Distance Semantics for Database Repair

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

[1] Department of Computer Science, The Academic College of Tel-Aviv, Israel
oarieli@mta.ac.il
[2] Department of Computer Science, Katholieke Universiteit Leuven, Belgium
{marcd,maurice}@cs.kuleuven.ac.be

Abstract. In many scenarios, a database instance violates a given set of integrity constraints. In such cases, it is often required to *repair* the database, that is, to restore its consistency. A primary motif behind the repairing approaches is the *principle of minimal change*, which is the aspiration to keep the recovered data as faithful as possible to the original (inconsistent) database. In this paper, we represent this qualitative principle quantitatively, in terms of distance functions and some underlying metrics, and so introduce a general framework for repairing inconsistent databases by distance-based considerations. The uniform way of representing repairs and their semantics clarifies the essence behind several approaches to consistency restoration in database systems, helps to compare the underlying formalisms, and relates them to existing methods of defining belief revision operators, merging data sets, and integrating information systems.

1 Introduction

Inconsistency of constraint data-sources is a widespread phenomenon. There are many reasons for that, among which are human errors in information handling or understanding, conflicts between the actual data and external integrity constraints, integration of contradicting data-sources, new constraints that are enforced on pre-existing data, and so forth. In such cases it is usually required to 'repair' the information, that is, restore its consistency. This task is usually closely related to the *principle of minimal change*, which is the aspiration to reach consistency by a minimal amount of modifications in the 'spoiled' data. To illustrate this, consider the following simple example:

Example 1. Consider a database with two data facts $\mathcal{D} = \{p, r\}$, and an integrity constraint $\mathcal{IC} = p \rightarrow q$. Under the closed world assumption [42, 43], stating that each atomic formula that does not appear in $\mathcal{D}$ is false, this database is clearly inconsistent, as $\mathcal{IC}$ is violated. Two ways of restoring consistency in this case are by inserting q to $\mathcal{D}$ or deleting p from $\mathcal{D}$. Moreover, assuming that integrity constraints cannot be altered, these are the most compact ways of repairing this database, in the sense that any other solution requires a larger amount of changes (i.e., insertions or retractions) in $\mathcal{D}$.

Consistency restoration by minimal change may be traced back to [16] and [49]. In the context of database systems, this notion was introduced by [1], and then considered by many others, including [2, 4, 5, 9, 10, 12, 18, 26, 27, 35, 37, 46, 47]. Some implementations of these methods are reported in [4, 23, 25, 33]. Despite their syntactic and semantic differences, as well as the different notions of repair used by different consistency maintenance formalisms, the rationality behind all these methods is of keeping the 'recovered' data 'as faithful as possible' to the original (inconsistent) data.

A common way of deriving minimal change is by distance considerations. This approach is very common, for instance, in belief revision [19, 28, 31, 39, 44], where the belief states

of the reasoner before and after the revision are kept as close to each other as possible. Closeness is specified in terms of distance semantics, using appropriate metrics. In this paper, following the same idea, we identify distance-based semantics at the heart of several repairing methods, and introduce a corresponding framework for data repair. In this respect, we follow Bertossi's remark in [7], that

> Identifying general properties of the reasonable repair semantics [...] is a very important research direction. Unifying principles seem to be necessary at this stage in order to have a better understanding of consistent query answering.

Although there are several approaches to database repair that do not involve distance considerations (most notably, those that are based on *set inclusion*, e.g. [1, 14, 26, 47]), our framework does capture all the known *quantitative* methods for database repair, including those that are based on *minimal cardinality* and similarity measures induced by *tableaux homomorphism*. Moreover, as the same distance-based considerations are also the nucleus of many approaches for belief revision and data integration, this work is not restricted to databases only, and can be easily generalized to other paradigms in which reasoning with inconsistency is involved.

The rest of this paper, which is a revised and extended version of the paper in [3], is organized as follows: In Section 2 we give a general representation of consistency restoration in database systems as a distance minimization problem. In Section 3 we consider different distance-based approaches to database repairing, and incorporate the notion of optimal matching (between the spoiled and the recovered data) for generalizing some existing repairing methods and defining several new ones. In Section 4 we show how our framework can be used also for merging independent data-sources. In Section 5 we consider some related works on database repairing and in Section 6 we conclude.

2 Database Repair as a Distance Minimization Problem

2.1 Motivation

We begin with an informal description of the problem at hand and how to handle it. For this, consider the following set of ground facts:

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

and two integrity constraints: One says that every employee has a salary, and the other constraint specifies that a director should earn more money than any other employee. Now, applying here the closed world assumption, we conclude that Bob is not a director. On the other hand, Bob earns the same amount of money as Alice, who *is* a director, so the second integrity constraint is violated.

One way of restoring the database consistency is by changing Alice's salary. A compact way of doing so, assuming that the new salary is unknown, is by *updating* the set of facts above, so that $\texttt{salary}(\mathit{Alice}, x)$ will replace $\texttt{salary}(\mathit{Alice}, 1000)$. Implicitly, the variable of this update is existentially quantified, as it represents a particular but unknown constant in the language of the problem domain. Note, however, that updates in which the value of x is less than or equal to 1000 are not useful, as they still violate one of the integrity constraints. To properly repair the database in this case one has to impose the restriction that $x > 1000$. This characterization of one way of restoring the database consistency is called a *potential repair* of the database.

Yet, not every potential repair necessarily describes an *optimal* way of repairing the database. For instance, another way of repairing the database above is by increasing the salary of Alice *and* decreasing the salary of Bob at the same time. Clearly, this potential repair is inferior to the previous one, as it requires additional modifications. Preference among the potential repairs is performed by a distance semantics that captures the intuition that the repaired database should be 'as close as possible' to the original, inconsistent database.

In what follows, we formalize this process and consider some useful metrics for making plausible choices among the potential repairs of a given database.

2.2 Repairing inconsistent databases

In the sequel, we denote by $\mathcal{L}$ a first-order language consisting of a finite set $\mathcal{P}$ of predicate symbols and a (possibly infinite) set $\mathfrak{C}$ of constants. $\mathcal{P}$ includes the constants $\mathtt{t}$ and $\mathtt{f}$, the binary predicates $=$ and $\neq$, and the arithmetic predicates $<$, $>$, $\leq$ and $\geq$. The atomic formulas in $\mathcal{L}$ are constructed from the predicates in $\mathcal{P}$ over tuples of variables and constants from $\mathfrak{C}$. We denote this set by $\mathsf{Atoms}(\mathcal{L})$, or just Atoms. A formula without variables is called ground. A ground atomic formula is sometimes called a fact. We denote the set of ground atoms by $\mathsf{GAtoms}(\mathcal{L})$, or just GAtoms. Compound formulas in $\mathcal{L}$ are constructed from the atomic formulas using the standard recursive rules for $\neg, \wedge, \vee, \forall$ and $\exists$. An $\mathcal{L}$-structure (interpretation) ν consists of a domain $\mathsf{Dom}(\nu)$, a domain element for every constant in $\mathfrak{C}$, and an n-ary relation on $\mathsf{Dom}(\nu)$ for every n-ary predicate symbol in $\mathcal{P}$. The logical symbols $\mathtt{t}$ and $\mathtt{f}$ are interpreted, respectively, by true and false; $=$ and $\neq$ by the identity relation and its negation; and the arithmetic predicates $<$, $>$, $\leq$ and $\geq$ by their standard arithmetic meaning. Herbrand interpretations for $\mathcal{L}$ are a particular kind of $\mathcal{L}$-structures, where the domain is equal to $\mathfrak{C}$ and the interpretation of the elements of $\mathfrak{C}$ is given by the identity function. Herbrand models of a set $\mathcal{S}$ of formulas in $\mathcal{L}$ are Herbrand interpretations for $\mathcal{L}$ that satisfy each formula in $\mathcal{S}$.

Definition 1 (databases). A *database* $\mathcal{DB}$ is a pair $(\mathcal{D}, \mathcal{IC})$, where $\mathcal{D}$ is a finite set of ground atomic facts, and $\mathcal{IC}$ is a finite and consistent set of formulas in $\mathcal{L}$.

