-preferred and $\leq_c$-preferred repair of consistent databases, as expected.

2.2. Representation of repairs by signed formulae

Let $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ be a fixed database that should be repaired. The goal of this section is to characterize the repair process of $\mathcal{DB}$ by a logical theory. A key observation in this respect is that a repair of $\mathcal{DB}$ boils down to ‘switching’ some atoms of $\mathcal{P}$ from false to true or from true to false. Therefore, to encode a repair, we introduce a switching atom s_p for every atom p in $\mathcal{P}$.[3] A switching atom s_p expresses whether the status of p switches in the repaired database with respect to the original database: s_p is true when p is involved in the repair, either by removing it or inserting it, and is false otherwise (that is, s_p holds iff $p \in \mathsf{Insert} \cup \mathsf{Retract}$). We denote by $\mathsf{switch}(\mathcal{P})$ the set of switching atoms corresponding to the elements of $\mathcal{P}$. I.e., $\mathsf{switch}(\mathcal{P}) = \{s_p \mid p \in \mathcal{P}\}$.

The truth of an atom $p \in \mathcal{P}$ in the repaired database can be easily expressed in terms of the switching atom s_p of p. We define the *signed literal* τ_p of p with respect to $\mathcal{D}$ as follows:

$$\tau_p = \begin{cases} \neg s_p & \text{if } p \in \mathcal{D}, \\ s_p & \text{otherwise.} \end{cases}$$

An atom p is true in the repaired database if and only if its signed literal τ_p is true.

Now, as the repaired database can be expressed in terms of the switching atoms, we can also formalize the consistency of the repaired

[3] In general, one can impose the requirement that inserted atoms belong to $\mathcal{E}^{\mathsf{I}}$ and deleted atoms belong to $\mathcal{E}^{\mathsf{R}}$, by introducing switching atoms only for the atoms in $(\mathcal{E}^{\mathsf{I}} \setminus \mathcal{D}) \cup (\mathcal{E}^{\mathsf{R}} \cap \mathcal{D})$. An atom of this set with a truth value true encodes either an insertion of an element in $\mathcal{E}^{\mathsf{I}} \setminus \mathcal{D}$ or a deletion of an element in $\mathcal{E}^{\mathsf{R}} \cap \mathcal{D}$.

database with respect to $\mathcal{IC}$ in terms of the switching atoms. This condition is expressed by the theory obtained from $\mathcal{IC}$ by simultaneously substituting signed literals τ_p for all atoms p occurring in $\mathcal{IC}$. Formally, for every formula ψ of $\mathcal{L}$, its *signed formula* with respect to $\mathcal{D}$ is defined as follows:

$$\overline{\psi} = \psi\,[\,\tau_{p_1}/p_1\,,\ \ldots\ ,\ \tau_{p_m}/p_m\,].$$

As we shall show below (Theorem 2.6), repairs of $\mathcal{DB}$ correspond to models of $\overline{\mathcal{IC}} = \{\overline{\psi} \mid \psi \in \mathcal{IC}\}$.

EXAMPLE 2.5. Consider again the database $\mathcal{DB} = (\{p\}, \{p \to q\})$ of Example 2.4. In this case $\tau_p = \neg s_p$ and $\tau_q = s_q$, hence the signed formula of $\psi = p \to q$ is $\overline{\psi} = \neg s_p \to s_q$, or, equivalently, $s_p \vee s_q$. Intuitively, this formula indicates that in order to restore the consistency of $\mathcal{DB}$, at least one of p or q should be 'switched', i.e., either p should be removed from the database or q should be inserted to it. Indeed, the three classical models of $\overline{\psi}$ are exactly the three valuations on $\{s_p, s_q\}$ that are associated with the three repairs of $\mathcal{DB}$ (see Example 2.4). As Theorem 2.6 below shows, this is not a coincidence.

Next we formulate the main correctness theorems of our approach. First we express the correspondences between updates and valuations of the switching atoms. Given an update $\mathcal{R} = (\mathsf{Insert}, \mathsf{Retract})$ of a database $\mathcal{DB}$, define a valuation $\nu^{\mathcal{R}}$ on $\mathsf{switch}(\mathcal{P})$ as follows:

$$\nu^{\mathcal{R}}(s_p) = t \text{ iff } p \in \mathsf{Insert} \cup \mathsf{Retract}.$$

$\nu^{\mathcal{R}}$ is called the valuation that is *associated with* $\mathcal{R}$. Conversely, a valuation ν of $\mathsf{switch}(\mathcal{P})$ *induces* a database update $\mathcal{R}^{\nu} = (\mathsf{Insert}, \mathsf{Retract})$, where $\mathsf{Insert} = \{p \notin \mathcal{D} \mid \nu(s_p) = t\}$ and $\mathsf{Retract} = \{p \in \mathcal{D} \mid \nu(s_p) = t\}$. Obviously, these mappings are the inverse of each other.

THEOREM 2.6. *For a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$, let $\overline{\mathcal{IC}} = \{\overline{\psi} \mid \psi \in \mathcal{IC}\}$.*

a) *if $\mathcal{R}$ is a repair of $\mathcal{DB}$ then $\nu^{\mathcal{R}}$ is a model of $\overline{\mathcal{IC}}$,*

b) *if ν is a model of $\overline{\mathcal{IC}}$ then $\mathcal{R}^{\nu}$ is a repair of $\mathcal{DB}$.*

Proof. For (a), suppose that $\mathcal{R}$ is a repair of $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$. Then, in particular, $\mathcal{D}^{\mathcal{R}} \models \mathcal{IC}$, where $\mathcal{D}^{\mathcal{R}} = (\mathcal{D} \cup \mathsf{Insert}) \setminus \mathsf{Retract}$. Let $\psi \in \mathcal{IC}$ and let $\mathcal{H}^{\mathcal{D}^{\mathcal{R}}}$ be the (unique) model of $\mathcal{D}^{\mathcal{R}}$ and $\mathsf{CWA}(\mathcal{D}^{\mathcal{R}})$. Then $\mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(\psi) = t$, and so it remains to show that $\nu^{\mathcal{R}}(\overline{\psi}) = \mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(\psi)$. The proof of this is by induction on the structure of ψ, and we show only the base step (the rest is trivial), i.e., for every atom $p \in \mathsf{Dom}$, $\nu^{\mathcal{R}}(\overline{p}) = \mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(p)$. Note that $\nu^{\mathcal{R}}(\overline{p}) = \nu^{\mathcal{R}}(\tau_p)$, hence:

if $p \in \mathcal{D} \setminus \mathsf{Retract}$, then $p \in \mathcal{D}^{\mathcal{R}}$, and so $\nu^{\mathcal{R}}(\overline{p}) = \nu^{\mathcal{R}}(\neg s_p) = \neg\nu^{\mathcal{R}}(s_p) = \neg f = t = \mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(p)$.

if $p \in \mathsf{Retract}$, then $p \in \mathcal{D} \setminus \mathcal{D}^{\mathcal{R}}$, thus $\nu^{\mathcal{R}}(\overline{p}) = \nu^{\mathcal{R}}(\neg s_p) = \neg\nu^{\mathcal{R}}(s_p) = \neg t = f = \mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(p)$.

if $p \in \mathsf{Insert}$, then $p \in \mathcal{D}^{\mathcal{R}} \setminus \mathcal{D}$, hence $\nu^{\mathcal{R}}(\overline{p}) = \nu^{\mathcal{R}}(s_p) = t = \mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(p)$.

if $p \notin \mathcal{D} \cup \mathsf{Insert}$, then $p \notin \mathcal{D}^{\mathcal{R}}$, and so $\nu^{\mathcal{R}}(\overline{p}) = \nu^{\mathcal{R}}(s_p) = f = \mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(p)$.

For part (b), suppose that ν is a model of $\overline{\mathcal{IC}}$. Let

$$\mathcal{R}^{\nu} = (\mathsf{Insert}, \mathsf{Retract}) = (\{p \notin \mathcal{D} \mid \nu(s_p) = t\}, \{p \in \mathcal{D} \mid \nu(s_p) = t\}).$$

We shall show that $\mathcal{R}^{\nu}$ is a repair of $\mathcal{DB}$. According to Definition 2.2, it is obviously an update of $\mathcal{DB}$. It remains to show that every $\psi \in \mathcal{IC}$ follows from $\mathcal{D}^{\mathcal{R}} = (\mathcal{D} \cup \mathsf{Insert}) \setminus \mathsf{Retract}$, i.e., that $\mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(\psi) = t$, where $\mathcal{H}^{\mathcal{D}^{\mathcal{R}}}$ is the model of $\mathcal{D}^{\mathcal{R}}$ and $\mathsf{CWA}(\mathcal{D}^{\mathcal{R}})$. Since ν is a model of $\overline{\mathcal{IC}}$, $\nu(\overline{\psi}) = t$, and so it remains to show that $\mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(\psi) = \nu(\overline{\psi})$. Again, the proof is by induction on the structure of ψ, and we show here only the base step, that is: for every atom $p \in \mathsf{Dom}$, $\mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(p) = \nu(\overline{p})$. Again, $\nu^{\mathcal{R}}(\overline{p}) = \nu^{\mathcal{R}}(\tau_p)$, hence

if $p \in \mathcal{D} \setminus \mathsf{Retract}$, then $p \in \mathcal{D}^{\mathcal{R}}$ and $\nu(s_p) = f$, thus $\mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(p) = t = \neg\nu(s_p) = \nu(\neg s_p) = \nu(\overline{p})$.

if $p \in \mathsf{Retract}$, then $p \in \mathcal{D} \setminus \mathcal{D}^{\mathcal{R}}$ and $\nu(s_p) = t$, hence $\mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(p) = f = \neg\nu(s_p) = \nu(\neg s_p) = \nu(\overline{p})$.

if $p \in \mathsf{Insert}$, then $p \in \mathcal{D}^{\mathcal{R}} \setminus \mathcal{D}$ and $\nu(s_p) = t$, therefore $\mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(p) = t = \nu(s_p) = \nu(\overline{p})$.

if $p \notin \mathcal{D} \cup \mathsf{Insert}$, then $p \notin \mathcal{D}^{\mathcal{R}}$ and $\nu(s_p) = f$, and so $\mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(p) = f = \nu(s_p) = \nu(\overline{p})$. □

The second part of the above theorem implies, in particular, that in order to compute repairs for a given database $\mathcal{DB}$, it is sufficient to find the models of the signed formulae that are induced by the integrity constraints of $\mathcal{DB}$; the pairs that are induced by these models are the repairs of $\mathcal{DB}$.

We have now established a correspondence between arbitrary repairs of a database and models of the signed theory $\overline{\mathcal{IC}}$. It remains to show how preferred repairs according to some preference relation correspond to a specific class of models of $\overline{\mathcal{IC}}$. We do this for the minimal cardinality preference relation $\leq_c$ and the set inclusion preference relation $\leq_i$.

For any two valuations ν_1, ν_2 of $\mathsf{switch}(\mathcal{P})$, denote $\nu_1 \preceq_c \nu_2$ if the number of switching atoms that are assigned the value true by ν_1 is less than those that are assigned true by ν_2. Similarly, denote $\nu_1 \preceq_i \nu_2$ if the set of the true switching atoms of ν_1 is a subset of the set of the true switching atoms of ν_2. Now, the following property is straightforward:

LEMMA 2.7. *Let $\mathcal{R}_1$, $\mathcal{R}_2$ be two updates of a database $(\mathcal{D}, \mathcal{IC})$ and let ν_1, ν_2 be two models of $\overline{\mathcal{IC}} = \{\overline{\psi} \mid \psi \in \mathcal{IC}\}$. Then:*

a) *if $\mathcal{R}_1 \leq_c \mathcal{R}_2$ then $\nu^{\mathcal{R}_1} \preceq_c \nu^{\mathcal{R}_2}$ and if $\mathcal{R}_1 \leq_i \mathcal{R}_2$ then $\nu^{\mathcal{R}_1} \preceq_i \nu^{\mathcal{R}_2}$.*

b) *if $\nu_1 \preceq_c \nu_2$ then $\mathcal{R}^{\nu_1} \leq_c \mathcal{R}^{\nu_2}$ and if $\nu_1 \preceq_i \nu_2$ then $\mathcal{R}^{\nu_1} \leq_i \mathcal{R}^{\nu_2}$.*

This lemma leads to the following simple characterizations of $\leq_c$-preferred and $\leq_i$-preferred models in terms of the models of $\overline{\mathcal{IC}}$.

