Database Repair by Signed Formulae

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

[1] Department of Computer Science, The Academic College of Tel-Aviv,
Antokolski 4, Tel-Aviv 61161, Israel
oarieli@mta.ac.il
[2] Department of Computer Science, Katholieke Universiteit Leuven,
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
{marcd,bertv,maurice}@cs.kuleuven.ac.be

Abstract. We introduce a *simple* and *practically efficient* method for repairing inconsistent databases. The idea is to properly *represent* the underlying problem, and then use off-the-shelf applications for efficiently *computing* the corresponding solutions.
Given a possibly inconsistent database, we represent the possible ways to restore its consistency in terms of *signed formulae*. Then we show how the 'signed theory' that is obtained can be used by a variety of computational models for processing quantified Boolean formulae, or by constraint logic program solvers, in order to rapidly and efficiently compute desired solutions, i.e., consistent repairs of the database.

1 Introduction

In this paper we consider a uniform representation of repairs of inconsistent relational databases, that is, a general description of how to restore the consistency of databases instances that do not satisfy a given set of integrity constraints. We then show how this description can be used by a variety of computational methodologies for efficiently 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).

Reasoning with inconsistent databases has been extensively studied in the last few years, especially in the context of integrating (possibly contradicting) independent data-sources.[1] In this paper we introduce a novel representation of the repair problem as a theory that consists of what we call *signed formulae*. Then we illustrate how off-the-shelf computational systems can use the theory to solve the problem, i.e., to compute repairs of the database. 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

[1] See., e.g., [1,4,9,10,13,14,19,20,23] for more details on reasoning with inconsistent databases and further references to related works.

D. Seipel and J.M. Turull-Torres (Eds.): FoIKS 2004, LNCS 2942, pp. 14–30, 2004.

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 [12], or the answer set programming solver dlv [15]) can be used to efficiently find the required repairs.

– For finding repairs that are minimal with respect to set inclusion, satisfiability solvers on appropriate quantified Boolean formulae (QBF) can be utilized. Again, we provide a polynomial-time transformation to (signed) QBF theories, and show how QBF solvers [5,11,16,17,18,21,26] can be used to restore the database consistency.

The rest of the paper is organized as follows: In the next section we formally define the underlying problem and in Section 3 we show how to represent it by signed formulae. In Sections 4 and 5 we show how constraint solvers for logic programs and quantified Boolean formulae can be utilized for computing database repairs based on the signed theories. In Section 6 we present some experimental results, and in Section 7 we conclude with some further remarks and observations.

2 Database Repairs

Let L be a first-order language, based on a fixed database schema S and a fixed domain D. Every element of D has a unique name. A *database instance* $\mathcal{D}$ consists of atoms in the language L that are instances of the schema S. As such, every database instance $\mathcal{D}$ has a finite active domain, $\mathcal{A}(\mathcal{D})$, which is a subset of D.

A *database* is a pair $(\mathcal{D}, \mathcal{IC})$, where $\mathcal{D}$ is a database instance, and $\mathcal{IC}$, the set of *integrity constraints*, is a finite and classically consistent set of formulae in L. Given a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$, we apply to it the closed word assumption, so only the facts that are explicitly mentioned in $\mathcal{D}$ are considered true. The underlying semantics of a database $(\mathcal{D}, \mathcal{IC})$ corresponds, therefore, to the *least Herbrand model* of $\mathcal{D}$ (notation: $\mathcal{H}^{\mathcal{D}}$), i.e., the model of $\mathcal{D}$ that assigns true to all the ground instances of atomic formulae in $\mathcal{D}$, and assigns false to all the other atoms.