The set $\mathcal{D}$ in the definition above is called the *database instance* of $\mathcal{DB}$. Its meaning is determined by the least Herbrand model of the conjunction of the facts in D. The formulas in $\mathcal{IC}$ are called *integrity constraints*. These formulas specify conditions that should be satisfied by the (least Herbrand model of the) database instance. We denote this by $\mathcal{D} \models \mathcal{IC}$. The set $\mathfrak{C}(\mathcal{DB})$ of the constants that appear in $\mathcal{D}$ and in $\mathcal{IC}$ is called the *active domain* of $\mathcal{DB}$. Note that as both $\mathcal{D}$ and $\mathcal{IC}$ are finite sets, $\mathfrak{C}(\mathcal{DB})$ is a *finite* subset of $\mathfrak{C}$. In what follows we denote by $\mathsf{Atoms}(\mathcal{DB})$ the atomic formulas that are constructed from the predicates in $\mathcal{P}$ over tuples of variables and constants from $\mathfrak{C}(\mathcal{DB})$. The subset of ground atoms is denoted $\mathsf{GAtoms}(\mathcal{DB})$.

Definition 2 (consistency). A database $(\mathcal{D}, \mathcal{IC})$ is *consistent* if $\mathcal{D} \models \mathcal{IC}$.

When a database is not consistent, one or more integrity constraints are violated, and so it is usually required to 'repair' the database, i.e., restore its consistency. We require that the repaired information would be 'as close as possible' to the original one. Implicitly, then, this criterion involves distance-based considerations and a corresponding metric. Below, we recall the relevant definition.

Definition 3 (distance functions). A total function $d : U \times U \to \mathbb{R}^+$ is called a *pseudo distance* on U if it is symmetric ($\forall u, v \in U \; d(u, v) = d(v, u)$) and preserves identity ($\forall u, v \in U \; d(u, v) = 0$ iff $u = v$). A *distance function* on U is a pseudo distance on U that satisfies the triangular inequality ($\forall u, v, w \in U \; d(u, v) \leq d(u, w) + d(w, v)$).

As we have noted in Section 2.1, a description of database repair can include non-ground atoms. The intuitive meaning of variables in a tuple is a substitution of faulty values in the original tuple by correct but unknown new values. The set of valid substitutions is represented by constraints. This allows us to modify only erroneous fragments of tuples instead of whole tuples. A constraint is defined as follows:

Definition 4 (constraints). A *constraint* $\mathcal{C}$ (for a database $\mathcal{DB}$) is a finite set of formulas of the form $x\Theta c$ or $x\Theta y$, where x, y are variables, c is a constant in $\mathfrak{C}(\mathcal{DB})$, and Θ is a ('built-in') predicate in $\{<, >, =, \neq, \leq, \geq\}$ such that for some set of values $\bar{c}$ of $\mathfrak{C}$, all constraint atoms in $\mathcal{C}[\bar{c}/\bar{x}]$ are satisfied. We denote by $\bigwedge \mathcal{C}$ the conjunction of constraints in $\mathcal{C}$.[3]

Definition 5 (updates). Let d be a pseudo distance function on 2^{GAtoms}. An *update* of a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ (w.r.t. d) is a pair $\langle \mathcal{U}[\overline{x}], \mathcal{C}[\overline{x}] \rangle$, where $\mathcal{U}[\overline{x}]$ is a subset of $\mathsf{Atoms}(\mathcal{DB})$ containing the variables $\overline{x}$, and $\mathcal{C}[\overline{x}]$ is a consistent set of constraints over the variables $\overline{x}$, such that for every solution $\overline{c}$ of $\mathcal{C}[\overline{x}]$, the distance between the original database instance $\mathcal{D}$ and $\mathcal{U}[\overline{c}/\overline{x}]$ is the same.

In what follows, when we refer to the distance $d(\mathcal{D}, \mathcal{U}[\overline{x}])$ between an original database $\mathcal{D}$ and an update component $\mathcal{U}[\overline{x}]$ of an update $\langle \mathcal{U}[\overline{x}], \mathcal{C}[\overline{x}] \rangle$, we mean $d(\mathcal{D}, \mathcal{U}[\overline{c}/\overline{x}])$, where $\overline{c}$ is a solution of $\mathcal{C}[\overline{x}]$.

An update represents a number of possible ways to modify a database (all of which are equally distant from the original database instance). A particular (ground) update is obtained by substituting the variables $\overline{x}$ in $\mathcal{U}[\overline{x}]$ by a solution of $\mathcal{C}[\overline{x}]$ (that is, a substitution for $\overline{x}$ that makes $\bigwedge \mathcal{C}$ true).

Note 1. Assuming that there is a global upper bound on the number of variables that may appear in an update, the number of possible updates is finite. This is so, since the number of atomic formulas in $\mathsf{Atoms}(\mathcal{DB})$ is finite (modulo equivalence), and as the number of variables in $\overline{x}$ is globally bounded, both the set of the possible $\mathcal{U}[\overline{x}]$, and the number of the possible constraints $\mathcal{C}[\overline{x}]$ that can be imposed on the same $\mathcal{U}[\overline{x}]$, are finite.

Note 2. As usual, there is a trade-off between the expressivity of database updates and their computability. Thus, while one may consider more expressive forms of representing constraints on updates in a more general or compact way, this may increase the computational complexity. For instance, by introducing disjunctions and representing constraints in, e.g., conjunctive normal form, the updates

$$\Big\{\big\langle\{P(a,c), P(a,d)\}, \emptyset\big\rangle, \quad \big\langle\{P(a,c), P(b,c)\}, \emptyset\big\rangle, \quad \big\langle\{P(a,d), P(b,c)\}, \emptyset\big\rangle\Big\}$$

may be also represented, e.g., in either of the following ways, in which disjunctions are used for expressing constraints:

$$\Big\{\big\langle\{P(a,x), P(b,c)\}, \{x = c \vee x = d\}\big\rangle, \quad \big\langle\{P(a,c), P(a,d)\}, \emptyset\big\rangle\Big\},$$

$$\Big\{\big\langle\{P(x,c), P(a,d)\}, \{x = a \vee x = b\}\big\rangle, \quad \big\langle\{P(a,c), P(b,c)\}, \emptyset\big\rangle\Big\},$$

$$\Big\{\big\langle\{P(a,x), P(b,c)\}, \{x = c \vee x = d\}\big\rangle, \quad \big\langle\{P(x,c), P(a,d)\}, \{x = a \vee x = b\}\big\rangle\Big\}.$$

Thus, superfluous (i.e., equivalent) representations of the same update should be identified and ruled out to avoid duplicate database repairs (see below).

[3] Note that $\bigwedge \emptyset = \mathsf{t}$.

Note that the notion of an update is only loosely related to the specific database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ at hand. A more robust link will be obtained through the following two definitions:

- Definition 6 (potential repairs) makes sure that the relevant updates will be only those in which $\mathcal{IC}$ is satisfiable in every solution, and
- Definition 7 (pre-repairs) assures that the potential repairs will be 'as close as possible' (in terms of distance functions) to $\mathcal{D}$.

Definition 6 (potential repairs). A *potential repair* of $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ is an update $\mathcal{R} = \langle \mathcal{U}[\overline{x}], \mathcal{C}[\overline{x}] \rangle$ of $\mathcal{DB}$ such that for every solution $\overline{c}$ of $\mathcal{C}[\overline{x}]$, it holds that $\mathcal{U}[\overline{c}/\overline{x}] \models \mathcal{IC}$ (in the sense of Definition 2). The set of all the potential repairs of $\mathcal{DB}$ is denoted $\mathsf{Potential}(\mathcal{DB})$.

Example 2. Consider again the database $\mathcal{DB}$ of Section 2.1. Using abbreviations with the obvious meanings, the following sets are examples of potential repairs for $\mathcal{DB}$:

$\mathcal{R}_1 = \langle \{\mathtt{emp}(Alice), \mathtt{emp}(Bob), \mathtt{sal}(Alice, 1000), \mathtt{sal}(Bob, 1000)\},\ \emptyset \rangle,$
$\mathcal{R}_2 = \langle \{\mathtt{emp}(Alice), \mathtt{emp}(Bob), \mathtt{dir}(Alice), \mathtt{dir}(Bob), \mathtt{sal}(Alice, 1000), \mathtt{sal}(Bob, 1000)\},\ \emptyset \rangle,$
$\mathcal{R}_3 = \langle \{\mathtt{emp}(Alice), \mathtt{emp}(Bob), \mathtt{dir}(Alice), \mathtt{sal}(Alice, 1100), \mathtt{sal}(Bob, 1000)\},\ \emptyset \rangle,$
$\mathcal{R}_4 = \langle \{\mathtt{emp}(Alice), \mathtt{emp}(Bob), \mathtt{dir}(Alice), \mathtt{sal}(Alice, x), \mathtt{sal}(Bob, 1000)\}, \{x > 1000\} \rangle,$
$\mathcal{R}_5 = \langle \{\mathtt{emp}(Alice), \mathtt{emp}(Bob), \mathtt{dir}(Alice), \mathtt{sal}(Alice, 1000), \mathtt{sal}(Bob, x)\}, \{x < 1000\} \rangle.$

These potential repairs represent different kinds of updates: $\mathcal{R}_1$ is obtained by *retracting* the fact that Alice is a director, $\mathcal{R}_2$ is obtained by *inserting* the fact that Bob is also a director, and $\mathcal{R}_3$ – $\mathcal{R}_5$ are obtained by *modifying* particular parts of tuples (namely, the salaries of Alice and Bob).