THEOREM 2.8. *For a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ let $\overline{\mathcal{IC}} = \{\overline{\psi} \mid \psi \in \mathcal{IC}\}$. Then:*

a) *if $\mathcal{R}$ is a $\leq_c$-preferred repair of $\mathcal{DB}$, then $\nu^{\mathcal{R}}$ is a $\leq_c$-minimal model of $\overline{\mathcal{IC}}$.*

b) *if ν is a $\leq_c$-minimal model of $\overline{\mathcal{IC}}$, then $\mathcal{R}^{\nu}$ is a $\leq_c$-preferred repair of $\mathcal{DB}$.*

Proof. By Theorem 2.6, the repairs of a database correspond exactly to the models of the signed theory $\overline{\mathcal{IC}}$. By Lemma 2.7, $\leq_c$-preferred repairs of $\mathcal{DB}$ (i.e., those with minimal cardinality) correspond to $\leq_c$-minimal models of $\overline{\mathcal{IC}}$. □

It follows that $\leq_c$-preferred repairs of a database can be computed by searching for models of $\overline{\mathcal{IC}}$ with minimal cardinality (called $\leq_c$-minimal models). We shall use this fact in Section 3, where we consider computations of preferred repairs.

A similar theorem holds also for $\leq_i$-preferred repairs:

THEOREM 2.9. *For a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ let $\overline{\mathcal{IC}} = \{\overline{\psi} \mid \psi \in \mathcal{IC}\}$. Then:*

a) *if $\mathcal{R}$ is an $\leq_i$-preferred repair of $\mathcal{DB}$, then $\nu^{\mathcal{R}}$ is an $\leq_i$-minimal model of $\overline{\mathcal{IC}}$.*

b) *if ν is an $\leq_i$-minimal model of $\overline{\mathcal{IC}}$, then $\mathcal{R}^\nu$ is an $\leq_i$-preferred repair of $\mathcal{DB}$.*

Proof. Similar to that of Theorem 2.8, replacing $\leq_c$ by $\leq_i$. □

2.3. First-order databases

We now turn to the first-order case. As we show below, using the standard technique of grounding, our method of database repairs by signed formulae may be applied in this case as well.

Let $\mathcal{L}$ be a language of first-order formulas based on a vocabulary consisting of the predicate symbols in a fixed database schema S and a finite set Dom of constants representing the elements of some domain of discourse. In a similar way to that considered in Section 2.1, it is possible to define a database instance $\mathcal{D}$ as a finite set of ground atoms in $\mathcal{L}$. The meaning of $\mathcal{D}$ is given by the conjunction of the atoms in $\mathcal{D}$ augmented with following three assumptions:

- the *Domain Closure Assumption* ($\mathsf{DCA}(\mathsf{Dom})$) states that all elements of the domain of discourse are named by constants in Dom,
- the *Unique Name Assumption* ($\mathsf{UNA}(\mathsf{Dom})$) states that different constants represent different objects, and
- the *Closed World Assumption* ($\mathsf{CWA}(\mathcal{D})$) states that each atom which is not explicitly mentioned in $\mathcal{D}$ is false.

These three assumptions are hard-wired in the inference mechanisms of the database and therefore are not made explicit in the integrity constraints. The meaning of a database instance under these three assumptions is formalized in a model theoretical way by the *least Herbrand model* semantics. The unique model of a database instance $\mathcal{D}$ is the least Herbrand model $\mathcal{H}^{\mathcal{D}}$, i.e., an interpretation in which the domain is Dom, each constant symbol $c \in \mathsf{Dom}$ is interpreted by itself, each predicate symbol $p \in S$ is interpreted by the set $\{(x_1, \ldots, x_n) \mid p(x_1, \ldots, x_n) \in \mathcal{D}\}$, and the interpretation of the equality predicate is the identity relation on Dom. As Dom may change during the lifetime of the database, it is sometimes called the *active domain* of the database. Again, we say that a first-order sentence ψ follows from $\mathcal{D}$ if the least Herbrand model of $\mathcal{D}$ satisfies ψ.

Now, a (first-order) database is a pair $(\mathcal{D}, \mathcal{IC})$, where $\mathcal{D}$ is a database instance, and the set $\mathcal{IC}$ of integrity constraints is a finite, consistent set of first-order sentences in $\mathcal{L}$. Consistency of databases is defined just as before.

As in the propositional case, an inconsistent first-order database $(\mathcal{D}, \mathcal{IC})$ can be repaired by inserting or deleting atoms about elements of Dom. However, there may be also other ways of repairing a database that do not have an equivalent in the propositional case:

- a database may be updated by adding new elements to Dom and inserting facts about them, or deleting elements from Dom and removing from the database instance all atoms in which they occur;
- a database may also be updated by equalizing different elements from Dom.

The following example illustrates these methods:

EXAMPLE 2.10.

a) Let $\mathcal{DB} = (\{P(a)\}, \{\forall x(P(x) \to Q(x))\})$. Clearly, this database is not consistent. When $\mathsf{Dom} = \{a\}$ the actual meaning of this database is given by $(\{P(a)\}, \{P(a) \to Q(a)\})$ and it is equivalent to the database considered in Examples 2.4 and 2.5 above. As noted in those examples, the repairs in this case, $\mathcal{R}_1 = (\{\}, \{P(a)\})$, $\mathcal{R}_2 = (\{Q(a)\}, \{\})$, and $\mathcal{R}_3 = (\{Q(a)\}, \{P(a)\})$, correspond, respectively, to removing $P(a)$ from the database, inserting $Q(a)$ to the database, and performing both actions simultaneously.

Suppose now that the database instance is $\{P(a), Q(b)\}$ and the domain of discourse is $\mathsf{Dom} = \{a, b\}$. Then the update $(\{a = b\}, \{\})$ would restore consistency by equalizing a and b. Notice that this solution violates the implicit constraint $\mathsf{UNA}(\mathsf{Dom})$.

b) Let $\mathcal{DB} = (\ \{P(a)\}\ ,\ \{\forall x(P(x) \to \exists y(y \neq x \wedge Q(x,y)))\}\ )$, and $\mathsf{Dom} = \{a\}$. Again, this database is not consistent. One of the repairs of this database is $\mathcal{R} = (\{Q(a,b)\}, \{\})$. It adds an element b to the domain Dom and restores the consistency of the integrity constraint, but this repair violates the implicit constraint $\mathsf{DCA}(\mathsf{Dom})$.

In the context of *database updating*, we need the ability to change the database domain and to merge and equalize two different objects of the database. However, this paper is about repairing database inconsistencies. In this context, it is much less clear whether database repairs that revise the database domain (and hence violate $\mathsf{DCA}(\mathsf{Dom})$) or revise the identity of objects (and hence violate $\mathsf{UNA}(\mathsf{Dom})$) can be viewed as acceptable repairs. In what follows we shall not consider such repairs as legitimate ones. From now on, we assume that a repair does not contain equality atoms and consists only of atoms in $\mathcal{L}$, and hence, does not force a revision of Dom. This boils down to the fact that $\mathsf{DCA}(\mathsf{Dom})$ and $\mathsf{UNA}(\mathsf{Dom})$ are considered as axioms of $\mathcal{IC}$ which must be preserved in all repairs. Under this assumption, it turns out to be easy to apply the propositional methods described in Section 2 on first-order databases. To do this, we use the standard process of *grounding*. We denote by $\mathsf{ground}(\psi)$ the grounding of a sentence ψ with respect to a finite domain Dom. That is,

- $\mathsf{ground}(\psi) = \psi$ if ψ is a ground atom,
- $\mathsf{ground}(\neg\psi) = \neg\mathsf{ground}(\psi)$,
  $\mathsf{ground}(\psi_1 \wedge \psi_2) = \mathsf{ground}(\psi_1) \wedge \mathsf{ground}(\psi_2)$,
  $\mathsf{ground}(\psi_1 \vee \psi_2) = \mathsf{ground}(\psi_1) \vee \mathsf{ground}(\psi_2)$,
- $\mathsf{ground}(\exists x\, \psi(x)) = \vee_{a \in \mathsf{Dom}}\, \psi[a/x]$,
  $\mathsf{ground}(\forall x\, \psi(x)) = \wedge_{a \in \mathsf{Dom}}\, \psi[a/x]$.
  (where $\psi[a/x]$ denotes the substitution in ψ of x by a).

Since Dom is finite, $\mathsf{ground}(\psi)$ is also finite. The resulting formula is further simplified as follows:

- substitution of true for equality $s = s$ and substitution of false for equality $s = t$ where $s \not\equiv t$,[4]
- elimination of truth values by the following rewriting rules:

$$\begin{array}{llllll} \mathsf{false} \wedge \varphi & \longrightarrow \mathsf{false} & \mathsf{true} \vee \varphi & \longrightarrow \mathsf{true} & \neg\mathsf{true} & \longrightarrow \mathsf{false} \\ \mathsf{true} \wedge \varphi & \longrightarrow \varphi & \mathsf{false} \vee \varphi & \longrightarrow \varphi & \neg\mathsf{false} & \longrightarrow \mathsf{true} \end{array}$$

Clearly, a sentence ψ is satisfied in $\mathcal{D}$ if and only if $\mathsf{ground}(\psi)$ is satisfied in $\mathcal{D}$. Now, the *Herbrand expansion* of a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ is the pair $(\mathcal{D}, \mathsf{ground}(\mathcal{IC}))$, where $\mathsf{ground}(\mathcal{IC}) = \{\mathsf{ground}(\psi) \mid \psi \in \mathcal{IC}\}$. As a Herbrand expansion of a given (first-order) database $\mathcal{DB}$ can be considered as a propositional database, we can apply Definition 2.2 on it for defining repairs of $\mathcal{DB}$.

PROPOSITION 2.11. *The database* $(\mathcal{D}, \mathcal{IC} \cup \{\mathsf{DCA}(\mathsf{Dom}), \mathsf{UNA}(\mathsf{Dom})\})$ *and the propositional database* $(\mathcal{D}, \mathsf{ground}(\mathcal{IC}))$ *have the same repairs.*

[4] In general, when a set $\mathcal{E}^{\mathsf{I}}$ of insertable atoms and a set $\mathcal{E}^{\mathsf{R}}$ of retractable atoms are specified, we substitute false for every atom $A \in \mathcal{P} \setminus (\mathcal{D} \cup \mathcal{E}^{\mathsf{I}})$, and true for every atom $A \in \mathcal{D} \setminus \mathcal{E}^{\mathsf{R}}$.

3. Computing preferred database repairs

In this section we show that various constraint solvers for logic programs (Section 3.1) and quantified Boolean formulae (Section 3.2) can be utilized for computing database repairs based on the signed theories. The complexity of these computations is also considered (Section 3.3).

3.1. Computing preferred repairs by model generation

First we show how solvers for constraint logic programs (CLPs), answer-set programming (ASP), and SAT solvers, can be used for computing $\leq_c$-preferred repairs (Section 3.1.1) and $\leq_i$-preferred repairs (Section 3.1.2). The experimental results are presented in Section 4.

3.1.1. *Computing $\leq_c$-preferred repairs*

In what follows we discuss two techniques to compute $\leq_c$-minimal Herbrand models. The first approach is based on using finite domain CLP solvers. Encoding the computation of $\leq_c$-preferred repairs using a finite domain constraint solver is a straightforward process. The switching atoms s_p are encoded as finite domain variables with domain $\{0, 1\}$. A typical encoding specifies the relevant constraints (i.e., the encoding of $\overline{\mathcal{IC}}$), assigns a special variable, `Sum`, for summing-up the values of the finite domain variables associated with the switching atoms (the sum corresponds to the number of true switching atoms), and searches for a solution with a minimal value for `Sum`.