Given a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$, let

$$\mathcal{DB}^{\mathcal{A}} = \mathcal{D} \cup \mathcal{IC}^{\mathcal{A}} = \mathcal{D} \cup \{\rho(\psi) \mid \psi \in \mathcal{IC},\ \rho : var(\psi) \to \mathcal{A}(\mathcal{D})\},$$

where ρ is a *ground substitution* of variables to the individuals of $\mathcal{A}(\mathcal{D})$, the active domain of $\mathcal{D}$.[2] $\mathcal{DB}^{\mathcal{A}}$ is called the *Herbrand expansion* of $\mathcal{DB}$. As $\mathcal{D}$, $\mathcal{IC}$, and $\mathcal{A}(\mathcal{D})$ are all finite sets, $\mathcal{DB}^{\mathcal{A}}$ is also finite, and so $\Sigma^{\mathcal{DB}} = \{p_1, p_2, \ldots, p_n\}$, the set of the (ground) atomic formulae that appear in $\mathcal{DB}^{\mathcal{A}}$, is finite as well. In

[2] Thus, e.g., $\rho(\forall x\, \psi(x)) = \psi(p_1) \wedge ... \wedge \psi(p_n)$ and $\rho(\exists x\, \psi(x)) = \psi(p_1) \vee ... \vee \psi(p_n)$, where $p_1, \ldots, p_n$ are the elements of $\mathcal{A}(\mathcal{D})$; the transformation for other formulae is standard.

what follows we shall assume that the databases are grounded w.r.t. their active domains, therefore we shall omit the superscripts of $\mathcal{IC}^{\mathcal{A}}$ and $\mathcal{DB}^{\mathcal{A}}$.

We say that a formula ψ *follows* from a database instance $\mathcal{D}$ (notation: $\mathcal{D} \models \psi$) if the minimal Herbrand model of $\mathcal{D}$ is also a model of ψ. A database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ is *consistent* if every formula in $\mathcal{IC}$ follows from $\mathcal{D}$ (notation: $\mathcal{D} \models \mathcal{IC}$).[3]

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

Definition 2.1. An *update* of a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ is a pair $(\mathsf{Insert}, \mathsf{Retract})$, s.t. $\mathsf{Insert} \cap \mathcal{D} = \emptyset$ and $\mathsf{Retract} \subseteq \mathcal{D}$.[4] A *repair* of $\mathcal{DB}$ is an update of $\mathcal{DB}$, for which $(\mathcal{D} \cup \mathsf{Insert} \setminus \mathsf{Retract},\ \mathcal{IC})$ is a consistent database.

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

Example 2.1. Let $\mathcal{DB} = \big(\{P(a)\}, \{\forall x(P(x) \rightarrow Q(x))\}\big)$. Clearly, this database is not consistent. The Herbrand expansion of $\mathcal{DB}$ is $(\{P(a)\}, \{P(a) \rightarrow Q(a)\})$, and it has three repairs, namely $\mathcal{R}_1 = (\{\}, \{P(a)\})$, $\mathcal{R}_2 = (\{Q(a)\}, \{\})$, and $\mathcal{R}_3 = (\{Q(a)\}, \{P(a)\})$ that correspond, respectively, to removing $P(a)$ from the database, inserting $Q(a)$ to the database, and performing both actions simultaneously.

Note that as the underlying semantics is determined by Herbrand interpretations, the Domain Closure Assumption[5] is implicit here, and should be regarded as another constraint that should be satisfied by every repair. Therefore, e.g., $(\{Q(b)\}, \{P(a)\})$ is *not* a repair of $\mathcal{DB}$ in this case, for any $b \neq a$. Another implicit assumption, induced by the use of Herbrand semantics, is that Clark's equality axioms are satisfied, and so the elements of $\Sigma^{\mathcal{DB}}$ are all different.