For selecting the best potential repairs we require that the repaired information would be as close as possible to the original one. Thus, given a distance function d on 2^{GAtoms}, the pre-repairs of a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ are the potential repairs of $\mathcal{DB}$ that are d-closest to $\mathcal{D}$:

Definition 7 (pre-repairs). A potential repair $\langle \mathcal{U}, \mathcal{C} \rangle$ of a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ is a *pre-repair* of $\mathcal{DB}$ with respect to a pseudo distance d on 2^{GAtoms}, if for every $\langle \mathcal{U}', \mathcal{C}' \rangle \in \mathsf{Potential}(\mathcal{DB})$ it holds that $d(\mathcal{U}, \mathcal{D}) \leq d(\mathcal{U}', \mathcal{D})$. The set of all pre-repairs of $\mathcal{DB}$ with respect to d is denoted by $\mathsf{Rep}_d(\mathcal{DB})$.

Pre-repairs describe plausible ways of repairing a given database. Yet, some of the pre-repairs provide a more faithful representation of a repair of the database. To see this, consider the following two potential repairs given in Example 2 for the database in Section 2.1:

$\mathcal{R}_3 = \langle \{\mathtt{emp}(Alice), \mathtt{emp}(Bob), \mathtt{dir}(Alice), \mathtt{sal}(Alice, 1100), \mathtt{sal}(Bob, 1000)\},\ \emptyset \rangle,$
$\mathcal{R}_4 = \langle \{\mathtt{emp}(Alice), \mathtt{emp}(Bob), \mathtt{dir}(Alice), \mathtt{sal}(Alice, x), \mathtt{sal}(Bob, 1000)\}, \{x > 1000\} \rangle.$

Suppose now that both $\mathcal{U}_3$ and $\mathcal{U}_4$ (the update components of $\mathcal{R}_3$ and $\mathcal{R}_4$, respectively) are equally distant from D (this is the case, e.g., when the underlying distance function is either d^1_Σ or d^2_Σ, considered in Section 3.4 below). Yet, it is obvious that $\mathcal{R}_4$ should be preferred over $\mathcal{R}_3$, since it does not guess the exact salary of Alice, but instead only states that this value should be bigger than 1000. To make this distinction among pre-repairs explicit, as well as to considerably reduce the amount of database repairs that should be taken into consideration, we make further preferences among pre-repairs according to Definition 9.

Definition 8 (models of an update). The *models* of an update $\mathfrak{U} = \langle \mathcal{U}[\overline{x}], \mathcal{C}[\overline{x}] \rangle$ is a set $mod(\mathfrak{U})$ of the least Herbrand models of the database instances $\mathcal{U}\ [\overline{c}/\overline{x}]$ for each solution $\overline{c}$ of $\mathcal{C}[\overline{x}]$.

Definition 9 (redundancy). An update $\mathfrak{U} = \langle \mathcal{U}, \mathcal{C} \rangle$ is *redundant* with respect to a set $\mathcal{S}$ of updates, if there is an update $\mathfrak{U}' = \langle \mathcal{U}', \mathcal{C}' \rangle$ in $\mathcal{S}$ such that $mod(\mathfrak{U}) \subseteq mod(\mathfrak{U}')$. A set $\mathcal{S}$ of updates is non-redundant if each of its elements is not redundant with respect to the rest of the set.

Definition 10 (repairs and repairing sets). A *repairing set* of a database $\mathcal{DB}$ with respect to a pseudo distance d is a subset $\Delta_d(\mathcal{DB})$ of $\mathsf{Rep}_d(\mathcal{DB})$ (the pre-repairs of $\mathcal{DB}$) such that:

- $\Delta_d(\mathcal{DB})$ is non-redundant, and
- for each $\mathcal{R} \in \mathsf{Rep}_d(\mathcal{DB}) \backslash \Delta_d(\mathcal{DB})$ there is an $\mathcal{R}' \in \Delta_d(\mathcal{DB})$ such that $mod(\mathcal{R}) \subseteq mod(\mathcal{R}')$.

An update is called a *repair* of $\mathcal{DB}$ if it is an element of some repairing set $\Delta_d(\mathcal{DB})$.

Example 3. Later in this paper (see Section 3.4), we shall consider the pseudo distances d^1_Σ and d^2_Σ on 2^{GAtoms}, according to which the potential repairs $\mathcal{R}_3 - \mathcal{R}_5$ in Example 2 are minimally distant from the database instance of $\mathcal{DB}$. It will follow, then, that these potential repairs are also pre-repairs of $\mathcal{DB}$. Yet, only $\mathcal{R}_4$ and $\mathcal{R}_5$ are repairs of $\mathcal{DB}$, while $\mathcal{R}_3$ is not a repair, since the latter is strictly redundant with respect to $\{\mathcal{R}_4\}$.

Given a database $\mathcal{DB}$ and a distance function d, its repairing set is not necessarily unique, since different sets may have interchangeable elements (that is, elements that may replace each other in the sets). For instance, a pre-repair of the form $\langle \{P(x)\}, \{x = c\} \rangle$ for some constant c, is interchangeable with $\langle \{P(c)\}, \emptyset \rangle$. Note, however, that interchangeable pre-repairs are equivalent, and so they have the same models. Moreover, modulo interchangeable pre-repairs, the repairing set $\Delta_d(\mathcal{DB})$ of $\mathcal{DB}$ is uniquely determined.

Proposition 1. *Let $\Delta^1_d(\mathcal{DB})$ and $\Delta^2_d(\mathcal{DB})$ be two repairing sets of the same database (with respect to the same distance). Then there is a one-to-one correspondence between repairs $\mathcal{R}_1 \in \Delta^1_d(\mathcal{DB})$ and repairs $\mathcal{R}_2 \in \Delta^2_d(\mathcal{DB})$ such that $\mathcal{R}_1$ and $\mathcal{R}_2$ have the same models.*

Proof. For a repair $\mathcal{R}_1 \in \Delta^1_d(\mathcal{DB})$, there is a repair $\mathcal{R}_2 \in \Delta^2_d(\mathcal{DB})$ such that $mod(\mathcal{R}_1) \subseteq mod(\mathcal{R}_2)$, and, in turn, there is a repair $\mathcal{R}_3 \in \Delta^1_d(\mathcal{DB})$ such that $mod(\mathcal{R}_2) \subseteq mod(\mathcal{R}_3)$. It follows immediately that $mod(\mathcal{R}_1) \subseteq mod(\mathcal{R}_2) \subseteq mod(\mathcal{R}_3)$, and hence $\mathcal{R}_1 = \mathcal{R}_3$ and $mod(\mathcal{R}_1) = mod(\mathcal{R}_2)$. Since a repairing set does not contain two repairs with the same models, this must be a one-to-one correspondence. □

This leads us to the following notions of query answering:

Definition 11 (query answering). A *query* $\mathcal{Q}(x_1, \ldots, x_n)$ is a first-order formula with free variables $x_1, \ldots, x_n$. Denote by $\mathcal{Q}[c_1/x_1, \ldots, c_n/x_n]$ the simultaneous substitution in $\mathcal{Q}$ of the variables x_i by the constants c_i $(i = 1, \ldots, n)$, respectively. Now, let d be a pseudo distance on 2^{GAtoms} and $\mathcal{Q}(x_1, \ldots, x_n)$ a query on $\mathcal{DB}$.

- A tuple $\langle c_1, \ldots, c_n \rangle$ is a *credulous answer* for $\mathcal{Q}$ if there exists an element $\mathcal{R} \in \Delta_d(\mathcal{DB})$ such that $\mathcal{R} \models \mathcal{Q}[c_1/x_1, \ldots, c_n/x_n]$ (i.e., each model of $\mathcal{R}$ satisfies $\mathcal{Q}[c_1/x_1, \ldots, c_n/x_n]$).
- A tuple $\langle c_1, \ldots, c_n \rangle$ is a *conservative answer* (alternatively, a *consistent query answer*) for $\mathcal{Q}$ if $\mathcal{R} \models \mathcal{Q}[c_1/x_1, \ldots, c_n/x_n]$ for every $\mathcal{R} \in \Delta_d(\mathcal{DB})$.