EXAMPLE 3.1. Below is a code for repairing the database of Example 2.5 with the Sicstus Prolog finite domain constraint solver CLP(FD) [23][5].

```
domain([Sp,Sq],0,1),                   %domain of the atoms
Sp #\/ Sq,                             %the signed theory
sum([Sp,Sq],#=,Sum),                   %Sum: num of true atoms
minimize(labeling([],[Sp,Sq]),Sum).    %resolve with min. sum
```

The solutions computed here are $[1, 0]$ and $[0, 1]$, and the value of `Sum` is 1. This means that the cardinality of the $\leq_c$-preferred repairs of $\mathcal{DB}$ should be 1, and that these repairs are induced by the valuations $\nu_1 = \{s_p : t, s_q : f\}$ and $\nu_2 = \{s_p : f, s_q : t\}$.[6] Thus, the two $\leq_c$-minimal

[5] A Boolean constraint solver would also be appropriate here. As the Sicstus Prolog Boolean constraint solver has no minimization capabilities, we prefer to use here the finite domain constraint solver.

[6] Here and in what follows we write $\nu = \{x_1 : a_1, \ldots, x_n : a_n\}$ to denote that $\nu(x_i) = a_i$ for $i = 1, \ldots, n$.

repairs here are $(\{\}, \{p\})$ and $(\{q\}, \{\})$, which indeed insert or retract exactly one atomic formula.

A second approach is based on using the disjunctive logic programming system DLV [31]. To compute $\leq_c$-minimal repairs using DLV, the signed theory $\overline{\mathcal{IC}}$ is transformed into a propositional clausal form. A clausal theory is a special case of a disjunctive logic program without negation in the body of the clauses. The stable models of a disjunctive logic program without negation as failure in the body of rules coincide exactly with the $\leq_i$-minimal models of such a program. Hence, by transforming the signed theory $\overline{\mathcal{IC}}$ to clausal form, DLV can be used to compute $\leq_i$-minimal Herbrand models. To eliminate models with non-minimal cardinality, *weak constraints* are used. A weak constraint is a formula for which a cost value is defined. With each model computed by DLV, a cost is defined as the sum of the cost values of all weak constraints satisfied in the model. The DLV system can be asked to generate models with minimal total cost. The set of weak constraints used to compute $\leq_c$-minimal repairs is exactly the set of all atoms s_p; each atom has cost 1. Clearly, $\leq_i$-minimal models of a theory with minimal total cost are exactly the models with least cardinality.

EXAMPLE 3.2. Below is a code for repairing the database of Example 2.5 with DLV.

```
Sp v Sq.                            %the clause
:~ Sp.                              %the weak constraints
:~ Sq.                              %(their cost is 1 by default)
```

Clearly, the solutions here are $\{s_p : t, s_q : f\}$ and $\{s_p : f, s_q : t\}$. These valuations induce the two $\leq_c$-minimal repairs of $\mathcal{DB}$, $\mathcal{R}_1 = (\{\}, \{p\})$ and $\mathcal{R}_2 = (\{q\}, \{\})$.

3.1.2. *Computing $\leq_i$-preferred repairs*

The $\leq_i$-preferred repairs of a database $(\mathcal{D}, \mathcal{IC})$ correspond to the $\leq_i$-minimal Herbrand models of the signed theory $\overline{\mathcal{IC}}$. Below we use this fact for introducing some simple techniques to compute an $\leq_i$-preferred repair by model generators; in Section 3.2 we consider another method that is based on reasoning with quantified Boolean formulae.

A. A naive algorithm

First, we consider a straightforward iterative algorithm for computing all the $\leq_i$-preferred repairs of the input database. The idea behind the following algorithm is to compute, at each iteration, one $\leq_i$-minimal model of the union of the signed theory $\overline{\mathcal{IC}}$ and the exclusion of all the repairs that have been constructed in previous iterations. By Theorem 2.9, then, this model induces an $\leq_i$-preferred repair of the input database. A pseudo-code of the algorithm is shown in Figure 1.

```
    input: a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$.
1.  $\mathcal{T} = \overline{\mathcal{IC}}$; Exclude-Previous-Repairs $= \emptyset$;
2.  do {
3.     $\mathcal{T} = \mathcal{T} \cup$ Exclude-Previous-Repairs;
4.     compute one $\leq_i$-minimal Herbrand model of $\mathcal{T}$,
       denote it by $M$;
5.     if $\{s_p \mid M(s_p) = t\} = \emptyset$ then
6.        return $(\emptyset, \emptyset)$ and exit;
          % this is the only preferred repair
7.     else {
8.        return the update that is associated with $M$;
9.        $\psi_M = \neg \bigwedge_{\{s_p \mid M(s_p)=t\}} s_p$;
10.       Exclude-Previous-Repairs =
              Exclude-Previous-Repairs $\cup$ $\{\psi_M\}$;
11.    }
12. } until there are no $\leq_i$-minimal models for $\mathcal{T}$;
       % no more repairs
```

Figure 1. $\leq_i$-preferred repairs computation by minimal models.

EXAMPLE 3.3. Consider the database of Examples 2.4 and 2.5. At the first iteration, one of the two $\leq_i$-minimal Herbrand models of $\mathcal{T} = \overline{\psi} = s_p \vee s_q$ is computed. Suppose, without a loss of generality, that it is $\{s_p : t, s_q : f\}$. The algorithm thus constructs the corresponding ($\leq_i$-preferred) repair, which is $(\{\}, \{p\})$. At the next iteration $\neg s_p$ is added to $\mathcal{T}$ and the only $\leq_i$-minimal Herbrand model of the extended theory is $\{s_p : f, s_q : t\}$. This model is associated with another $\leq_i$-preferred repair of the input database, which is $(\{q\}, \{\})$, and this is the output of the second iteration. At the third iteration $\neg s_q$ is added, and the resulting theory is not consistent anymore. Thus, this theory has no $\leq_i$-minimal models, and the algorithm terminates. In particular, therefore, the third repair of the database (which is *not* an $\leq_i$-preferred one) is not produced by the algorithm.

In the last example the algorithm produces exactly the set of the $\leq_i$-preferred repairs of the input database. It is not difficult to see that this is the case for *any* input database. First, by Theorem 2.6, every database update that is produced by the algorithm (in line 8) is a repair, since it is associated with a valuation (M) that is a model of $\overline{\mathcal{IC}}$ (as M is an $\leq_i$-minimal model of $\mathcal{T}$). Moreover, by the next proposition, the output of the algorithm is exactly the set of the $\leq_i$-preferred repairs of the input database.

PROPOSITION 3.4. *A database update is produced by the algorithm of Figure 1 for input $\mathcal{DB}$ iff it is an $\leq_i$-preferred repair of $\mathcal{DB}$.*

Proof. One direction of the proposition immediately follows from the definition of the algorithm (see lines 4 and 8 in Figure 1). The converse follows from Theorem 2.9 and the fact that `Exclude-Previous-Repairs` blocks the possibility that the same repair will be computed more than once. □

Observe that Proposition 3.4 also implies the termination of the algorithm of Figure 1.

B. Some more robust methods

The algorithm described above implements a direct and simple method of computing all the $\leq_i$-preferred repairs, but it assumes the existence of an (external) procedure that computes one $\leq_i$-minimal Herbrand model of the underlying theory. In what follows we describe three techniques of using ASP/CLP/SAT-solvers for efficiently computing the desired repairs, without relying on any external process.

I. One possible technique is based on SAT-solvers. These solvers, e.g. zChaff [50], do not directly compute minimal models, but can be easily extended to do so. The algorithm uses the SAT-solver to generate models of the theory $\mathcal{T}$, until it finds a minimal model. Minimality of a model M of $\mathcal{T}$ can be verified by checking the unsatisfiability of $\mathcal{T}$, augmented with the axioms $\bigvee_{p \in M} \neg p$ and $\bigwedge_{p \notin M} \neg p$. The model M is minimal exactly when these axioms are inconsistent with $\mathcal{T}$. A pseudo-code of an algorithm that implements this approach is shown below.

```
if $\mathcal{T}$ is not satisfiable then halt;
while sat($\mathcal{T}$) {          % as long as $\mathcal{T}$ is satisfiable
    $M$ := solve($\mathcal{T}$);              % find a model of $\mathcal{T}$
    $\mathcal{T}$ := $\mathcal{T} \cup \left\{ \bigvee_{p \in M} \neg p \,,\, \bigwedge_{p \notin M} \neg p \right\}$;
}
return $M$         % this is an $\leq_i$-minimal model of $\mathcal{T}$
```

We have tested this approach using the SAT solver zChaff [50]; the results are discussed in Section 4.

II. Another possibility is to adapt CLP-techniques to compute $\leq_i$-minimal models of Boolean constraints. The idea is simply to make sure that whenever a Boolean variable (or a finite domain variable with domain $\{0, 1\}$) is selected for being assigned a value, one first assigns the value 0 before trying to assign the value 1.

PROPOSITION 3.5. *If the above strategy for value selection is used, then the first computed model is an $\leq_i$-minimal model.*

Proof. Consider the search tree of the CLP-problem. Each path in this tree represents a value assignment to a subset of the constraint variables. Internal nodes, correspond to partial solutions, are labeled with the variable selected by the labeling function of the solver and have two children: the left child assigns value 0 to the selected variable and the right child assigns value 1. We say that node n_2 is on the right of a node n_1 in this tree if n_2 appears in the right subtree, and n_1 appears in the left subtree of the deepest common ancestor node of n_1 and n_2. It is then easy to see that in such a tree, each node n_2 to the right of a node n_1 assigns the value 1 to the variable selected in this ancestor node, whereas n_1 assigns value 0 to this variable. Consequently, the left-most node in the search tree which is a model of the Boolean constraints, is $\leq_i$-minimal. □

In CLP-systems such as Sicstus Prolog, one can control the order in which values are assigned to variables. We have implemented the above strategy and discuss the results in Section 4.

EXAMPLE 3.6. Below is a code for computing an $\leq_i$-preferred repair of the database of Example 2.5, using CLP(FD).

```
domain([Sp,Sq],0,1),                % domain of the atoms
Sp #\/ Sq,                          % the signed theory
labeling([up,leftmost],[Sp,Sq]).    % find min. solution
```

For computing *all* the $\leq_i$-minimal repairs, a call to a procedure, `compute_minimal([Sp,Sq])`, should replace the last line of the code above. This procedure is defined as follows:

```
compute_minimal(Vars):-    % find one minimal solution
    once(labeling([up,leftmost],Vars)),
    bb_put(min_repair,Vars).
compute_minimal(Vars):-    % find another solution
    bb_get(min_repair,Solution),
    exclude_repair(Solution,Vars),
    compute_minimal(Vars).

exclude_repair(Sol,Vars):- % exclude previous solutions
    exclude_repair(Sol,Vars,Constraint),
    call(#\ Constraint).

exclude_repair([],[],1).
exclude_repair([1|Ss],[V|Vs],V#=1 #/\ C):-
    exclude_repair(Ss,Vs,C).
exclude_repair([0|Ss],[V|Vs],C):-
    exclude_repair(Ss,Vs,C).
```

Note that the code above is the exact encoding for the Sicstus Prolog solver of the algorithm in Figure 1.

III. A third option, mentioned already in Section 3.1.1, is to transform $\overline{\mathcal{IC}}$ to clausal form and use the DLV system. In this case the weak constraints are not needed.

3.2. Computing $\leq_i$-preferred repairs by QBF solvers

Quantified Boolean formulae (QBFs) are propositional formulae extended with quantifiers $\forall, \exists$ over propositional variables. It has been shown that this language is useful for expressing a variety of computational paradigms, such as default reasoning [20], circumscribing inconsistent theories [21], paraconsistent preferential reasoning [6], and computations of belief revision operators (see [29], as well as Section 5 below). In this section we show how QBF solvers can be used for computing the $\leq_i$-preferred repairs of a given database. In this case it is necessary to add to the signed formulae of $\overline{\mathcal{IC}}$ an axiom (represented by a quantified Boolean formula) that expresses $\leq_i$-minimality, i.e., that an $\leq_i$-preferred repair is not included in any other database repair. Then, QBF solvers such as QUBOS [12], EVALUATE [22], QUIP [30], QSOLVE [32], QuBE [35], QKN [41], SEMPROP [43], and DECIDE [54], can be applied to the signed quantified Boolean theory that is obtained, in order to compute the $\leq_i$-preferred repairs of the database. Below we give a formal description of this process.