As the example above shows, there are many ways to repair a given database, some of them may not be very natural or sensible. It is usual, therefore, to specify some preference criterion on the possible repairs, and to apply only those 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* [1,4,9,10,14,19,20], i.e.,

$(\mathsf{Insert}, \mathsf{Retract}) \leq_i (\mathsf{Insert}', \mathsf{Retract}')$, if $\mathsf{Insert} \cup \mathsf{Retract} \subseteq \mathsf{Insert}' \cup \mathsf{Retract}'$,

or *minimal cardinality* [4,13,23], i.e.,

$(\mathsf{Insert}, \mathsf{Retract}) \leq_c (\mathsf{Insert}', \mathsf{Retract}')$, if $|\mathsf{Insert}| + |\mathsf{Retract}| \leq |\mathsf{Insert}'| + |\mathsf{Retract}'|$.

[3] That is, there is no integrity constraint that is violated in $\mathcal{D}$.

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

[5] Namely, that the domain of every variable is in the set $\Sigma^{\mathcal{DB}}$ of the ground atoms that appear in $\mathcal{DB}$.

Both criteria above reflect the intuitive feeling that a 'natural' way to repair an inconsistent database should require some minimal amount of changes, therefore the recovered data 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 $\mathcal{R}_2$ of Example 2.1 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.

3 Representation of Repairs by Signed Formulae

In what follows we represent (preferred) repairs in terms of what we call 'signed formulae'. Then we incorporate corresponding solvers in order to compute the repairs.

For every (ground) atom $p \in \Sigma^{\mathcal{DB}}$ we introduce a new atom, s_p, intuitively understood as 'switch p', or 'change the status of p', that is, s_p holds iff $p \in \mathsf{Insert} \cup \mathsf{Retract}$. For every integrity constraint $\psi \in \mathcal{IC}$ we define a new formulae, $\overline{\psi}$, obtained from ψ by simultaneously substituting every appearance of an atom p by a corresponding expression τ_p that is defined as follows:

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

The formula $\overline{\psi} = \psi[\tau_{p_1}/p_1, \ldots, \tau_{p_m}/p_m]$ (i.e., the simultaneous substitution in ψ of all the atomic formulae p_i, $1 \leq i \leq m$, by their 'signed expressions' τ_{p_i}) is called the *signed formula* that is obtained from ψ.

Given a repair $\mathcal{R} = (\mathsf{Insert}, \mathsf{Retract})$ of a database $\mathcal{DB}$, define a valuation $\nu^{\mathcal{R}}$ on $\{s_p \mid p \in \Sigma^{\mathcal{DB}}\}$ 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 on $\{s_p \mid p \in \Sigma^{\mathcal{DB}}\}$ *induces* a database update $\mathcal{R}^{\nu} = (\mathsf{Insert}, \mathsf{Retract})$, where $\mathsf{Insert} = \{p \notin \mathcal{D} \mid \nu(s_p) = t\}$ and $\mathsf{Retract} = \{p \in \mathcal{D} \mid \nu(s_p) = t\}$.[6] Obviously, these mappings are the inverse of each other.

Example 3.1. Let $\mathcal{DB} = (\{p\}, \{p \rightarrow q\})$ be a ground representation of the database considered in Example 2.1. In this case, the sign formula of $\psi = p \rightarrow q$ is $\overline{\psi} = \neg s_p \rightarrow s_q$, 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 2.1). The next theorem shows that this is not a coincidence.

[6] Clearly, $\mathcal{R}^{\nu}$ is an update of $\mathcal{DB}$, but it is not necessarily a repair of $\mathcal{DB}$ (see Definition 2.1).

Theorem 3.1. *Let $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ be a database. Denote: $\overline{\mathcal{IC}} = \{\overline{\psi} \mid \psi \in \mathcal{IC}\}$.*

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 $\mathcal{H}^{\mathcal{D}^{\mathcal{R}}}$ be the least Herbrand model of $\mathcal{D}^{\mathcal{R}}$, and let $\psi \in \mathcal{IC}$. 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 $p \in \Sigma^{\mathcal{DB}}$, $\nu^{\mathcal{R}}(\overline{p}) = \mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(p)$. Indeed,