Example 4. Consider again Example 1. Here there are eight possible updates for $\mathcal{D}$, six of which are potential repairs: $\mathsf{Potential}(\mathcal{DB}) = \{\{\}, \{q\}, \{r\}, \{p, q\}, \{q, r\}, \{p, q, r\}\}$. Note that here we identify (potential and pre-) repairs with their update components, as in the propositional case the constraint components are always empty. For choosing the repairs (which are also the pre-repairs in this case), let's have the cardinality of the symmetric difference between sets as the distance function d at hand (that is, $d(A, B) = |A \backslash B| + |B \backslash A|$). This leaves us with two (optimal) repairs: $\Delta_d(\mathcal{DB}) = \mathsf{Rep}_d(\mathcal{DB}) = \{\{p, q, r\}, \{r\}\}$. It follows that, in this case, r is the only atomic formula that conservatively follows from $\mathcal{DB}$.

3 Distance Semantics for Database Repair

3.1 Distance functions

The choice of the distance function (and so the metric at hand) plays a crucial role in the repairing process. There are many possibilities to measure distances between the spoiled database instance and its potential repairs. Below, we recall two common definitions of such distances. An exposition of distance functions for simple expressions can be found in [40, Section 3.5].

Example 5. Let d be a distance function on a finite set $\mathcal{S}$. For $A, B \in 2^{\mathcal{S}}$, define:

- *The Hausdorff distance* [20]:

$$d(A,B) = \max\Big(\max_{a\in A}\min_{b\in B} d(a,b),\ \max_{b\in B}\min_{a\in A} d(a,b)\Big).$$

- *Eiter and Mannila's distance* [22]:

$$d(A,B) = \frac{1}{2}\Big(\sum_{a\in A}\min_{b\in B} d(a,b) + \sum_{b\in B}\min_{a\in A} d(a,b)\Big).$$

It is known that the Hausdorff distance is a distance function on $2^{\mathcal{S}}$ and Eiter–Mannila's distance is a pseudo distance on $2^{\mathcal{S}}$.

In what follows, we consider pseudo distances that are defined by matching functions (between the elements of the original database instance and the elements of a potential repair) and by aggregation functions that evaluate the quality of those matchings.

Definition 12 (aggregation functions). Let f be a total function that accepts a multiset of real numbers and returns a real number.

- f is called a numeric *aggregation function* if it is non-decreasing in the values of its argument,[4] $f(\{x_1,\ldots,x_n\}) = 0$ if $x_1 = \ldots x_n = 0$, and $\forall x \in \mathbb{R}\ f(\{x\}) = x$.
- A numeric aggregation function f such that $f(\{x_1,\ldots,x_n\}) = 0$ only if $x_1 = \ldots x_n = 0$ is called *strict*.

Aggregation functions are, e.g., a summation or the average of the distances, the maximum value among those distances (which yields a worst case analysis), a median value (for mean case analysis), and so forth. Such functions are common in data integration systems (see also Section 4 below). Note, also, that with the exception of the median value, all the functions mentioned above are strict.

Definition 13 (optimal matchings and d_f). Let $A, B \subseteq \mathsf{GAtoms}$, d a pseudo distance on GAtoms, and f a numeric aggregation function.

a) A *matching* m between A and B is a subset of $A\times B$ such that for every $(a_1,b_1),(a_2,b_2) \in m$, $a_1 = a_2$ iff $b_1 = b_2$.

[4] That is, the function value is non-decreasing when an element in the multiset is replaced by a larger element.

b) For a matching m between A and B, let $m(A) = \{b \mid \exists(a,b) \in m\}$ and $m^{-1}(B) = \{a \mid \exists(a,b) \in m\}$. Denote:

$$d_f(m,A,B) = f\Big(\Big\{d(a,b) \mid (a,b) \in m\Big\} \bigcup \Big\{d(a,B) \mid a \in A \setminus m^{-1}(B)\Big\} \bigcup \Big\{d(b,A) \mid b \in B \setminus m(A)\Big\}\Big),$$

where, for every element e and set $\mathcal{S}$, $d(e,\mathcal{S}) = \frac{1}{2}\max\{d(p,q) \mid p,q \in \mathsf{GAtoms}\}$. [5]

That is, d_f is obtained by applying f on the distances among matched elements and on the distances among non-matched elements and the other set.

c) A matching m between sets A and B is called $\{d,f\}$-*optimal* if for every matching m' between A and B, $d_f(m,A,B) \leq d_f(m',A,B)$.

d) Denote $d_f(A,B) = d_f(m,A,B)$, where m is a $\{d,f\}$-optimal matching between A and B.[6]

Example 6. Consider again the database of Example 1, together with the summation function f and the *drastic distance* d^u, defined as follows:

$$d^u(x,y) = 0 \text{ if } x = y \text{ and } d^u(x,y) = 1 \text{ otherwise.}$$

It is easy to verify that d^u is indeed a distance function on GAtoms. Now, as noted in Example 4, both $\mathcal{P} = \{p,q,r\}$ and $\mathcal{P}' = \{p,q\}$ are (among others) potential repairs of $\mathcal{D} = \{p,r\}$. An optimal matching between $\mathcal{D}$ and $\mathcal{P}$ relates p and r in $\mathcal{D}$ to the same atoms in $\mathcal{P}$ and leaves q unmatched. Thus, $d_f^u(\mathcal{D},\mathcal{P}) = 0 + 0 + \frac{1}{2} = \frac{1}{2}$. Similarly, $d_f^u(\mathcal{D},\mathcal{P}') = 1$, as an optimal matching between $\mathcal{D}$ and $\mathcal{P}'$ either leaves q and r unmatched (in which case the distance is $0 + \frac{1}{2} + \frac{1}{2}$) or connects both of them (and so the distance is equal to $d^u(p,p) + d^u(r,q) = 0 + 1$). The fact that $d_f^u(\mathcal{D},\mathcal{P}) < d_f^u(\mathcal{D},\mathcal{P}')$ is explained by the need to make only one modification for repairing $\mathcal{D}$ by $\mathcal{P}$, while $\mathcal{P}'$ requires two modifications in $\mathcal{D}$ (see also Example 7 below).

Proposition 2. *Let d be a pseudo distance on GAtoms and f a strict aggregation function. Then the function d_f introduced in Definition 13(d) is a pseudo distance on 2^{GAtoms}.*

Proof. Since d is symmetric, m is a $\{d,f\}$-optimal matching between A and B iff m^{-1} is a $\{d,f\}$-optimal matching between B and A. In this case, $d_f(m,A,B) = d_f(m^{-1},B,A)$, and so $d_f(A,B) = d_f(B,A)$. For identity preservation, note that the optimal matching between a set A and itself is the identity function I on A, and so $d_f(A,A) = d_f(I,A,A) = f(\{d(a,a) \mid a \in A\}) = 0$. In case that $A \neq B$, for every matching m between A and B we have that $d_f(m,A,B) > 0$ (this is so since at least one distance between matched elements is strictly positive, and as f is strict, its value in this case must be strictly positive as well). Thus, $d_f(A,B) > 0$ whenever $A \neq B$, and so $d_f(A,B) = 0$ iff $A = B$. □

3.2 Aggregation-based repairs

Proposition 2 induces a particular yet useful way to obtain database repairs. Distance functions on 2^{GAtoms} can be defined by a combination of aggregation functions and distance functions on GAtoms.

[5] Alternatively, one could define, for a nonempty set S, $d(e,S) = \frac{1}{2}\max\{d(e,s) \mid s \in S\}$. We shall use the former definition, which is independent of x and S, as a uniform handling of the unmatched elements will simplify the computations in what follows.

[6] As all the optimal matchings have the same d_f-value, $d_f(A,B)$ is well-defined.

Definition 14. The *pre-repairs* of a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ with respect to a pseudo distance d on GAtoms and a numeric aggregation function f, are the elements of the set

$$\mathsf{Rep}_{d,f}(\mathcal{DB}) = \mathsf{Rep}_{d_f}(\mathcal{DB}) =$$
$$\big\{\mathcal{R} = \langle \mathcal{U}, \mathcal{C}\rangle \in \mathsf{Potential}(\mathcal{DB}) \mid \forall \langle \mathcal{U}', \mathcal{C}'\rangle \in \mathsf{Potential}(\mathcal{DB})\ d_f(\mathcal{U}, \mathcal{D}) \leq d_f(\mathcal{U}', \mathcal{D})\big\}.$$

Accordingly, the *repairing set* of $\mathcal{DB}$ with respect to d and f is the maximal subset $\Delta_{d,f}(\mathcal{DB})$ of $\mathsf{Rep}_{d,f}(\mathcal{DB})$ that is non-redundant with respect to $\mathsf{Rep}_{d,f}(\mathcal{DB})$; that is: $\Delta_{d,f}(\mathcal{DB}) = \Delta_{d_f}(\mathcal{DB})$.