3.2.1. *Quantified Boolean formulae*
In what follows we shall denote propositional formulae by Greek lower-case letters (usually ψ, ϕ) and QBFs by Greek upper-case letters (e.g., Ψ, Φ). Intuitively, the meaning of a QBF of the form $\exists p\, \forall q\, \psi$ is that there exists a truth assignment of p such that ψ is true for every truth assignment of q. Next we formalize this intuition.

As usual, we say that an occurrence of an atomic formula p is *free* if it is not in the scope of a quantifier $\mathsf{Q}p$, for $\mathsf{Q} \in \{\forall, \exists\}$, and we denote by $\Psi[\phi_1/p_1, \ldots, \phi_m/p_m]$ the uniform substitution of each free occurrence of a variable p_i in Ψ by a formula ϕ_i, for $i = 1, \ldots, m$. The notion of a *valuation* is extended to QBFs as follows: Given a function $\nu_{\rm at} : \mathsf{Dom} \cup \{\mathsf{t}, \mathsf{f}\} \to \{t, f\}$ s.t. $\nu(\mathsf{t}) = t$ and $\nu(\mathsf{f}) = f$, a valuation ν on QBFs is recursively defined as follows:

$$\nu(p) = \nu_{\rm at}(p) \text{ for every atom } p \in \mathsf{Dom} \cup \{\mathsf{t}, \mathsf{f}\},$$

$$\nu(\neg\psi) = \neg\nu(\psi),$$

$$\nu(\psi \circ \phi) = \nu(\psi) \circ \nu(\phi), \text{ where } \circ \in \{\wedge, \vee, \to, \leftrightarrow\},$$

$$\nu(\forall p\, \psi) = \nu(\psi[\mathsf{t}/p]) \wedge \nu(\psi[\mathsf{f}/p]),$$

$$\nu(\exists p\, \psi) = \nu(\psi[\mathsf{t}/p]) \vee \nu(\psi[\mathsf{f}/p]).$$

A valuation ν *satisfies* a QBF Ψ if $\nu(\Psi) = t$; ν is a *model* of a set Γ of QBFs if it satisfies every element of Γ. A QBF Ψ is *entailed by* a set Γ of QBFs (notation: $\Gamma \models \Psi$) if every model of Γ is also a model of Ψ. In what follows we shall use the following notations: for two valuations ν_1 and ν_2 we denote by $\nu_1 \leq \nu_2$ that for every atomic formula p, $\nu_1(p) \to \nu_2(p)$ is true. We shall also write $\nu_1 < \nu_2$ to denote that $\nu_1 \leq \nu_2$ and $\nu_2 \not\leq \nu_1$.

3.2.2. *Representing $\leq_i$-preferred repairs by signed QBFs*
It is well-known that quantified Boolean formulae can be used for representing circumscription [49], thus they properly express logical minimization [20, 21]. In our case we use this property for expressing minimization of repairs w.r.t. set inclusion.

Given a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$, denote by $\overline{\mathcal{IC}}_\wedge$ the conjunction of all the elements in $\overline{\mathcal{IC}}$ (i.e., the conjunction of all the signed formulae that are obtained from the integrity constraints of $\mathcal{DB}$). Consider the following QBF, denoted by $\Psi_{\mathcal{DB}}$:

$$\forall s'_{p1}, \ldots, s'_{p_n} \Big(\overline{\mathcal{IC}}_\wedge \,[\, s'_{p_1}/s_{p_1}, \ldots, s'_{p_n}/s_{p_n} \,] \to \\ \big(\textstyle\bigwedge_{i=1}^{n} (s'_{p_i} \to s_{p_i}) \to \bigwedge_{i=1}^{n} (s_{p_i} \to s'_{p_i}) \big) \Big).$$

Consider a model ν of $\overline{\mathcal{IC}}_{\wedge}$, i.e., a valuation for $s_{p_1}, \ldots, s_{p_n}$ that makes $\overline{\mathcal{IC}}_{\wedge}$ true. The QBF $\Psi_{\mathcal{DB}}$ expresses that every interpretation μ (valuation for $s'_{p_1}, \ldots, s'_{p_n}$) that is a model of $\overline{\mathcal{IC}}_{\wedge}$, has the property that $\mu \leq \nu$ implies $\nu \leq \mu$, i.e., there is no model μ of $\overline{\mathcal{IC}}_{\wedge}$, s.t. the set $\{s_p \mid \nu(s_p) = t\}$ properly contains the set $\{s_p \mid \mu(s_p) = t\}$. In terms of database repairs, this means that if $\mathcal{R}^{\nu} = (\mathsf{Insert}, \mathsf{Retract})$ and $\mathcal{R}^{\mu} = (\mathsf{Insert}', \mathsf{Retract}')$ are the database repairs that are associated, respectively, with ν and μ, then $\mathsf{Insert}' \cup \mathsf{Retract}' \not\subset \mathsf{Insert} \cup \mathsf{Retract}$. It follows, therefore, that in this case $\mathcal{R}^{\nu}$ is an $\leq_i$-preferred repair of $\mathcal{DB}$, and in general $\Psi_{\mathcal{DB}}$ represents $\leq_i$-minimality.

EXAMPLE 3.7. For the database $\mathcal{DB}$ of Examples 2.4 and 2.5, $\overline{\mathcal{IC}} \cup \Psi_{\mathcal{DB}}$ is the following theory Γ:

$$\Big\{ s_p \vee s_q \ ,$$
$$\forall s'_p \forall s'_q \Big((s'_p \vee s'_q) \rightarrow ((s'_p \rightarrow s_p) \wedge (s'_q \rightarrow s_q) \ \rightarrow \ (s_p \rightarrow s'_p) \wedge (s_q \rightarrow s'_q)) \Big)$$
$$\Big\}.$$

The models of Γ are those that assign t either to s_p or to s_q, but not to both of them, i.e., $\nu_1 = (s_p : t, s_q : f)$ and $\nu_2 = (s_p : f, s_q : t)$. The database updates that are induced by these valuations are, respectively, $\mathcal{R}^{\nu_1} = (\{\}, \{p\})$ and $\mathcal{R}^{\nu_2} = (\{q\}, \{\})$. By Theorem 3.8 below, these are the only $\leq_i$-preferred repairs of $\mathcal{DB}$.

THEOREM 3.8. *Let $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ be a database and $\overline{\mathcal{IC}} = \{\overline{\psi} \mid \psi \in \mathcal{IC}\}$. Then:*

a) *if $\mathcal{R}$ is an $\leq_i$-preferred repair of $\mathcal{DB}$ then $\nu^{\mathcal{R}}$ is a model of $\overline{\mathcal{IC}} \cup \Psi_{\mathcal{DB}}$,*

b) *if ν is a model of $\overline{\mathcal{IC}} \cup \Psi_{\mathcal{DB}}$ then $\mathcal{R}^{\nu}$ is an $\leq_i$-preferred repair of $\mathcal{DB}$.*

Proof. Suppose that $\mathcal{R} = (\mathsf{Insert}, \mathsf{Retract})$ is an $\leq_i$-preferred repair of $\mathcal{DB}$. In particular, it is a repair of $\mathcal{DB}$ and so, by Theorem 2.6, $\nu^{\mathcal{R}}$ is a model of $\overline{\mathcal{IC}}$. Since Theorem 2.6 also assures that a database update that is induced by a model of $\overline{\mathcal{IC}}$ is a repair of $\mathcal{DB}$, in order to prove both parts of the theorem, it remains to show that the fact that $\nu^{\mathcal{R}}$ satisfies $\Psi_{\mathcal{DB}}$ is a necessary and sufficient condition for assuring that $\mathcal{R}$ is $\leq_i$-minimal among the repairs of $\mathcal{DB}$. Indeed, $\nu^{\mathcal{R}}$ satisfies $\Psi_{\mathcal{DB}}$ iff for every valuation μ that satisfies $\overline{\mathcal{IC}}_{\wedge}$ and for which $\mu \leq \nu^{\mathcal{R}}$, it is also true that $\nu^{\mathcal{R}} \leq \mu$. Thus, $\nu^{\mathcal{R}}$ satisfies $\Psi_{\mathcal{DB}}$ iff there is no model

μ of $\overline{\mathcal{IC}}$ s.t. $\mu < \nu^{\mathcal{R}}$, iff (by Theorem 2.6 again) there is no repair $\mathcal{R}'$ of $\mathcal{DB}$ s.t. $\nu^{\mathcal{R}'} < \nu^{\mathcal{R}}$, iff there is no repair $\mathcal{R}' = (\mathsf{Insert}', \mathsf{Retract}')$ s.t. $\mathsf{Insert}' \cup \mathsf{Retract}' \subset \mathsf{Insert} \cup \mathsf{Retract}$, iff $\mathcal{R}$ is an $\leq_i$-minimal repairs of $\mathcal{DB}$. □

DEFINITION 3.9. [4, 5] $\mathcal{Q}$ is a *consistent query answer* of a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ if it holds in (the databases that are obtained from) *all* the $\leq_i$-preferred repairs of $\mathcal{DB}$.

An immediate consequence of Theorem 3.8 is that consistent query answering [4, 5, 37] may be represented in our context in terms of a consequence relation as follows:

COROLLARY 3.10. *$\mathcal{Q}$ is a consistent query answer of a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ iff $\overline{\mathcal{IC}} \cup \Psi_{\mathcal{DB}} \models \mathcal{Q}$.*

The last corollary and Section 3.1.2 provide, therefore, some additional methods for consistent query answering, all of them are based on signed theories.

3.3. Complexity

We conclude this section by an analysis of the computational complexity of the underlying problem. As we show below, Theorem 3.8 allows us to draw upper complexity bounds for the following two main approaches to database integration.

a) A *skeptical (conservative) approach* to query answering (considered, e.g., in [4, 5, 37]), in which an answer to a query $\mathcal{Q}$ and a database $\mathcal{DB}$ is evaluated with respect to (the databases that are obtained from) *all* the $\leq_i$-preferred repairs of $\mathcal{DB}$ (i.e., computations of consistent query answers; see Definition 3.9 above).

a) A *credulous approach* to the same problem, according to which queries are evaluated with respect to *some* $\leq_i$-preferred repair of $\mathcal{DB}$.

COROLLARY 3.11. *Credulous query answering lies in Σ_2^P, and skeptical query answering is in Π_2^P.*

Proof. By Theorem 3.8, credulous query answering is equivalent to satisfiability checking for $\overline{\mathcal{IC}} \cup \Psi_{\mathcal{DB}}$, and skeptical query answering is equivalent to entailment checking for the same theory (see also Corollary 3.10 above). Thus, these decision problems can be encoded by QBFs in prenex normal form with exactly one quantifier alternation. The corollary is obtained, now, by the following well-known result:

PROPOSITION 3.12. [60] *Given a propositional formula ψ, whose atoms are partitioned into $i \geq 1$ sets $\{p^1_1, \ldots, p^1_{m_1}\}, \ldots, \{p^i_1, \ldots, p^i_{m_i}\}$, deciding whether*

$$\exists p^1_1, \ldots, \exists p^1_{m_1}, \forall p^2_1, \ldots, \forall p^2_{m_2}, \ldots, \mathsf{Q} p^i_1, \ldots, \mathsf{Q} p^i_{m_i} \psi$$

is true, is Σ^P_i-complete (where $\mathsf{Q} = \exists$ if i is odd and $\mathsf{Q} = \forall$ if i is even). Also, deciding if

$$\forall p^1_1, \ldots, \forall p^1_{m_1}, \exists p^2_1, \ldots, \exists p^2_{m_2}, \ldots, \mathsf{Q} p^i_1, \ldots, \mathsf{Q} p^i_{m_i} \psi$$

is true, is Π^P_i-complete (where $\mathsf{Q} = \forall$ if i is odd and $\mathsf{Q} = \exists$ if i is even). $\Box$

As shown, e.g., in [37], the complexity bounds specified in the last corollary are strict, i.e., these decision problems are hard for the respective complexity classes.