– $p \in \mathcal{D} \setminus \mathsf{Retract} \Rightarrow p \in \mathcal{D}^{\mathcal{R}} \Rightarrow \nu^{\mathcal{R}}(\overline{p}) = \nu^{\mathcal{R}}(\neg s_p) = \neg\nu^{\mathcal{R}}(s_p) = \neg f = t = \mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(p)$.
– $p \in \mathsf{Retract} \Rightarrow p \in \mathcal{D} \setminus \mathcal{D}^{\mathcal{R}} \Rightarrow \nu^{\mathcal{R}}(\overline{p}) = \nu^{\mathcal{R}}(\neg s_p) = \neg\nu^{\mathcal{R}}(s_p) = \neg t = f = \mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(p)$.
– $p \in \mathsf{Insert} \Rightarrow p \in \mathcal{D}^{\mathcal{R}} \setminus \mathcal{D} \Rightarrow \nu^{\mathcal{R}}(\overline{p}) = \nu^{\mathcal{R}}(s_p) = t = \mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(p)$.
– $p \notin \mathcal{D} \cup \mathsf{Insert} \Rightarrow p \notin \mathcal{D}^{\mathcal{R}} \Rightarrow \nu^{\mathcal{R}}(\overline{p}) = \nu^{\mathcal{R}}(s_p) = f = \mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(p)$.

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 2.1, it is obviously an update. 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 least Herbrand model of $\mathcal{D}^{\mathcal{R}}$. 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 only the base step, that is: for every $p \in \Sigma^{\mathcal{DB}}$, $\mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(p) = \nu(\overline{p})$:

– $p \in \mathcal{D} \setminus \mathsf{Retract} \Rightarrow p \in \mathcal{D}^{\mathcal{R}}$, $\nu(s_p) = f \Rightarrow \mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(p) = t = \neg\nu(s_p) = \nu(\neg s_p) = \nu(\overline{p})$.
– $p \in \mathsf{Retract} \Rightarrow p \in \mathcal{D} \setminus \mathcal{D}^{\mathcal{R}}$, $\nu(s_p) = t$, $\Rightarrow \mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(p) = f = \neg\nu(s_p) = \nu(\neg s_p) = \nu(\overline{p})$.
– $p \in \mathsf{Insert} \Rightarrow p \in \mathcal{D}^{\mathcal{R}} \setminus \mathcal{D}$, $\nu(s_p) = t$, $\Rightarrow \mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(p) = t = \nu(s_p) = \nu(\overline{p})$.
– $p \notin \mathcal{D} \cup \mathsf{Insert} \Rightarrow p \notin \mathcal{D}^{\mathcal{R}}$, $\nu(s_p) = f$, $\Rightarrow \mathcal{H}^{\mathcal{D}^{\mathcal{R}}}(p) = f = \nu(s_p) = \nu(\overline{p})$. □

The last 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 $\mathcal{DB}$.

Example 3.2. Consider again the (grounded) database of Examples 2.1 and 3.1. The corresponding signed formula $\overline{\psi} = s_p \vee s_q$ has three models $\{s_p : t, s_q : f\}$, $\{s_p : f, s_q : t\}$, and $\{s_p : t, s_q : t\}$.[7] These models induce, respectively, three pairs, $(\{\}, \{p\})$, $(\{q\}, \{\})$, and $(\{q\}, \{p\})$, which are the repairs of $\mathcal{DB}$ (cf. Example 2.1).

[7] We are denoting here by $p : x$ the fact that the atom p is assigned the value x by the corresponding valuation.

4 Computing Preferred Repairs by Model Generation

In this section 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 and $\leq_i$-preferred repairs. The experimental results are presented in Section 6.

4.1 Computing $\leq_c$-Preferred Repairs

By Theorem 3.1, the repairs of a database correspond exactly to the models of the signed theory. It is straightforward to see that $\leq_c$-preferred repairs of $\mathcal{DB}$ (i.e., those with minimal cardinality) correspond to models of $\overline{\mathcal{IC}}$ that minimize the number of t-assignments of the atoms s_p. Hence, the problem is to find Herbrand models for $\overline{\mathcal{IC}}$ with minimal cardinality (called $\leq_c$-minimal Herbrand models).