Clearly, Definition 14 is a particular case of Definition 7, since d_f, obtained from d and f by Definition 13(d), is a pseudo distance (Proposition 2).

The advantage of the aggregation-based presentation of repairs is that it allows to express, in a natural way, distances between sets in terms of distances between the elements of those sets. As we shall see, this allows to encode within the distance function many practical considerations in the context of database systems (e.g., whether the matched elements are parts of a primary key, etc.).

Example 7. Consider again the database $\mathcal{DB} = (\{p, r\}, \{p \to q\})$ of Example 1 together with the drastic distance ($d = d^u$; see Example 6) and the summation aggregation ($f = \Sigma$). The (updates of the) six potential repairs of $\mathcal{DB}$ and their distances from $\mathcal{D} = \{p, r\}$ are given in the table below (the optimal matchings are computed just as illustrated in Example 6).

No.	Potential Repair	$d^u_\Sigma(\cdot, \mathcal{D})$	Actions
1	$\{p, q, r\}$	$\frac{1}{2}$	insert q
2	$\{p, q\}$	1	insert q, delete r
3	$\{q, r\}$	1	insert q, delete p
4	$\{q\}$	$1\frac{1}{2}$	insert q, delete p and r
5	$\{r\}$	$\frac{1}{2}$	delete p
6	$\{\}$	1	delete p and r

It follows, then, that the repairs in this case are $\mathcal{R}_1 = \langle\{p, q, r\}, \emptyset\rangle$ and $\mathcal{R}_5 = \langle\{r\}, \emptyset\rangle$. Among the potential repairs, these repairs require a minimal amount of modifications in $\mathcal{D}$. As Proposition 4 below shows, this is not a coincidence. Note also that d^u_Σ is in fact the symmetric distance between sets, so there is no wonder that $\mathcal{R}_1$ and $\mathcal{R}_5$ are exactly the same repairs as those obtained in Example 4. Again, we have that, e.g., r conservatively (and so credulously) follows from $\mathcal{DB}$, while p as well as q credulously (but not conservatively) follow from $\mathcal{DB}$.

The next proposition shows that, as expected, there is nothing to repair in consistent databases.

Proposition 3. If $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ is a consistent database and f is a strict aggregation function, then for every pseudo distance d, $\Delta_{d,f}(\mathcal{DB}) = \{\langle \mathcal{D}, \emptyset\rangle\}$.

Proof. If $\mathcal{DB}$ is consistent, then $\langle \mathcal{D}, \emptyset\rangle$ is obviously a potential repair. Moreover, for every distance function d and aggregation function f, the identity function I on $\mathcal{D}$ is a $\{d, f\}$-optimal matching, as $d_f(I, \mathcal{D}, \mathcal{D}) = 0$ (by the identity preservation of pseudo distance functions and the second condition in Definition 12), and so $d_f(\mathcal{D}, \mathcal{D}) = 0$. For every other set $A \in 2^{\mathsf{GAtoms}}$, it clearly holds that, for each matching m between $\mathcal{D}$ and A, $d_f(m, \mathcal{D}, A) > 0$, and hence, for each pre-repair $\langle \mathcal{U}, \mathcal{C}\rangle$ not equivalent to $\langle \mathcal{D}, \emptyset\rangle$, $d(\mathcal{D}, \mathcal{U}) > 0$. It follows that $\{\langle \mathcal{D}, \emptyset\rangle\}$ is a repairing set. □

3.3 Repairs independent of the domain of discourse

The distance-aggregation function d^u_Σ, obtained by a summation of the drastic distances d^u between matched elements, is frequently used for repairing databases (see also Example 7). These functions are 'blind' to the domain of discourse at hand, in the sense that both d^u and d^u_Σ refer, respectively, to the elements of GAtoms and of 2^{GAtoms} in a uniform way. Indeed, as it is shown below (Proposition 4), $d^u_\Sigma(A, B)$ is determined only by the cardinality of the symmetric distance between A and B rather by the particular meaning or the properties of the elements in those sets. We call such distances (and the repairs that are obtained by them) *domain independent*. In this section we consider such distances and repairs.

First, we show that the metric that is obtained by d^u_Σ corresponds to the Hamming distance between sets of formulae (also known as the symmetric distance, or the Dalal distance [16]).

Proposition 4. *Let $|\mathcal{S}|$ be the size of $\mathcal{S}$. Then $d^u_\Sigma(A, B) = \frac{1}{2}(|A \setminus B| + |B \setminus A|)$.*

Proof. Note first that, as already noted, the summation Σ is an aggregation function and d^u is a distance function on GAtoms. Now, let $x \in (A \cup B) \setminus (A \cap B)$. If x is linked to another element y, then as $d^u(x, y) = 1$, the 'contribution' of x to $d^u_\Sigma(A, B)$ is $\frac{1}{2}$ (y contributes the other half). Otherwise, x is not linked to any element of the other set $S \in \{A, B\}$, and so $d(x, S) = \frac{1}{2}$. In any case, every element outside the intersection of A and B contributes $\frac{1}{2}$ to $d^u_\Sigma(A, B)$. Also, for any matching m that maps the elements in $A \cap B$ to themselves, $d^u_\Sigma(m, A, B) = d^u_\Sigma(m, A \setminus B, B \setminus A)$, since $d^u(x, x) = 0$. It follows, then, that a matching m of A and B that is the identity on $A \cap B$, is optimal in this case, and $d_f(m, A, B) = \frac{1}{2}|\{x \mid x \in (A \cup B) \setminus A \cap B\}|$. For such an m, $d_f(A, B) = d_f(m, A, B)$, and so we are done. □

The distance function d^u_Σ corresponds to the following cardinality-based repair, considered e.g. in [2, 4, 6, 37]

Definition 15. A *pairwise*[7] *repair* of $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ is a pair $(\mathsf{Insert}, \mathsf{Retract})$ of two sets of ground atomic facts, such that: 1. $\mathsf{Insert} \cap \mathcal{D} = \emptyset$, 2. $\mathsf{Retract} \subseteq \mathcal{D}$, 3. $(\mathcal{D} \cup \mathsf{Insert} \setminus \mathsf{Retract}, \mathcal{IC})$ is a consistent database, 4. $(\mathsf{Insert}, \mathsf{Retract})$ is minimal in its cardinality: there is no pair $(\mathsf{Insert}', \mathsf{Retract}')$ that satisfies conditions 1–3 and for which it holds that $|\mathsf{Insert}' \cup \mathsf{Retract}'| < |\mathsf{Insert} \cup \mathsf{Retract}|$.

Note 3. A different way of repairing databases is obtained by exchanging the cardinality-based requirement in item 4 by a set inclusion criterion. This is the basic idea behind the method introduced in [1], which inspires many other works on (domain independent) database repair and consistent query answering (see, e.g., [2, 5, 9, 10, 27, 26, 33]). Clearly, every pairwise repair that is obtained by Definition 15 is also a repair according to the set inclusion approach, but the converse is not necessarily true. Comparative studies of the two repair methods appear in [6, 37].

Proposition 5. Consider a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ together with the drastic distance function d^u and the summation aggregation function Σ. Then $(\mathsf{Insert}, \mathsf{Retract})$ is a pairwise repair of $\mathcal{DB}$ iff there is a pre-repair $\mathcal{R} = \langle \mathcal{U}, \emptyset \rangle \in \mathsf{Rep}_{d^u_\Sigma}(\mathcal{DB})$, such that $\mathsf{Insert} = \mathcal{U} \setminus \mathcal{D}$ and $\mathsf{Retract} = \mathcal{D} \setminus \mathcal{U}$.