4. Experiments and comparative study

The idea of using formulae that introduce new ('signed') variables aimed at designating the truth assignments of other related variables is used, for different purposes, e.g. in [7, 8, 19, 20]. In the area of database integration, signed variables are used in [37], and have a similar intended meaning as in our case. In [37], however, only $\leq_i$-preferred repairs are considered, and a rewriting process for converting relational queries over a database with constraints to extended disjunctive queries (with two kinds of negations) over a database without constraints, must be employed. As a result, only solvers that are able to process disjunctive Datalog programs and compute their stable models (e.g., DLV), can be applied. In contrast, as we have already noted above, motivated by the need to find *practical* and *effective* methods for repairing inconsistent databases, signed formulae serve here as a representative platform that can be directly used by a variety of off-the-shelf applications for computing (either $\leq_i$-preferred or $\leq_c$-preferred) repairs. In what follows we examine some of these applications and compare their appropriateness to the kind of problems that we are dealing with.

We have randomly generated instances of a database, consisting of three relations: *teacher* of schema (`teacher_name`), *course* of schema (`course_name`), and *teaches* of schema (`teacher_name`, `course_name`). Also, the following two integrity constraints were specified:

ic1: A course is given by one teacher:

$$\forall X\, \forall Y\, \forall Z\, \Big((teacher(X) \wedge teacher(Y) \wedge course(Z) \wedge teaches(X,Z) \wedge teaches(Y,Z)) \;\rightarrow\; X = Y \Big)$$

ic2: Each teacher gives at least one course:

$$\forall X\, \Big(teacher(X) \rightarrow \exists Y (course(Y) \;\wedge\; teaches(X,Y)) \Big)$$

The next four test cases (identified by the enumeration below) were considered:

1. Small database instances with **ic1** as the only constraint.
2. Larger database instances with **ic1** as the only constraint.
3. Databases with $\mathcal{IC} = \{\mathbf{ic1}, \mathbf{ic2}\}$, where the number of courses is the same as the number of teachers.
4. Databases with $\mathcal{IC} = \{\mathbf{ic1}, \mathbf{ic2}\}$ and fewer courses than teachers.

Note that in the first two test cases, only retractions of database facts are needed in order to restore consistency, in the third test case both insertion and retractions may be needed, and the last test case is unsolvable, as the theory is not satisfiable.

For each benchmark we generated a sequence of instances with an increasing number of database facts, and tested them w.r.t. the following applications:

- **ASP/CLP-solvers:** DLV [31] (release 2003-05-16), CLP(FD) [23] (version 3.10.1).
- **QBF-solvers:** SEMPROP [43] (release 24.02.02), QuBE-BJ [35] (release number 1.3).
- **SAT-solvers:** A minimal-model generator based on zChaff [50].

The goal was to construct $\leq_i$-preferred repairs within a time limit of five minutes. The systems DLV and CLP(FD) were tested also for constructing $\leq_c$-preferred repairs. All the experiments were done on a Linux machine, 800MHz, with 512MB memory. Tables I–IV show the results for providing the first answer.[7]

[7] Times are given in seconds, empty cells mean that timeout is reached without an answer, vars is the number of variables, IC is the number of grounded integrity constraints, and size is the size of the repairs. We focus on the computation of one minimal model. The reason is simply that in most sizable applications, the computation of all minimal models is not feasible (there are too many of them).

Table I. Results for test case 1.

Test info.			$\leq_i$-repairs						$\leq_c$-repairs	
No.	vars	IC	size	DLV	CLP	zChaff	SEMPROP	QuBE	DLV	CLP
1	20	12	8	0.005	0.010	0.024	0.088	14.857	0.011	0.020
2	25	16	7	0.013	0.010	0.018	0.015		0.038	0.020
3	30	28	12	0.009	0.020	0.039	0.100		0.611	0.300
4	35	40	15	0.023	0.020	0.008	0.510		2.490	1.270
5	40	48	16	0.016	0.020	0.012	0.208		3.588	3.220
6	45	42	17	0.021	0.030	0.008	0.673		12.460	10.350
7	50	38	15	0.013	0.020	0.009	0.216		23.146	20.760
8	55	50	20	0.008	0.030	0.018	1.521		29.573	65.530
9	60	58	21	0.014	0.030	0.036	3.412		92.187	136.590
10	65	64	22	0.023	0.030	0.009	10.460		122.399	171.390
11	70	50	22	0.014	0.030	0.019	69.925			
12	75	76	27	0.021	0.030	0.010	75.671			
13	80	86	29	0.021	0.030	0.009	270.180			
14	85	76	30	0.022	0.030	0.010				
15	90	78	32	0.024	0.040	0.020				
16	95	98	35	0.027	0.040	0.047				
17	100	102	40	0.017	0.040	0.016				
18	105	102	37	0.018	0.040	0.033				
19	110	124	43	0.030	0.040	0.022				
20	115	116	44	0.027	0.040	0.041				

Table II. Results for test case 2.

Test info.			$\leq_i$-repairs			
No.	vars	IC	size	DLV	CLP	zChaff
1	480	470	171	0.232	0.330	0.155
2	580	544	214	0.366	0.440	0.051
3	690	750	265	0.422	0.610	0.062
4	810	796	300	0.639	0.860	0.079
5	940	946	349	0.815	1.190	0.094
6	1080	1108	410	1.107	1.560	0.123
7	1230	1112	428	1.334	2.220	0.107
8	1390	1362	509	1.742	2.580	0.135
9	1560	1562	575	2.254	3.400	0.194
10	1740	1782	675	2.901	4.140	0.182
11	1930	2042	719	3.592	5.260	0.253

Table III. Results for test case 3.

Test info.		$\leq_i$-repairs				$\leq_c$-repairs	
No.	vars	size	DLV	CLP	zChaff	DLV	CLP
1	25	4	0.008	0.030	0.066	0.010	0.05
2	36	9	0.008	0.030	0.087	0.070	0.42
3	49	15	0.027	0.250	0.050	0.347	9.48
4	64	23	0.019	0.770	0.013	2.942	58.09
5	81	30	0.012	4.660	0.102	26.884	
6	100	34	0.021		0.058	244.910	
7	121	38	0.626		1.561		
8	144	47	0.907		2.192		
9	169	51	0.161		0.349		
10	196	68	1.877		4.204		
11	225	70	8.496		16.941		

Table IV. Results for test case 4.

Test info.			$\leq_i$-repairs			$\leq_c$-repairs	
No.	teachers	courses	DLV	CLP	zChaff	DLV	CLP
1	5	4	0.001	0.01	0.001	0.001	0.001
2	7	5	0.005	0.13	0.010	0.005	0.120
3	9	6	0.040	1.41	0.020	0.042	1.400
4	11	7	0.396	17.18	0.120	3.785	17.170
5	13	8	3.789		1.050	44.605	
6	15	9	44.573		13.370		
7	17	10					

The results of the first benchmark (Table I) already indicate that DLV, CLP, and zChaff perform much better than the QBF-solvers. In fact, among the QBF-solvers that were tested, only SEMPROP could repair within the time limit most of the database instances of benchmark 1, and none of them could successfully repair (within the time restriction) the larger database instances, tested in benchmark 2.

Another observation from Tables I–IV is that DLV, CLP, and the zChaff-based system, perform very good for minimal inclusion greedy algorithms. However, when using DLV and CLP for cardinality minimization, their performance is much worse. This is due to an exhaustive search for a $\leq_c$-minimal solution.

While in benchmark 1 the time differences among DLV, CLP, and zChaff, for computing $\leq_i$-repairs are marginal, in the other benchmarks the differences become more evident. Thus, for instance, zChaff performs better than the other solvers w.r.t. bigger database instances with many simple constraints (see benchmark 2), while DLV performs better when the problem has bigger and more complicated sets of constraints (see benchmark 3). The SAT approach with zChaff was the fastest in detecting unsatisfiable situations (see benchmark 4). As shown in Table IV, detecting unsatisfiability requires a considerable amount of time, even for small instances.

Some of the conclusions from the experiments may be summarized as follows:

1. In principle, QBF-solvers, CLP-solvers, ASP-solvers, and SAT-solvers are all adequate tools for computing database repairs.

2. All the QBF-solvers, as well as DLV and zChaff, are ‘black-boxes’ that accept the problem specification in a certain format. In contrast, CLP(FD) provides a more ‘open’ environment, in which it is possible to incorporate problem-specific search algorithms, such as the greedy algorithm for finding $\leq_i$-minimal repairs (see Section 3.1.2).

3. Currently, the performance of the QBF-solvers is considerably below that of the other solvers. Moreover, most of the QBF-solvers require that the formulae are represented in prenex CNF, and specified in Dimacs or Rintanen format. These requirements are usually space-demanding. In our context, the fact that many QBF-solvers (e.g., SEMPROP and QuBE-BJ) return only yes/no answers (according to the satisfiability of the input theory), is another problem, since it is impossible to construct repairs only by these answers. One needs to be able to extract the assignments to the outmost existentially quantified variables (as done, e.g., by DECIDE [54]).

   Despite these drawbacks of QBF-solvers, reasoning with QBFs seems to be particularly suitable for our needs, since this framework provides a natural way to express minimization (in our case, representations of optimal repairs). It is most likely, therefore, that future versions of QBF-solvers will be the basis of powerful mechanisms for handling consistency in databases.

5. Relations to merging and revision operators

In this section we link our approach to two related areas, namely belief revision and data merging. The general purpose in both areas is to determine what kind of information is *rational* to support in the context of dynamically evolving systems. It is common to describe ideal properties of the merging and/or revision process by a list of general *postulates*, reflecting some desiderata of the underlying operator. The idea is to describe the change in the data at an *abstract level*, independent on how the information is represented or manipulated.[8] In what follows we show that some basic postulates of the merging/revision operators are satisfied in our case, which implies that our framework may also be useful for purposes other than consistency restoration.

5.1. Data merging

The goal of a merging system is to synthesize a coherent belief from distributed data sources. This goal has attracted many studies with different methods for this task. Each method may be associated with a merging operator that formally describes the integration process under consideration. Among the merging operators that have been proposed are that of Baral et al. [13, 14], which is based on computations of maximal consistent subsets of the global information, Lin and Mendelzon's theory of merging by majority operators [47] that resolve contradictions among different sources by 'majority votes', Liberatore and Schaerf's merging operator [44] that in case of conflicts selects the most 'plausible' source(s) and ignores the others, and the operators in [16] that merge prioritized data sources in the context of possibilistic logic. Recently, Delgrande and Schaub [28] introduced two other types of merging operators, one of which produces a belief-set retaining as much as possible of the contents of the distributed sources, and the other one produces a new (possibly empty) belief-set to which the original sources are 'projected'.[9] In [42], Konieczny and Pino-Pérez take a more abstract view and study postulates for so-called IC merging operators, i.e. operators that merge a multi-set of belief bases and a set of integrity constraints into a consistent belief base that satisfies the integrity constraints.

In this section, we illustrate how our database repair methods can be viewed as an application of an operator for merging different databases into a database which satisfies a given set of integrity constraints. This operator takes a set of database instances $\mathfrak{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$ and a set $\mathcal{IC}$ of integrity constraints, and returns a formula $\Delta_{\mathcal{IC}}(\mathfrak{D})$ that

[8] See the seminal paper of Alchourrón, Gärdenfors and Makinson (AGM) [1], as well as, e.g., [26, 34, 39, 40, 42, 44].