Theorem 4.1. *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 a $\leq_c$-preferred repair of $\mathcal{DB}$, then $\nu^{\mathcal{R}}$ is a $\leq_c$-minimal Herbrand model of $\overline{\mathcal{IC}}$.*

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

We discuss two techniques to compute $\leq_c$-minimal Herbrand models. The first approach is to use a finite domain CLP solver. Encoding the computation of $\leq_c$-preferred repair using a finite domain constraint solver is a straightforward process. The 'switch 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 all the signed variables that are assigned the value '1', and asks for a solution with a minimal value for `Sum`.

Example 4.1. Below is a code for repairing the database of Example 3.2 with Sicstus Prolog finite domain constraint solver CLP(FD) [12][8].

```
domain([Sp,Sq],0,1),                  % domain of the signed atoms
Sp #\/ Sq,                            % the signed theory
sum([Sp,Sq],#=,Sum),                  % Sum = num of vars with val 1
minimize(labeling([],[Sp,Sq]),Sum). % find a solution 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 $\nu_2 = \{s_p : f, s_q : t\}$. Thus, the two $\leq_c$-minimal repairs here are $(\{\}, \{p\})$ and $(\{q\}, \{\})$, which indeed insert or retract exactly one atomic formula.

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

A second approach is to use the disjunctive logic programming system DLV [15]. 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 4.2. Below is a code for repairing the database of Example 3.2 with DLV.

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

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 $\mathcal{R}_2 = (\{q\}, \{\})$.

4.2 Computing $\leq_i$-Preferred Repairs

The $\leq_i$-preferred repairs of a database correspond to minimal Herbrand models with respect to set inclusion of the signed theory $\overline{\mathcal{IC}}$. 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). We consider here three simple techniques to compute a $\leq_i$-preferred repair. In the next section we consider another more complex method.

I. One technique, mentioned already in the previous section, is to transform $\overline{\mathcal{IC}}$ to clausal form and use the DLV system. In this case the weak constraints are not needed.

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 4.1. *If the above strategy for value selection is used, then the first computed model is provably a $\leq_i$-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 6.

III. A third technique considered here uses SAT-solvers. SAT-solvers, such as zChaff [25], do not compute directly 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 $\mathcal{T}$. This approach has been tested using the SAT solver zChaff [25]; the results are discussed in Section 6.

5 Computing $\leq_i$-Preferred Repairs by QBF Solvers

In this section we show how solvers for quantified Boolean formulae (QBFs) 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 [5], EVALUATE [11], QUIP [16], QSOLVE [17], QuBE [18], QKN [21], SEMPROP [22], and DECIDE [26], 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.

5.1 Quantified Boolean Formulae

Quantified Boolean formulae (QBFs) are propositional formulae extended with quantifiers $\forall, \exists$ over propositional variables. 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 $\nu_{\text{at}} : \Sigma^{\mathcal{DB}} \cup \{\mathsf{t}, \mathsf{f}\} \to \{t, f\}$ s.t. $\nu(\mathsf{t}) = t$ and $\nu(\mathsf{f}) = f$, a valuation ν on QBFs is recursively defined as follows:

$\nu(p) = \nu_{\text{at}}(p)$ for every $p \in \Sigma^{\mathcal{DB}} \cup \{\mathsf{t}, \mathsf{f}\}$,
$\nu(\neg\psi) = \neg\nu(\psi)$,
$\nu(\psi \circ \phi) = \nu(\psi) \circ \nu(\phi)$, where $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$,
$\nu(\forall p\, \psi) = \nu(\psi[\mathsf{t}/p]) \wedge \nu(\psi[\mathsf{f}/p])$,
$\nu(\exists p\, \psi) = \nu(\psi[\mathsf{t}/p]) \vee \nu(\psi[\mathsf{f}/p])$.

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: $\Gamma \vdash \Psi$) 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, $\nu_1(p) \rightarrow \nu_2(p)$ is true. We shall also write $\nu_1 < \nu_2$ to denote that $\nu_1 \leq \nu_2$ and $\nu_2 \not\leq \nu_1$.

5.2 Representing $\leq_i$-Preferred Repairs by Signed QBFs