Proof. Given a pairwise repair $(\mathsf{Insert}, \mathsf{Retract})$ of $(\mathcal{D}, \mathcal{IC})$, let $\mathcal{R} = \langle \mathcal{U}, \emptyset \rangle$, where $\mathcal{U} = (\mathcal{D} \cup \mathsf{Insert}) \setminus \mathsf{Retract}$. By condition (3) of Definition 15 it is clear that $\mathcal{R}$ is a potential repair, and by condition (4) of the same definition it follows that the Hamming distance between $\mathcal{U}$ and $\mathcal{D}$ is minimal among the distances between the other updates of the potential repairs

[7] This adjective is added to distinguish this kind of repairs from repairs in our sense.

of $\mathcal{DB}$ and $\mathcal{D}$ (otherwise, if there is a potential repair $\mathcal{R}' = \langle \mathcal{U}', \mathcal{C}' \rangle$ with a smaller Hamming distance to $\mathcal{D}$, the pair $(\mathsf{Insert}', \mathsf{Retract}') = (\mathcal{U}' \setminus \mathcal{D}, \mathcal{D} \setminus \mathcal{U}')$ satisfies conditions (1)–(3) of Definition 15 and $|\mathsf{Insert}' \cup \mathsf{Retract}'| < |\mathsf{Insert} \cup \mathsf{Retract}|$, which is a contradiction to the assumption that $(\mathsf{Insert}, \mathsf{Retract})$ is a pairwise repair of $\mathcal{DB}$). By Proposition 4, $d^u_\Sigma(\mathcal{U}, \mathcal{D})$ is minimal in $\{d^u_\Sigma(\mathcal{U}', \mathcal{D}) \mid \langle \mathcal{U}', \mathcal{C}' \rangle \text{ is a potential repair of } \mathcal{DB}\}$, thus $\mathcal{R}$ is a pre-repair of $\mathcal{DB}$.

Conversely, for a pre-repair $\mathcal{R} = \langle \mathcal{U}, \emptyset \rangle \in \mathsf{Rep}_{d^u_\Sigma}(\mathcal{DB})$ such that $\mathcal{U} \subseteq \mathsf{Atoms}(\mathcal{DB})$, let $(\mathsf{Insert}, \mathsf{Retract}) = (\mathcal{U} \setminus \mathcal{D}, \mathcal{D} \setminus \mathcal{U})$. This pair clearly satisfy conditions (1) and (2) of Definition 15. Condition (3) is met as well because in our case, $(\mathcal{D} \cup \mathsf{Insert} \setminus \mathsf{Retract}, \mathcal{IC}) = (\mathcal{U}, \mathcal{IC})$, and this is a consistent database since $\mathcal{R}$ is a potential repair of $\mathcal{DB}$ (and so it satisfies $\mathcal{IC}$). Finally, similar considerations as above show that the pair $(\mathsf{Insert}, \mathsf{Retract})$ is minimal in the sense of condition (4). Thus, $(\mathsf{Insert}, \mathsf{Retract})$ is a pairwise repair of $\mathcal{DB}$. □

Note 4. As Proposition 5 shows, pairwise repairs correspond to *pre*-repairs rather than to repairs. The reason for this is that in our framework repairs are represented by arbitrary (i.e, not necessarily ground) atomic formulas. This induces a compact way of representing database modifications, in which several pre-repairs can be captured simultaneously, and so particular pairwise repairs may be redundant. To see this, consider the following database:

$$\mathcal{DB} = \Big(\big\{ \mathtt{mother}(Jane) \big\},\ \big\{ \forall x \big(\mathtt{mother}(x) \rightarrow \exists n (\mathtt{num\_childs}(x, n) \wedge 1 \leq n \leq 20) \big) \big\} \Big).$$

In this case, for every constant $1 \leq i \leq 20$, we have that $(\{\mathtt{num\_childs}(Jane, i)\}, \emptyset)$ is a pairwise repair of $\mathcal{DB}$, and so, by Proposition 5, it is also a pre-repair of $\mathcal{DB}$ (with respect to d^u_Σ). However, each one of those 20 pre-repairs is redundant with respect to (the repair) $\langle \{\mathtt{mother}(Jane), \mathtt{num\_childs}(Jane, x)\}, \{x \geq 1,\ x \leq 20\} \rangle$.

Note 5. From Proposition 5 and the fact that the unique pairwise repair of a consistent database is $(\emptyset, \emptyset)$, it follows that for a consistent database $\mathcal{DB}$, $\Delta_{d^u, \Sigma}(\mathcal{DB}) = \{\langle \mathcal{D}, \emptyset \rangle\}$ (which is a particular case of Proposition 3).

It is also interesting to check the distance-based functions of Example 5 when the domain independent distance d^u is taken as the basic distance function. In this case the Hausdorff distance is reduced to 0 if $A = B$ and 1 otherwise. While this is still a distance function, it is clearly useless for making subtle preferences among potential repairs. The Eiter–Mannila distance, on the other hand, is more appropriate in this case and, as in Proposition 5, it is related to pairwise repairing. Indeed, for d^u, the Eiter–Mannila distance between the original database $\mathcal{D}$ and its repair $\langle \mathcal{D} \cup \mathsf{Insert} \setminus \mathsf{Retract}, \mathcal{C} \rangle$ is $\frac{1}{2}(|\mathsf{Insert}| + |\mathsf{Retract}|)$. In this case we get the Ramon–Bruynooghe matching-based distance [41], which is a distance function (and not only a pseudo distance, cf. Example 5).

3.4 Repairs dependent on the domain of discourse

Consider again the potential repairs considered in Example 2 (regarding the database of Section 2.1). If we repair the database using e.g. d^u_Σ as the underlying distance function, we get that $\mathcal{R}_1$ and $\mathcal{R}_2$ are strictly preferred over the other three potential repairs. Indeed, $d^u_\Sigma(\mathcal{U}_1, \mathcal{D}) = d^u_\Sigma(\mathcal{U}_2, \mathcal{D}) = \frac{1}{2}$, since the cost of the optimal matching between the original database instance $\mathcal{D}$ and the repaired database that is obtained by $\mathcal{R}_1$ (respectively, by $\mathcal{R}_2$) is the cost of the retracted (respectively, inserted) fact $\mathtt{dir}(Alice)$ (respectively, $\mathtt{dir}(Bob)$) that cannot be matched to any element in the original database. On the other hand, the optimal matching between the original database and the repaired database that is obtained by $\mathcal{R}_5$ for instance, links each one of $\mathtt{emp}(Alice)$, $\mathtt{emp}(Bob)$, $\mathtt{sal}(Alice, 1000)$ and $\mathtt{dir}(Alice)$ to the same facts in the repaired database, and relates $\mathtt{sal}(Bob, 1000)$ to $\mathtt{sal}(Bob, x)$ (for some $x < 1000$). The resulting distance is therefore $d^u_\Sigma(\mathcal{U}_5, \mathcal{D}) = 0 + 0 + 0 + 0 + 1 = 1$, and

similarly $d^u_\Sigma(\mathcal{U}_3, \mathcal{D}) = d^u_\Sigma(\mathcal{U}_4, \mathcal{D}) = 1$. However, in this case, the potential repairs $\mathcal{R}_3$, $\mathcal{R}_4$, and $\mathcal{R}_5$, that require salary changes, seem more plausible than the potential repairs $\mathcal{R}_1$ and $\mathcal{R}_2$ that require removal or insertion of database facts, as it is more realistic here to assume that the problem is due to a typographic error in the salary information. Moreover, $\mathcal{R}_1$ and $\mathcal{R}_2$ are more drastic then the other potential repairs, as they either cause information loss (there is no data about the directors of the company, in the case of $\mathcal{R}_1$) or reduce the information reliability (due to the introduction of unfaithful facts, in the case of $\mathcal{R}_2$). It is clear, then, that simple cardinality considerations are not useful here, and more delicate considerations are required, in which each one of $\mathcal{R}_3$, $\mathcal{R}_4$, and $\mathcal{R}_5$ will be preferred over $\mathcal{R}_1$ and $\mathcal{R}_2$.

This simple example demonstrates a common phenomenon in many inconsistent first-order databases: A specific tuple may contain both correct and erroneous components. In such cases, deleting or inserting entire tuples would not properly solve the problem. Indeed, the need to rectify an error within a tuple without deleting the whole tuple has been acknowledged in [5] (see Example 6.2 of that paper), and is also the main motivation behind the work in [8] on fixing (numerical) attributes and in [46, 47] on database repairing by updates; See also Section 5 below.

A more subtle preference criterion is obtained by the distance function, defined in [38]:

$$d^1\big(P(t_1,\ldots,t_m), Q(s_1,\ldots,s_n)\big) = \begin{cases} 1 & \text{if } P \neq Q, \\ \dfrac{1}{2m}\displaystyle\sum_{i=1}^{m} d^u(t_i, s_i) & \text{otherwise.} \end{cases}$$

Here, for different predicate symbols the distance d^1 is maximal. However, when the predicate symbols are the same, the distance linearly increases with the number of arguments that have different values, and is at most $\frac{1}{2}$. The intuition behind this is that longer tuples are more error-prone and that multiple errors in the same tuple are less likely.