[9] Thus, for instance, the common information $(p \wedge q) \vee (\neg p \wedge \neg q)$ of the belief sets $(p \wedge q)$ and $(\neg p \wedge \neg q)$ must be included in the belief-set of the former merging operator, while this formula may not be part of the merged belief set of the latter one. See [28] for more information.

characterizes all the repairs of $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$, where $\mathcal{D} = \cup_{\mathcal{D}_i \in \mathfrak{D}} \mathcal{D}_i$. In our context, we assume that the elements in $\mathfrak{D}$ share the same domain Dom. Consequently, the domain closure axiom DCA(Dom) and the unique name axioms UNA(Dom) hold. In addition, we assume that although each database might have incomplete knowledge, together all databases in $\mathfrak{D}$ have complete knowledge, and so we can apply the closed world assumption CWA on $\mathcal{D} = \cup_{\mathcal{D}_i \in \mathfrak{D}} \mathcal{D}_i$. It may happen that this database instance does not satisfy the integrity constraints $\mathcal{IC}$. In that case, every repair of $\mathcal{D}$ is a possible solution for the integration problem. Rather than returning the set of all repaired databases, our merging operator returns a *formula* that characterizes all repaired databases at once. This formula is a QBF expressing that the repaired databases satisfy the integrity constraints and have minimal Hamming distance with respect to the united database instance $\mathcal{D}$. Now, according to Theorem 3.8, this formula is represented by $\overline{\mathcal{IC}}_\wedge \wedge \Psi_{\mathcal{DB}}$, where $\Psi_{\mathcal{DB}}$ is the formula

$$\forall s'_{p1}, \ldots, s'_{p_n} \Big(\overline{\mathcal{IC}}_\wedge \, [\, s'_{p_1}/s_{p_1}, \ldots, s'_{p_n}/s_{p_n} \,] \to \\ (\textstyle\bigwedge_{i=1}^{n} (s'_{p_i} \to s_{p_i}) \to \bigwedge_{i=1}^{n} (s_{p_i} \to s'_{p_i}))\Big).$$

Accordingly, we define

$$\Delta_{\mathcal{IC}}(\mathfrak{D}) \ := \ \overline{\mathcal{IC}}_\wedge \wedge \Psi_{\mathcal{DB}}.$$

By Theorem 3.8, the databases that are obtained from the models of this QBF are indeed the closest consistent databases to $(\cup_{\mathcal{D}_i \in \mathfrak{D}} \mathcal{D}_i, \mathcal{IC})$. As such, $\Delta_{\mathcal{IC}}(\mathfrak{D})$ characterizes a class of possible worlds that correspond to the set of the $\leq_i$-preferred repaired databases. Each one of these worlds satisfies DCA(Dom) and UNA(Dom) and consequently, DCA(Dom) and UNA(Dom) should be considered as implicit integrity constraints of $\mathcal{IC}$. Below we formalize these considerations.

DEFINITION 5.1. A *database merging context* is a pair $\mathcal{U} = (\mathfrak{D}, \mathcal{IC})$, where $\mathfrak{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$ is a set of database instances, all of them having the same domain of discourse Dom, and $\mathcal{IC}$ is a set of first-order formulae (the ‘global integrity constraints’).

Given a merging context $\mathcal{U}$, the *united database* of $\mathcal{U}$ is a database $\mathcal{DB}^{\mathcal{U}}$ whose database instance consists of the union of all the elements in $\mathfrak{D}$. The QBF $\Delta_{\mathcal{IC}}(\mathfrak{D})$ defined above is called the *merging theory* of $\mathcal{U}$ (or the *repair characterization* of $\mathcal{DB}^{\mathcal{U}}$).

The next proposition immediately follows from Theorem 3.8:

PROPOSITION 5.2. *Let $\mathcal{U}$ be a database merging context. A pair $\mathcal{R}$ = (Insert, Retract) is an $\leq_i$-preferred repair of the united database $\mathcal{DB}^{\mathcal{U}}$ iff $\nu^{\mathcal{R}}$ is a model of the merging theory $\Delta_{\mathcal{IC}}(\mathfrak{D})$.*

Next we evaluate our merging operator by checking to what extent it satisfies the postulates of [42] (cf. [42, Definition 3.1]). Since the formula $\Delta_{\mathcal{IC}}(\mathfrak{D})$ is defined in terms of $\mathsf{switch}(\mathcal{L})$ while in [42] the merged belief base is expressed in terms of the original language $\mathcal{L}$, we shall use signed versions of the postulates.

The first postulate makes sure that the merging result preserves all the integrity constraints.

M0 $\Delta_{\mathcal{IC}}(\mathfrak{D}) \models \psi$ for every signed integrity constraint $\psi \in \overline{\mathcal{IC}}$.

Another basic property of data merging is its consistency.

M1 $\Delta_{\mathcal{IC}}(\mathfrak{D})$ is consistent.

The next postulate refers to situations in which the distributed databases are consistent with the integrity constraints.

M2 If $\mathcal{DB}^{\mathcal{U}}$ is a consistent database, then $\Delta_{\mathcal{IC}}(\mathfrak{D})$ is logically equivalent to $\bigwedge_{s_p \in \mathsf{switch}(\mathcal{L})} \neg s_p$.

In other words, [**M2**] states that if $\mathcal{DB}^{\mathcal{U}}$ is a consistent database, then $\mathcal{DB}^{\mathcal{U}}$ itself should be the unique repaired database. It follows, then, that nothing should be modified in case that the union of the distributed data is consistent with respect to the set of integrity constraints.

M3 If $\mathcal{IC}_1$ is logically equivalent to $\mathcal{IC}_2$ and $\cup_{\mathcal{D}_i \in \mathfrak{D}_1} \mathcal{D}_i = \cup_{\mathcal{D}_i \in \mathfrak{D}_2} \mathcal{D}_i$, then $\Delta_{\mathcal{IC}_1}(\mathfrak{D}_1)$ is logically equivalent to $\Delta_{\mathcal{IC}_2}(\mathfrak{D}_2)$.

This postulate states the principle of *irrelevancy of syntax*, that is, if the unions of database instances in $\mathfrak{D}_1$ and in $\mathfrak{D}_2$ are identical, and if $\mathcal{IC}_1$ is logically equivalent to $\mathcal{IC}_2$, then the result of merging with respect to $\mathcal{IC}_1$ will be the same as the merging with respect to $\mathcal{IC}_2$.

M4 $\Delta_{\mathcal{IC}_1}(\mathfrak{D}) \wedge \Delta_{\mathcal{IC}_2}(\mathfrak{D}) \models \Delta_{\mathcal{IC}_1 \cup \mathcal{IC}_2}(\mathfrak{D})$.

The last postulate is a weaker version of a postulate adapted from [38] (see also [42]), which corresponds to the extended AGM postulate ($K\dot{+}7$) for revision, but with respect to integrity constraints.

The next proposition vindicates our claim that the repair methodology induces a merging operator.

PROPOSITION 5.3. *Let $\mathcal{U} = (\mathfrak{D}, \mathcal{IC})$ be a database merging context. Then postulates* [**M0**] — [**M4**] *are valid for* $\Delta_{\mathcal{IC}}(\mathfrak{D})$.

Proof. Indeed, **[M0]** follows from the definition of $\Delta_{\mathcal{IC}}(\mathfrak{D})$ which contains $\overline{\mathcal{IC}}_\wedge$; **[M1]** follows from the assumption that $\mathcal{IC}$ is consistent, which implies that $\overline{\mathcal{IC}}$ is consistent as well and has a minimal model (also, by part (a) of Theorem 3.8, $\Delta_{\mathcal{IC}}(\mathfrak{D})$ always has a classical model). For **[M2]**, we observed earlier that if $\mathcal{DB}^{\mathcal{U}}$ is consistent then the empty update $(\emptyset, \emptyset)$ is the only $\leq_i$-minimal repair and $\Delta_{\mathcal{IC}}(\mathfrak{D})$ has only one model, which assigns false to all switching atoms s_p. Thus, this QBF is equivalent to $\bigwedge_{i=1}^{n} \neg s_{p_i}$. **[M3]** is clearly satisfied since if $\cup_{\mathcal{D}_i \in \mathfrak{D}_1} \mathcal{D}_i = \cup_{\mathcal{D}_i \in \mathfrak{D}_2} \mathcal{D}_i$ and if $\mathcal{IC}_1 \leftrightarrow \mathcal{IC}_2$ then $\overline{\mathcal{IC}_1} \leftrightarrow \overline{\mathcal{IC}_2}$ as well. In this case it is evident that $\Delta_{\mathcal{IC}_1}(\mathfrak{D}_1)$ and $\Delta_{\mathcal{IC}_2}(\mathfrak{D}_2)$ are logically equivalent. Finally, **[M4]** immediately follows from the fact that a minimal model of $\overline{\mathcal{IC}}_1$ which is also a minimal model of $\overline{\mathcal{IC}}_2$, is a minimal model of $\overline{\mathcal{IC}_1 \wedge \mathcal{IC}_2}$.
□

Postulates **[M0]** — **[M4]** above correspond to postulates that are also satisfied by Konieczny and Pino-Pérez's IC merging operator [42] (denoted in [42] by **[IC0]**, **[IC1]**, **[IC2]**, **[IC3]** and **[IC7]**; see Theorem 3.7 in that paper).[10] There are, however, a number of important differences between the present operator and IC merging operators of [42]. Some differences are due to the fact that our operator merges *databases* under explicit constraints together with implicit constraints DCA(Dom), UNA(Dom) and CWA. These constraints are absent in the framework of [42], where the underlying operators merge belief bases rather than databases. Another difference is that the present operator $\Delta_{\mathcal{IC}}$, unlike IC merging operators and unlike other operators that merge by 'majority votes' [46, 47], is *majority independent,* i.e., if $\mathcal{D}^n$ denotes the multiset that consists of n copies of $\mathcal{D}$, then $\Delta_{\mathcal{IC}}(\mathcal{D}_1 \cup \mathcal{D}_2^n) = \Delta_{\mathcal{IC}}(\mathcal{D}_1 \cup \mathcal{D}_2)$. Intuitively, this means that a repair of a database that is obtained by merging two different database instances, is independent of the 'popularity' of those instances. In particular, merging (and then repairing) multi-sets of database instances is the same as merging sets of database instances. In contrast, every IC merging operator in the sense of [42] is majority dependent (see [42, Theorem 3.3]).

By the above discussion, it is not surprising that there are some postulates of IC merging operators that $\Delta_{\mathcal{IC}}$ does *not* satisfy. Below we consider two of them:

[10] In fact, **[M3]** is a stronger version of **[IC3]**, as the latter postulate uses bijections for defining equivalence between two sets of integrity constraints. Also, **[M4]** is a weaker version of **[IC7]**.

PROPOSITION 5.4. *There exist two database merging contexts* $\mathcal{U}_1 = (\mathfrak{D}_1, \mathcal{IC})$ *and* $\mathcal{U}_2 = (\mathfrak{D}_2, \mathcal{IC})$, *with the same domain*[11], *such that*

a) $\Delta_{\mathcal{IC}}(\mathfrak{D}_1) \wedge \Delta_{\mathcal{IC}}(\mathfrak{D}_2) \not\models \Delta_{\mathcal{IC}}(\mathfrak{D}_1 \cup \mathfrak{D}_2)$.

b) $\Delta_{\mathcal{IC}}(\mathfrak{D}_1 \cup \mathfrak{D}_2) \not\models \Delta_{\mathcal{IC}}(\mathfrak{D}_1) \wedge \Delta_{\mathcal{IC}}(\mathfrak{D}_2)$.

Proof. Consider $\mathcal{D}_1 = \{p\}$, $\mathcal{D}_2 = \{q\}$, and $\mathcal{IC} = \{(p \wedge q) \vee r\}$. Then the models of $\Delta_{\mathcal{IC}}(\{\mathcal{D}_1\})$ are the minimal valuations satisfying the formula $(\neg s_p \wedge s_q) \vee s_r$, the models of $\Delta_{\mathcal{IC}}(\{\mathcal{D}_2\})$ are the minimal valuations satisfying $(s_p \wedge \neg s_q) \vee s_r$, and the models of $\Delta_{\mathcal{IC}}(\{\mathcal{D}_1, \mathcal{D}_2\})$ are the minimal valuations that satisfy the formula $(\neg s_p \wedge \neg s_q) \vee s_r$. It follows, then, that $\{s_p : f,\ s_q : f,\ s_r : t\}$ is a model of $\Delta_{\mathcal{IC}}(\{\mathcal{D}_1\}) \wedge \Delta_{\mathcal{IC}}(\{\mathcal{D}_2\})$ that does *not* satisfy $\Delta_{\mathcal{IC}}(\{\mathcal{D}_1, \mathcal{D}_2\})$ (as it is not a *minimal* model of $(\neg s_p \wedge \neg s_q) \vee s_r$). This shows one part of the proposition. For the other part, consider the same example. Then $\Delta_{\mathcal{IC}}(\{\mathcal{D}_1\}) \wedge \Delta_{\mathcal{IC}}(\{\mathcal{D}_2\})$ is consistent (we have shown that it has a model), yet $\{s_p : f,\ s_q : f,\ s_r : f\}$ is a model of $\Delta_{\mathcal{IC}}(\{\mathcal{D}_1, \mathcal{D}_2\})$ that is *not* a model of $\Delta_{\mathcal{IC}}(\{\mathcal{D}_1\}) \wedge \Delta_{\mathcal{IC}}(\{\mathcal{D}_2\})$. □

The positive counterparts of the two properties mentioned in the last proposition (that is, where $\not\models$ is replaced by $\models$) are denoted in [42] by **[IC5]** and **[IC6]**, respectively. We note that **[IC6]** is falsified also by the merging operators of [28], mentioned before.