It is well-known that quantified Boolean formulae can be used for representing circumscription [24], thus they properly express logical minimization [7,8]. 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 $\Psi_{\mathcal{DB}}$:

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

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 a $\leq_i$-preferred repair of $\mathcal{DB}$, and in general $\Psi_{\mathcal{DB}}$ represents $\leq_i$-minimality.

Example 5.1. With the database $\mathcal{DB}$ of Examples 2.1, 3.1, and 3.2, $\overline{\mathcal{IC}} \cup \Psi_{\mathcal{DB}}$ 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 5.1 below, these are the only $\leq_i$-preferred repairs of $\mathcal{DB}$.

Theorem 5.1. *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 3.1, $\nu^{\mathcal{R}}$ is a model of $\overline{\mathcal{IC}}$. Since Theorem 3.1 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 3.1 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 $\mathcal{DB}$. □

Note 5.1. (*Complexity results*) A skeptical (conservative) approach to query answering is considered, e.g., in [1,19], where 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 $\mathcal{DB}$. A credulous approach to the same problem evaluates queries with respect to *some* $\leq_i$-preferred repair of $\mathcal{DB}$. Theorem 5.1 implies the following upper complexity bounds for these approaches:

Corollary 5.1. *Credulous query answering lies in Σ_2^P, and skeptical query answering is in Π_2^P.*

Proof. By Theorem 5.1, credulous query answering is equivalent to satisfiability checking for $\overline{\mathcal{IC}} \cup \Psi_{\mathcal{DB}}$, and conservative query answering is equivalent to entailment checking for the same theory (see also Corollary 5.2 below). 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 5.1. [27] *Given a propositional formula ψ, whose atoms are partitioned into $i \geq 1$ sets $\{p_1^1, \ldots, p_{m_1}^1\}, \ldots, \{p_1^i, \ldots, p_{m_i}^i\}$, deciding whether*

$$\exists p_1^1, \ldots, \exists p_{m_1}^1, \forall p_1^2, \ldots, \forall p_{m_2}^2, \ldots, \mathsf{Q} p_1^i, \ldots, \mathsf{Q} p_{m_i}^i \psi$$

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

$$\forall p_1^1, \ldots, \forall p_{m_1}^1, \exists p_1^2, \ldots, \exists p_{m_2}^2, \ldots, \mathsf{Q} p_1^i, \ldots, \mathsf{Q} p_{m_i}^i \psi$$

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

As shown, e.g., in [19], the complexity bounds specified in the last corollary are strict, i.e., these decision problems are hard for the respective complexity classes.

Note 5.2. (*Consistent query answering*) Another consequence of Theorem 5.1 is that the conservative approach to query answering [1,19] may be represented in our context in terms of a consequence relation as follows:

Corollary 5.2. *$\mathcal{Q}$ is a consistent query answer of a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ in the sense of* [1,19] *iff $\overline{\mathcal{IC}} \cup \Psi_{\mathcal{DB}} \vdash \mathcal{Q}$.*

The last corollary and Section 4.2 provide, therefore, some additional methods for consistent query answering, all of them are based on signed theories.

6 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 [2,3,6,7]. In the area of database integration, signed variables are used in [19], and have a similar intended meaning as in our case. In [19], 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 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 the schema (`teacher_name`), *course* of the schema (`course_name`), and *teaches* of the schema (`teacher_name`, `course_name`). Also, the following two integrity constraints were specified:

ic1 A course is given by one teacher:

$$\forall X\, \forall Y\, \forall Z \Big(\big(\mathit{teacher}(X) \ \wedge\ \mathit{teacher}(Y) \ \wedge\ \mathit{course}(Z) \wedge\ \mathit{teaches}(X,Z) \wedge \mathit{teaches}(Y,Z) \big) \ \rightarrow\ X = Y \Big)$$

ic2 Each teacher gives at least one course:

$$\forall X \Big(\mathit{teacher}(X) \ \rightarrow\ \exists Y \big(\mathit{course}(Y) \wedge \mathit{teaches}(X,Y) \big) \Big)$$

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 equals the number of teachers.
4. Databases with $\mathcal{IC} = \{\mathbf{ic1}, \mathbf{ic2}\}$ and with 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 [15] (release 2003-05-16), CLP(FD) [12] (version 3.10.1).