Proposition 6. *d^1 is a distance function (in the sense of Definition 3), which is bounded by 1.*

Proof. Easily verified (see also [38, Theorem 5]). □

According to d^1 together with a summation as the distance aggregation function, the distance between the database instance $\mathcal{D}$ of Example 2 and (the update component of) $\mathcal{R}_1$ is still $\frac{1}{2}$, and so is the distance between $\mathcal{D}$ and (the update component of) $\mathcal{R}_2$. On the other hand, the distance between $\mathcal{D}$ and $\mathcal{U}_5$ (the update component of $\mathcal{R}_5$) is the same as the distance between $\texttt{salary}(Bob, 1000)$ and $\texttt{salary}(Bob, x)$, which is $\frac{1}{4}(0+1) = \frac{1}{4}$. Similarly, $d^1_\Sigma(\mathcal{U}_3, \mathcal{D}) = d^1_\Sigma(\mathcal{U}_4, \mathcal{D}) = \frac{1}{4}$. It follows, then, that now the potential repairs that modify information within tuples are preferred over potential repairs that remove or insert complete tuples, as intuitively expected.

Nienhuys-Cheng's distance d^1 can be further refined to reveal other considerations. For instance, under the assumption that primary keys are less error-prone, one may consider the following variation of d^1:

Definition 16. Below we denote primary key values by underscores, and assume, without loss of generality, that they precede the non-key values. Define:

$$d^2\big(P(\underline{t_1},\ldots,\underline{t_k},t_{k+1},\ldots,t_m), Q(\underline{s_1},\ldots,\underline{s_l},t_{l+1},\ldots,t_n)\big) =$$
$$\begin{cases} 1 & \text{if } P \neq Q \text{ or } \exists 1 \leq i \leq k \text{ s.t. } \underline{t_i} \neq \underline{s_i}, \\ \dfrac{1}{2m}\displaystyle\sum_{i=k+1}^{m} d^u(t_i, s_i) & \text{otherwise.} \end{cases}$$

Example 8. Consider again the database $\mathcal{DB}$ of Section 2.1. There are five options regarding the fact $\texttt{salary}(Alice, 1000)$: Keeping it unchanged, changing the first argument (employee-name), changing the second argument (salary), changing the whole tuple, or deleting it altogether. Assuming that employee-name is the primary key for the salary relation, we have that according to d^2, the costs of these options are 0, 1, $\frac{1}{4}$, 1, and $\frac{1}{2}$, respectively. Note, also, that in this case, according to the aggregation-based repair with d^2 and Σ, the two repairs of the database are the following:

$$\langle\{\texttt{emp}(Alice),\ \texttt{emp}(Bob),\ \texttt{dir}(Alice),\ \texttt{sal}(Alice, v_1),\ \texttt{sal}(Bob, 1000)\}, \{v_1 > 1000\}\rangle,$$

$$\langle\{\texttt{emp}(Alice),\ \texttt{emp}(Bob),\ \texttt{dir}(Alice),\ \texttt{sal}(Alice, 1000),\ \texttt{sal}(Bob, v_2)\}, \{v_2 < 1000\}\rangle.$$

That is, consistency restoration is obtained here by salary corrections.

3.5 Linking instead of matching

The notion of (optimal) matching between the elements of a database instance and the elements of its repair may be weakened. Instead of relating each database fact with at most one atomic formula of a repair and vice versa, it is possible to associate a database fact with *several* atoms of a repair. This is called *linking*. Optimal linking and the induced distance between sets are defined just as in Definition 13.

Example 9. Consider a database instance $\mathcal{D} = \{\texttt{teaches}(John, DB)\}$ and integrity constraints that no-one teaches *DB* (since, e.g., this course is cancelled), and that a lecturer must give at least two courses. Here, a repair with respect to d^1_Σ could be

$$\mathcal{R} = \langle\mathcal{U}, \mathcal{C}\rangle = \langle\{\texttt{teaches}(John, x_1), \texttt{teaches}(John, x_2)\}, \{x_1 \neq x_2, x_1 \neq DB, x_2 \neq DB\}\rangle.$$

Each one of the two optimal matchings in this case relates the database fact to one of the two elements in $\mathcal{U}$, leaving the other one unmatched. In the notations of the previous section, then, $d^1_\Sigma(\mathcal{D},\mathcal{U}) = \frac{1}{2} + \frac{1}{4}$. If linking is used instead of matching, there is only one optimal linking between $\mathcal{D}$ and $\mathcal{U}$, which associates the two new facts in $\mathcal{U}$ with the old one in $\mathcal{D}$, hence in this case $d^1_\Sigma(\mathcal{D},\mathcal{U}) = \frac{1}{4} + \frac{1}{4}$.

3.6 Complexity

Computing all the repairs of a given database is not tractable, as even for propositional databases the number of repairs of a database could be exponential in the database's size. Indeed, the database $(\{p_1, \ldots, p_n\}, \{p_i \rightarrow q_i\}_{i=1}^n)$ has 2^n repairs when $d = d^u$ and $f = \Sigma$. These repairs correspond to all the combinations of inserting q_i or removing p_i, for $i = 1, \ldots, n$. In an attempt to overcome this problem, most of the existing algorithms for query answering do not compute the repairs themselves, but make inferences using rewriting techniques [1], logic programming paradigms [2, 21, 25–27], (hyper-)graph computations [14, 15], and proof theoretic methods, such as analytic tableaux [10]. In the general case, however, these techniques are not tractable. For instance, the approach in [2, 26] of specifying database repairs as stable models of disjunctive logic programs is Π^P_2-complete (see [17]). This is also the case for the query answering computations with (signed) quantified Boolean formulas, considered in [5]. Tractability of query evaluation for inconsistent databases is usually reached only for restricted syntactical forms of the integrity constraints. For instance, the rewriting process in [1], which is a tractable way of evaluating queries with respect to the set-inclusion semantics (see Note 3), is limited to binary universal constraints.

The next proposition shows that intractability retains in our case as well:

Proposition 7. [37] *Let $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ be a database with denial constraints,*[8] *and let $\mathcal{Q}$ be a query that is a conjunction of ground literals. Then conservative query answering for $\mathcal{Q}$ with respect to the d^u_Σ-distance semantics is $\mathsf{P}^{\mathsf{NP}(\log(n))}$-complete.*[9]

Proof. By [37, Theorem 4] and the fact that d^u_Σ-semantics corresponds to pairwise repairs (Proposition 5). □

Another interesting property of the d^u_Σ-distance semantics explored in [37] is that for databases with denial constraints, conservative query answering and credulous query answering (Definition 11) are polynomially reducible.[10]

As in the case of inclusion-based semantics, also in the cardinality-based semantics obtained by the d^u_Σ-distance, conservative query answering may be undecidable unless restrictions are imposed on the syntactical form of the integrity constraints. This can be shown in a similar way as that of [11] (in which decidability is considered with respect to set inclusion semantics), where, intuitively, undecidability stems from the possible presence of cycles among inclusion dependencies of the form $\forall\overline{x}\exists\overline{y}(P(\overline{x}) \rightarrow Q(\overline{x}', \overline{y}))$ for $\overline{x}' \subseteq \overline{x}$, and from the possibility of using arbitrary elements from the (infinite) set $\mathfrak{C}$, as constants in the repaired database.

Undecidability results for domain dependent repairs are also easily obtained for sufficiently expressible integrity constraints. See, for instance, [8, Theorem 1] for one example in the context of database repair by fixing numerical attributes (see also Section 5 below).

We note, finally, that in general, the distance functions themselves do not add extra computational complexity to the problem. This is demonstrated, for instance, by the following results:

Proposition 8. [41] *Computing $d^u_\Sigma(A, B)$ is polynomial in the size of A and B.*

Proposition 9. *Computing $d^1_\Sigma(A, B)$ and $d^2_\Sigma(A, B)$ is polynomial in the sizes of A, B, and the maximal arity of the predicates in A and B.*

Proof. Follows from the fact that if the time to compute $d(x, y)$ is bounded by T for every $x \in A, y \in B$, then the time to compute $d_\Sigma(A, B)$ is bounded by a polynom in $|A|$, $|B|$, and T (see [41]). □

The main computational difficulty of database repairs remains, therefore, the large amount of potential repairs at hand. Extensive surveys on the computational complexity of existing approaches to database repair and consistent query answering appear in [6, 11, 13, 14, 37] (see also [47] for complexity results regarding update-based repairing).