5.2. Belief revision

The purpose of a belief revision theory is to describe how a 'belief-base' is obtained by the revision of a belief set $\mathcal{D}$ by some new information, μ. A belief revision operator $\circ$ therefore describes the kind of information change that should be made in face of new (possibly contradicting) information. Often, the underlying operator forces only a minimal amount of data modifications, keeping the revised information 'as close as possible' to the ground information ($\mathcal{D}$) on one hand, and maintaining consistency with the new information (μ), on the other hand. This criterion, often called *the principle of minimal change*, is one of the most widely advocated postulates of belief revision theory.

Identically, $\Delta_{\mathcal{IC}}(\{\mathcal{D}\})$ may be viewed as representing the (construction of) databases that are as close as possible to $\mathcal{D}$, and do not contradict $\mathcal{IC}$. It follows, then, that a belief revision operator $\circ$ may be defined in our case as follows:

$$\mathcal{D} \circ \mu \ = \ \Delta_\mu(\{\mathcal{D}\}).$$

[11] This assumption is needed for assuring that $(\mathfrak{D}_1 \cup \mathfrak{D}_2, \mathcal{IC})$ would also be a database merging context.

The intended meaning in our context of this operator is to describe 'how to revise $\mathcal{D}$ in order to be compatible with μ'.

Once again, the fact that here we consider databases rather than belief bases (and hence the constraints DCA, UNA and CWA are implicitly enforced in our case) implies some obvious differences between the present approach and those of, e.g., [38, 42, 44]. Still, it is possible to show that the revision operator that our formalism induces, satisfies some well-known postulates in the literature of belief revision. Indeed, by postulates **[M0]** – **[M4]** it is obvious that the following properties are satisfied by $\circ$:

R0 $\mathcal{D} \circ \mu$ implies μ.

R1 $\mathcal{D} \circ \mu$ is consistent.

R2 If $\mathcal{D} \cup \mu$ is consistent, then $\mathcal{D} \circ \mu$ should be logically equivalent to $\bigwedge_{s_p \in \mathsf{switch}(\mathcal{L})} \neg s_p$. [12]

R3 If μ_1 is logically equivalent to μ_2, then $(\mathcal{D} \circ \mu_1)$ is logically equivalent to $(\mathcal{D} \circ \mu_2)$.

R4 $(\mathcal{D} \circ \mu_1) \wedge (\mathcal{D} \circ \mu_2) \models \mathcal{D} \circ (\mu_1 \cup \mu_2)$.

We note, finally, that these postulates resemble those of Katsuno and Mendelzon [38]. See [38, 42] for a more detailed discussion on the postulates of [38] and their relations to the IC merging postulates [42] and the AGM postulates of belief revision operators [1].

6. Summary and concluding remarks

This work provides further evidence for the well-known fact that in many cases a proper representation of a given problem is a major step in finding robust solutions to it. In our case, a uniform method for encoding the restoration of database consistency by signed formulae allows us to use off-the-shelf solvers for efficiently computing the desired repairs.

As shown in Corollary 3.11, the task of repairing a database is on the second level of the polynomial hierarchy, hence it is not tractable. However, despite the high computational complexity of the problem, the experimental results of Section 4 show that our method of repairing

[12] Thus, $\mathcal{D}$ should not be revised in this case.

databases by signed theories is *practically appealing*, as it allows a rapid construction of repairs for large problem instances.

There are several other existing implementations of database integration, among which are the $\mathcal{A}$system [10, 11] that is based on abductive logic programming, Subrahmanian's amalgamating system [57] that is based on multiple-valued (annotated) logic, Liberatore and Schaerf's BReLS system [45] that integrates propositional (possibly prioritized) sources without integrity constraints by a preference semantics on database interpretations, and the system for data repair of Franconi et al. [33] that is based on the language DLP$^{\omega}$ for disjunctive logic programming with constraints, supported by DLV [31]. As noted above, the main advantages of our approach in comparison to those implementations is its simplicity, generality, and the fact that it can be easily implemented by a diversity of off-the-shelf solvers. However, in its current form, this approach has some substantial drawbacks as well. One of which is that repair computation is restricted only to the Herbrand universe, while quite interesting repairs may exist outside this universe. To see this, consider the following example:

EXAMPLE 6.1. Suppose that there are two courses that cannot be taught by the available lecturers, e.g., because they do not have the necessary expertise. There are two possibilities to meet the demand that every course has a lecturer: one solution is to insert a new person with the necessary expertise for both courses; the other possibility is to insert two different persons, each one with expertise for one course.[13] Either solution lies outside the corresponding Herbrand universe, as it requires the introduction of new objects. In contrast to the abductive system presented in [10, 11] (which can compute both solutions), the solvers considered here cannot find these solutions, because grounding is inherent by the Domain Closure Assumption. Computing repairs outside the Herbrand universe is a subject for future work.

Another interesting topic for future exploration is related to definitions of *domain dependent* preference criteria among repairs (note that the preference criteria considered here, namely set inclusion $\leq_i$ and minimal cardinality $\leq_c$, are domain independent). To see the usefulness of this, consider the following example:

[13] An interesting question that arises here is which one of these solutions should be preferred. The first one involves a smaller amount of objects to be introduced, while the other solution makes less commitments, as it does not require that the same person must have expertise on two different courses.

EXAMPLE 6.2. Consider a database with the following instance

$$\left\{\begin{array}{lll} employee(Alice), & salary(Alice, 1000), & \\ employee(Bob), & salary(Bob, 1000), & director(Bob) \end{array}\right\}$$

and two integrity constraints: one specifies that every employee has a salary, and the other (violated) one says that the director should earn more money than the other employees. Now, consider two repairs of this database:

$$\begin{aligned} \mathcal{R}_1 &= (\{salary(Bob, 1100)\}, \{salary(Bob, 1000)\}), \\ \mathcal{R}_2 &= (\{\}, \{employee(Alice), salary(Alice, 1000)\}). \end{aligned}$$

$\mathcal{R}_1$, which updates the salary of Bob, has the same cardinality as $\mathcal{R}_2$, which removes Alice from the database. However, in this case one would usually prefer the former repair over the latter one.

It is quite evident that in the last example a domain dependent criterion should be incorporated for preferring $\mathcal{R}_1$ over $\mathcal{R}_2$. Such a criterion can be based on the polynomial-time computable distance function between sets of elements introduced in [52]. This distance is based on a notion of matching: each element of each set is linked with at most one element of the other set and the distance is defined as the cost of an optimal matching, that is: the sum of the distances of the matched elements that results in a minimal value (where elements that are not linked account for half of the maximal distance)[14]. Consider, for instance, a distance function d, defined by $d(x, y) = 0$ if $x = y$ and $d(x, y) = 1$ otherwise. In Example 6.2, then, the optimal matchings between the original database and the repaired database that is obtained by $\mathcal{R}_1$ links each one of $employee(Alice)$, $employee(Bob)$, $salary(Alice, 1000)$ and $director(Bob)$ to the same fact in the repaired database, and matches $salary(Bob, 1000)$ to $salary(Bob, 1100)$. The resulting distance is therefore $0 + 0 + 0 + 0 + 1 = 1$ (which is identical to the distance between $\mathsf{Insert} = \{salary(Bob, 1100)\}$ and $\mathsf{Retract} = \{salary(Bob, 1000)\}$). The cost of the optimal matching between the original database and the repaired database that is obtained by $\mathcal{R}_2$, is the cost of the two retracted elements that cannot be linked to an element in the repaired database, which is $\frac{1}{2} + \frac{1}{2} = 1$ (again, this is also the distance between $\mathsf{Insert} = \{\}$ and $\mathsf{Retract} = \{employee(Alice), salary(Alice, 1000)\}$). In other words, the distance between the original and repaired database correspond to half of the cardinality of the repair, hence the preferred repairs under this distance function correspond to the $\leq_c$-minimal repairs.

[14] Note that in case that the distance between the elements is a metric (i.e., the distance to oneself is zero, the distance is symmetric, and the triangular inequality holds), the distance function between the corresponding sets is a metric as well.

A more robust preference criterion is obtained by the distance function defined in [24]:

$$dist(P(t_1,\dots,t_m),Q(s_1,\dots,s_n)) = \begin{cases} 1 & \text{if } P \neq Q, \\ \frac{1}{2n}\sum_{i=1}^{n} d(t_i,s_i) & \text{otherwise.} \end{cases}$$

The distance between the original database of Example 6.2 and the database repaired by $\mathcal{R}_1$ in that example is therefore the same as the distance between $salary(Bob, 1000)$ and $salary(Bob, 1100)$, which is $\frac{1}{4}(0+1) = \frac{1}{4}$. Note that according to this definition, the distance between the original database and the database repaired by $\mathcal{R}_2$ is still 1. It follows, then, that now $\mathcal{R}_1$ becomes the preferred repair, as intuitively expected. Computationally, the distance between a database and its repair based on $(\mathsf{Insert}, \mathsf{Retract})$ corresponds to the distance between Insert and $\mathsf{Retract}$. Hence, we expect that it would be feasible to adopt the methods described in this paper also for computing preferred repairs when the preference criteria are domain dependent. This is, however, outside the scope of the present paper, and remains a topic for future work.

To conclude this section (as well as the whole paper), it is worth putting the current work in a broader perspective. Often, a database becomes inconsistent when it contains information that arrives from different (distributed) data sources, so (as we hinted in Section 5) the current work on restoration of database consistency is particularly relevant in the context of data integration. However, a comprehensive solution to this problem has to address some further issues that have not been considered here. One important topic is, e.g., that of *schema integration*, that is, the ability to uniformly represent and reason with independent databases that contain information about a common domain, but may have different schemas. Detailed discussions on schema matching and related aspects may be found, e.g., in [15, 17, 51, 58, 59]. Another issue that is often raised in the context of database integration is related to the management of dynamically evolving data sources. This task sometimes requires modifications of the domain of discourse and revisions of integrity constraints. When the set of integrity constraints is given in a clause form, methods of *dynamic logic programming* [2, 3] may be useful for this purpose. When the types of changes are predictable, or can be characterized in some sense, *abductive theories* (in the context of extended disjunctive logic programs) or temporal integrity constraints (in the context of *temporal databases*) can also be used in order to specify how to treat new information. See, e.g., [10, 55] for a representation of knowledge base updates by abductive theories, and [48, 56] for a discussion on temporal integrity constraints and temporal databases in logic programming based formalisms.

Acknowledgement

We would like to thank the reviewers for a careful reading of this paper and for many helpful comments and suggestions.