- **QBF-solvers:**
  SEMPROP [22] (release 24.02.02), QuBE-BJ [18] (release 1.3), DECIDE [26].
- **SAT-solvers:**
  A minimal-model generator based on zChaff [25].

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 1–4 show the results for providing the first answer.[9]

The results of the first benchmark (Table 1) 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. Also, we encountered some space limitation problems and a bug[10] in DECIDE, and this discouraged us from using it in our experiments.

Another observation from Tables 1–4 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

[9] Times are in 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.

[10] For the unsatisfiable QBF $\exists xy \forall uv((x \vee y) \wedge (u \vee v))$, the answer $x = 1$ and $y = 0$ is returned. The system developers were notified about this and the bug is being fixed.

Table 1. 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 2. Results for test case 2

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

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 4, detecting unsatisfiability requires a considerable amount of time, even for small instances.

Table 3. 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 4. 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					

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 4.2).
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).
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.

7 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 5.1, 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 6 show that our method of repairing databases by signed theories is *practically appealing*, as it allows a rapid construction of repairs for large problem instances.

References

1. 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.
2. 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.
3. O.Arieli and M.Denecker. Reducing preferential paraconsistent reasoning to classical entailment. *Journal of Logic and Computation* 13(4), pp.557–580, 2003.
4. 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.
5. 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.
6. P.Besnard, T.Schaub. Signed systems for paraconsistent reasoning. *Journal of Automated Reasoning* 20(1), pp.191–213, 1998.
7. 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.

8. 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.
9. L.Bertossi, J.Chomicki, A.Cortes, 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.
10. 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.
11. 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.
12. 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, 1997.
13. M.Dalal. Investigations into a theory of knowledge base revision. *Proc. National Conference on Artificial Intelligence* (AAAI'98), AAAI Press, pp.475–479, 1988.
14. 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.
15. 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.
16. U.Egly, T.Eiter, H.Tompits, and S.Woltran. Solving advanced reasoning tasks using quantified Boolean formulas. *Proc. National Conf. on Artificial Intteligence* (AAAI'00), AAAI Press, pp.417–422, 2000.
17. R.Feldmann, B.Monien, and S.Schamberger. A distributed algorithm to evaluate quantified Boolean formulae. *Proc. National Conf. on Artificial Intteligence* (AAAI'00), AAAI Press, pp. 285–290, 2000.
18. 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.
19. 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.
20. 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.
21. H.Kleine-Büning, M.Karpinski, and A.Fögel. Resolution for quantified Boolean formulas. *Journal of Information and Computation* 177(1), pp.12–18, 1995.
22. 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, pp.160–175, 2002.

23. 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.
24. J.McCarthy. Applications of circumscription to formalizing common-Sense knowledge. *Artificial Intelligence* 28, pp.89–116, 1986.
25. M.Moskewicz, C.Madigan, Y.Zhao, L.Zhang, and S.Malik. Chaff: Engineering an efficient SAT solver. *Proc. 39th Design Automation Conference*, 2001.
26. 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.
27. C.Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science* 3(1), pp.23–33, 1976.