References

1. C.E.Alchourrón, P.Gärdenfors, and D.Makinson. On the logic of theory change: Partial meet contraction and revision function. *Journal of Symbolic Logic* 50, pp.510–530, 1985.
2. J.J.Alferes, J.A.Leite, L.M.Pereira, and P.Quaresma. Planning as abductive updating, *Proc. of the Symposium on AI Planning and Intelligent Agents* (AISB'00), pp.1–8, 2000.
3. J.J.Alferes, L.M.Pereira, H.Przymusinska, and T.C.Przymusinski. LUPS – a language for updating logic programs. *Artificial Intelligence* 138(1-2), pp.87–116, 2002.
4. M.Arenas, L.Bertossi, and J.Chomicki. Consistent query answers in inconsistent databases. *Proc. 18th ACM Symp. on Principles of Database Systems* (PODS'99), pp.68–79, 1999.
5. M.Arenas, L.Bertossi, and J.Chomicki. Answer Sets for Consistent Query Answering in Inconsistent Databases. *Theory and Prcartice of Logic Programming* 3(4–5), pp.393–424, 2003
6. O.Arieli. Paraconsistent preferential reasoning by signed quantified Boolean formulae. *Proc. 16th European Conference on Artificial Intelligence* (ECAI'04), R.López de Mántaras and L.Saitta, editors, pp.773–777, IOS Press, 2004.
7. O.Arieli and M.Denecker. Modeling paraconsistent reasoning by classical logic. *Proc. 2nd Symp. on Foundations of Information and Knowledge Systems* (FoIKS'02), T.Eiter and K.D.Schewe, editors, LNCS 2284, Springer, pp.1–14, 2002.
8. O.Arieli and M.Denecker. Reducing preferential paraconsistent reasoning to classical entailment. *Journal of Logic and Computation* 13(4), pp.557–580, 2003.
9. O.Arieli, M.Denecker, B.Van Nuffelen, and M.Bruynooghe. Database repair by signed formulae. *Proc. 3rd International Symposium on Foundations of Information and Knowledge Systems* (FoIKS'04), D.Seipel and J.M.Turull Torres, editors, LNCS 2942, Springer, pp.14–30, 2004.
10. O.Arieli, M.Denecker, B.Van Nuffelen, and M.Bruynooghe. Coherent integration of databases by abductive logic programming. *Journal of Artificial Intelligence Research,* 21, pp.245–286, 2004.
11. O.Arieli, B.Van Nuffelen, M.Denecker, and M.Bruynooghe. Coherent composition of distributed knowledge-bases through abduction. *Proc. 8th Int. Conf. on Logic Programming, Artificial Intelligence and Reasoning* (LPAR'01),

A.Nieuwenhuis and A.Voronkov, editors, LNCS 2250, Springer, pp.620–635, 2001.

12. A.Ayari and D.Basin. QUBOS: Deciding quantified Boolean logic using propositional satisfiability solvers. *Proc. 4th Int. Conf. on Formal Methods in Computer-Aided Design* (FMCAD'02), M.D.Aagaard and J.W.O'Leary, editors, LNCS 2517, Springer, pp.187–201, 2002.
13. C.Baral, S.Kraus, and J.Minker. Combining multiple knowledge bases. *IEEE Trans. on Knowledge and Data Enginnering* 3(2), pp.208–220, 1991.
14. C.Baral, S.Kraus, J.Minker, and V.S.Subrahmanain. Combining multiple knowledge bases consisting of first order theories. *Computational Intelligence* 8, pp.45–71, 1992.
15. C.Batini, M.Lenzerini, and B.B.Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys* 18(4), pp.323–364, 1986.
16. S.Benferhat, D.Dubois, S.Kaci, and H.Prade. Possibilistic merging and distance-based fusion of propositional information. *Annals of Mathematics and Artificial Intelligence* 34(1–3), pp.217–252, 2002.
17. L.Bertossi, J.Chomicki, A.Cortés, and C.Gutierrez. Consistent answers from integrated data sources. *Proc. Flexible Query Answering Systems* (FQAS'2002), A.Andreasen et al., editors, LNCS 2522, Springer, pp.71–85, 2002.
18. L.Bertossi and C.Schwind. Analytic tableau and database repairs: Foundations. *Proc. 2nd Int. Symp. on Foundations of Information and Knowledge Systems* (FoIKS'02), T.Eiter and K.D.Schewe, editors, LNCS 2284, Springer, pp.32–48, 2002.
19. P.Besnard, T.Schaub. Signed systems for paraconsistent reasoning. *Journal of Automated Reasoning* 20(1), pp.191–213, 1998.
20. P.Besnard, T.Schaub, H.Tompits, and S.Woltran. Paraconsistent reasoning via quantified Boolean formulas, part I: Axiomatizing signed systems. *Proc. 8th European Conf. on Logics in Artificial Intelligence* (JELIA'02), S.Flesca et al., editors, LNAI 2424, Springer, pp.320–331, 2002.
21. P.Besnard, T.Schaub, H.Tompits, and S.Woltran. Paraconsistent reasoning via quantified Boolean formulas, part II: Circumscribing inconsistent theories. *Proc. 7th European Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty* (ECSQARU'03), T.D.Nielsen and N.L.Zhang, editors, LNAI 2711, Springer, pp.528–539, 2003.
22. M.Cadoli, M.Schaerf, A.Giovanardi, and M.Giovanardi. An Algorithm to evaluate quantified Boolean formulae and its experimental evaluation. *Automated Reasoning* 28(2), pp.101–142, 2002.
23. M.Carlsson, G.Ottosson and B.Carlson. An open-ended finite domain constraint solver. *Proc. 9th Int. Symp. on Programming Languages, Implementations, Logics, and Programs* (PLILP'97), LNCS 1292, Springer, pp.191–206, 1997.
24. S.H.Nienhuys–Cheng. Distance between Herbrand interpretations: A measure for approximations to a target concept. *Proc. 7th Int. Workshop on Inductive Logic Programming* (ILP'97), LNCS 1297, Springer, pp.213–226, 1997.
25. M.Dalal. Investigations into a theory of knowledge base revision. *Proc. National Conference on Artificial Intelligence* (AAAI'98), AAAI Press, pp.475–479, 1988.
26. A.Darwiche and J.Pearl. On the logic of iterated belief revision. *Artificial Intelligence* 89, pp.1–29, 1997.

27. S.de Amo, W.Carnielli, and J.Marcos. A logical framework for integrating inconsistent information in multiple databases. *Proc. 2nd Int. Symp. on Foundations of Information and Knowledge Systems* (FoIKS'02), T.Eiter and K.D.Schewe, editors, LNCS 2284, Springer, pp.67–84, 2002.
28. J.Delgrande and T.Schaub. Two approaches to merging knowledge bases. *Proc. 9th European Conference on Logic in Artificial Intelligence* (JELIA'04), LNCS 3229, Springer, pp.426–438, 2004.
29. J.Delgrande, T.Schaub, H.Tompits, and S.Woltran. On Computing belief change operations using quantified Boolean formulas. *Journal of Logic and Computation* 14(6), pp.801–826, 2004.
30. U.Egly, T.Eiter, H.Tompits, and S.Woltran. Solving advanced reasoning tasks using quantified Boolean formulas. *Proc. National Conf. on Artificial Intelligence* (AAAI'00), AAAI Press, pp.417–422, 2000.
31. T.Eiter, N.Leone, C.Mateis, G.Pfeifer, and F.Scarcello. The KR system dlv: Progress report, comparisons and benchmarks. *Proc. 6th Int. Conf. on Principles of Knowledge Representation and Reasoning* (KR'98), Morgan Kaufmann Publishers, pp.406–417, 1998.
32. R.Feldmann, B.Monien, and S.Schamberger. A distributed algorithm to evaluate quantified Boolean formulae. *Proc. National Conf. on Artificial Intelligence* (AAAI'00), AAAI Press, pp. 285–290, 2000.
33. E.Franconi, A.L.Palma, N.Leone, D.Perri, and F.Scarcello. Census data repair: A challenging application of disjunctive logic programming. *Proc. 8th Internations Conference on Logic Programming, Artificial Intelligence and Reasoning* (LPAR'01), A.Nieuwenhuis and A.Voronkov, editors, LNCS 2250, Springer, pp.561–578, 2001.
34. D.Gabbay, O.Rodrigues, A.Russo. Revision by translation. In *Information, Uncertainty, and Fusion*, B.Bouchon-Meunier, R.R.Yager, and L.Zadeh, editors, Kluwer Academic Publishers, pp.3–32, 2000.
35. E.Giunchiglia, M.Narizzano, and A.Tacchella. QuBE: A system for deciding quantified Boolean formulas satisfiability. *Proc. 1st Int. Conf. on Automated Reasoning* (IJCAR'01), R.Gor, A.Leitsch, and T.Nipkow, editors, LNCS 2083, Springer, pp.364–369, 2001.
36. G.Greco, S.Greco, and E.Zumpano. A logic programming approach to the integration, repairing and querying of inconsistent databases. *Proc. 17th Int. Conf. on Logic Programming* (ICLP'01), LNCS 2237, Springer, pp.348–363, 2001.
37. S.Greco and E.Zumpano. Querying inconsistent databases. *Proc. Int. Conf. on Logic Programming and Automated Reasoning* (LPAR'2000), M.Parigot and A.Voronkov, editors, LNAI 1955, Springer, pp.308–325, 2000.
38. H.Katsuno and A.O.Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence* 52, pp.263–294,, 1991.
39. G.Kern-Isberner. The principle of conditional preservation in belief revision. *Proc. 2nd Symp. on Foundations of Information and Knowledge Systems* (FoIKS'02), T.Eiter and K.D.Schewe, editors, LNCS 2284, Springer, pp.105–129, 2002.
40. G.Kern-Isberner. A thorough axiomatization of a principle of conditional preservation in belief revision. *Annals of Mathematics and Artificial intelligence* 40(1–2), pp.127–164, 2004.
41. H.Kleine-Büning, M.Karpinski, and A.Flögel. Resolution for quantified Boolean formulas. *Journal of Information and Computation* 177(1), pp.12–18, 1995.
42. S.Konieczny and R.Pino Pérez. Merging information under constraints: a logical framework. *Journal of Logic and Computation* 12(5), pp.773-808, 2002.

43. R.Letz. Lemma and model caching in decision procedures for quantified Boolean formulas. *Proc. TABLEAUX'2002*, U.Egly and G.C.Fermüler, editors, LNAI 2381, Springer, pp.160–175, 2002.
44. P.Liberatore and M.Schaerf. Arbitration (or how to merge knowledge bases). *IEEE Transactions on Knowledge and Data Enginnering* 10, pp.76–90, 1998.
45. P.Liberatore and M.Schaerf. BReLS: A system for the integration of knowledge bases. *Proc Int. Conf. on Principles of Knowledge Representation and Reasoning* (KR'2000), Morgan Kaufmann Publishers, pp.145–152, 2000.
46. J.Lin. Integration of weighted knowledge bases. *Artificial Intelligence* 83, pp.363–378, 1996.
47. J.Lin and A.O.Mendelzon. Merging databases under constraints. *Int. Journal of Cooperative Information Systems* 7(1), pp.55–76, 1998.
48. C.A.Mareco and L.Bertossi. Specification and implementation of temporal databases in a bitemporal event calculus. *Advance in Conceptual Modeling*, LNCS 1727, Springer, pp.74–85, 1999.
49. J.McCarthy. Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence* 28, pp.89–116, 1986.
50. M.Moskewicz, C.Madigan, Y.Zhao, L.Zhang, and S.Malik. Chaff: Engineering an efficient SAT solver. *Proc. 39th Design Automation Conference* (DAC'01), pp.530–535, 2001.
51. E.Rahm, and P.A.Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal* 10(4), pp.334–350, 2001.
52. J.Ramon and M.Bruynooghe. A polynomial time computable metric between point sets. *Acta Informatica* 37(10), pp.765–780, 2001.
53. R.Reiter. On closed world databases. In: H.Gallaire and J.Minker, editors, *Logic and Databases*, Plenum Press, pp.55–76, 1978.
54. J.T.Rintanen. Improvements of the evaluation of quantified Boolean formulae. *Proc. 16th Int. Joint Conf. on Artificial Intelligence* (IJCAI'99), Morgan Kaufmann Publishers, pp.1192–1197. 1999.
55. C.Sakama and K.Inou. An abductive framework for computing knowledge base updates. *Theory and Prcartice of Logic Programming* 3(6), pp.671–715, 2003.
56. S.M.Sripada. Efficient implementation of the event calculus for temporal database applications. *Proc. Int. Conf. on Logic Programming* (ICLP'95), pp.99–113, 1995.
57. V.S.Subrahmanian. Amalgamating knowledge-bases. *ACM Transantions on Database Systems* 19(2), pp.291–331, 1994.
58. J.Ullman. Information integration using logical views. *Theoretical Computer Science* 239(2), pp.189–210, 2000.
59. B.Van Nuffelen, A.Cortés-Calabuig, M.Denecker, O.Arieli, and M.Bruynooghe. Data integration using ID-logic. *Proc. 16th Int. Conf. on Advanced Information Systems Engineering* (CAiSE'04), LNCS 3084, Springer, pp.67–81, 2004.
60. C.Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science* 3(1), pp.23–33, 1976.