4 Integration of Constraint Data-Sources

Integration of autonomous data-sources under global integrity constraints (see [30]) is closely related to database repair. The main differences between the two problems is that, in contrast to database instances, data-sources may contain negative facts and not only positive ones. Also, the closed world assumption is no longer assumed. In this section we show how our framework may be used for defining operators for the merging problem as well.

[8] I.e., $\mathcal{IC}$ consists of closed formulae of the form $\forall\overline{x}_1 \ldots \overline{x}_n \neg\big(R_1(\overline{x}_1) \wedge \ldots \wedge R_n(\overline{x}_n) \wedge \phi(\overline{x}_1 \ldots \overline{x}_n)\big)$, where ϕ is a Boolean expression consisting of atomic formulas and built-in predicates.

[9] That is, the decision problem can be solved by a polynomial-time algorithm that makes $O(\log(n))$ calls to an NP-oracle, where n is the size of $\mathcal{D}$.

[10] This is not the case when repair by set-inclusion is involved, as in that case conservative answering with denial constraints is NP-complete, while credulous answering is in P; see [15].

Example 10. [30] Four flat co-owners discuss the construction of a swimming pool (s), a tennis-court (t) and a private car-park (p). Building two or more items will increase the rent (r), otherwise the rent will not be changed.

The owners' opinions are represented by the following four data-sources: $\mathcal{D}_1 = \mathcal{D}_2 = \{s, t, p\}$, $\mathcal{D}_3 = \{\neg s, \neg t, \neg p, \neg r\}$, $\mathcal{D}_4 = \{t, p, \neg r\}$. The impact on the rent may be represented by the constraint $\mathcal{IC} = \{r \leftrightarrow \big((s \wedge t) \vee (s \wedge p) \vee (t \wedge p)\big)\}$. Here, $q \in \mathcal{D}_i$ denotes that owner i supports q, and $\neg q \in \mathcal{D}_i$ denotes that owner i is against q. If $q, \neg q \notin \mathcal{D}_i$, i does not have an opinion about q. Note that although the opinion of owner 4 violates the integrity constraint (while the solution must preserve the constraint), it is still taken into account.

In situations such as that of Example 10 it is often required to find a solution that will satisfy the global integrity constraints and will be as close as possible to each data source. This implies that, under the following observations, our framework is adequate for the merging problem as well. Denote by $\mathfrak{D}$ the set of the sources to be merged. Then:

- Instead of database instances, which are sets of atomic facts, data sources are sets of *literals* (that is, atomic formulas or their negation). So, instead of Atoms we refer now to $\mathsf{LIT} = \{P(\bar{t}) \mid P(\bar{t}) \in \mathsf{Atoms}\} \cup \{\neg P(\bar{t}) \mid P(\bar{t}) \in \mathsf{Atoms}\}$. As before, an update is a pair $\langle \mathcal{U}, \mathcal{C} \rangle$, where $\mathcal{U}$ is a consistent set of elements in LIT (i.e., $\mathcal{U}$ is a set without complementary literals), and $\mathcal{C}$ is a set of constraints. The set $\mathsf{Merge}(\mathfrak{D}, \mathcal{IC})$ of the potential merging of $\mathfrak{D}$ under $\mathcal{IC}$ consists of the updates that satisfy all the integrity constraints in $\mathcal{IC}$.
- A *merging* of data-sources $\mathfrak{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ with respect to the integrity constraints $\mathcal{IC}$ is a straightforward generalization of the notion of database repair (cf. Definitions 13 and 14):
  - A *merging context* is a triple $\mathfrak{M} = \langle d, f, g \rangle$, where d is a pseudo distance function, and f, g are aggregation functions (referring, respectively, to the distances inside a source and among the sources).
  - For a merging context $\mathfrak{M} = \langle d, f, g \rangle$, a set $\mathfrak{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ of data-sources, and a potential merging $\mathcal{M} \in \mathsf{Merge}(\mathfrak{D}, \mathcal{IC})$, let
  $$d_{g,f}(\mathcal{M}, \mathfrak{D}) = g\big(\{d_f(\mathcal{M}, \mathcal{D}_1), \ldots, d_f(\mathcal{M}, \mathcal{D}_n)\}\big).$$
  - The *pre-mergings* of the data-sources in $\mathfrak{D}$ under the integrity constraints in $\mathcal{IC}$, and with respect to the merging context $\mathfrak{M} = \langle d, f, g \rangle$, are the elements of the set
  $$\big\{\mathcal{M} \in \mathsf{Merge}(\mathfrak{D}, \mathcal{IC}) \mid \forall \mathcal{M}' \in \mathsf{Merge}(\mathfrak{D}, \mathcal{IC})\ d_{g,f}(\mathcal{M}, \mathfrak{D}) \leq d_{g,f}(\mathcal{M}', \mathfrak{D})\big\}.$$
  - The *merging* of the data-sources in $\mathfrak{D}$ under the integrity constraints in $\mathcal{IC}$, and with respect to the merging context $\mathfrak{M} = \langle d, f, g \rangle$, are the maximal non-redundant pre-mergings of $\mathfrak{D}$. This set is denoted by $\Delta_{\mathfrak{M}}(\mathcal{DB})$.

Example 11. Consider again Example 10 and two merging contexts: $\mathfrak{M}_1 = \langle d^u, \Sigma, \Sigma \rangle$, $\mathfrak{M}_2 = \langle d^u, \Sigma, \max \rangle$. According to $\mathfrak{M}_1$ the summation of the distances to the source is minimized, and in $\mathfrak{M}_2$ minimization of maximal distances is used for choosing optimal solutions. The potential mergings in this case are listed in the table below.

No.	Potential merge	$d^u_\Sigma(\cdot, \mathcal{D}_1)$	$d^u_\Sigma(\cdot, \mathcal{D}_2)$	$d^u_\Sigma(\cdot, \mathcal{D}_3)$	$d^u_\Sigma(\cdot, \mathcal{D}_4)$	$d^u_{\Sigma,\Sigma}(\cdot, \mathfrak{D})$	$d^u_{\max,\Sigma}(\cdot, \mathfrak{D})$
$\mathcal{M}_1$	$\{s, t, p, r\}$	$\frac{1}{2}$	$\frac{1}{2}$	4	$1\frac{1}{2}$	$6\frac{1}{2}$	4
$\mathcal{M}_2$	$\{s, t, \neg p, r\}$	$1\frac{1}{2}$	$1\frac{1}{2}$	3	$2\frac{1}{2}$	$8\frac{1}{2}$	3
$\mathcal{M}_3$	$\{s, \neg t, p, r\}$	$1\frac{1}{2}$	$1\frac{1}{2}$	3	$2\frac{1}{2}$	$8\frac{1}{2}$	3
$\mathcal{M}_4$	$\{s, \neg t, \neg p, \neg r\}$	$2\frac{1}{2}$	$2\frac{1}{2}$	1	$2\frac{1}{2}$	$8\frac{1}{2}$	$2\frac{1}{2}$
$\mathcal{M}_5$	$\{\neg s, t, p, r\}$	$1\frac{1}{2}$	$1\frac{1}{2}$	3	$1\frac{1}{2}$	$7\frac{1}{2}$	3
$\mathcal{M}_6$	$\{\neg s, t, \neg p, \neg r\}$	$2\frac{1}{2}$	$2\frac{1}{2}$	1	$1\frac{1}{2}$	$7\frac{1}{2}$	$2\frac{1}{2}$
$\mathcal{M}_7$	$\{\neg s, \neg t, p, \neg r\}$	$2\frac{1}{2}$	$2\frac{1}{2}$	1	$1\frac{1}{2}$	$7\frac{1}{2}$	$2\frac{1}{2}$
$\mathcal{M}_8$	$\{\neg s, \neg t, \neg p, \neg r\}$	$3\frac{1}{2}$	$3\frac{1}{2}$	0	$2\frac{1}{2}$	$9\frac{1}{2}$	$3\frac{1}{2}$

The optimal merging in each context is determined by the minimal values in the two rightmost columns. According to $\mathfrak{M}_1$, $\mathcal{M}_1$ is the best potential merging, and so the owners decide to build all the three facilities. As a result, the rent increases. According to $\mathfrak{M}_2$, however, $\mathcal{M}_4$, $\mathcal{M}_6$ and $\mathcal{M}_7$ are the optimal mergings, which implies that only one out of the three facilities will be built, and so the rent will remain the same.[11] Thus, e.g., r is a consistent query answer with respect to $\mathfrak{M}_1$ while $\neg r$ is a consistent answer with respect to $\mathfrak{M}_2$.

The last example demonstrates the application of merging strategies among equally important sources. However, there are situations in which certain sources are preferred over other sources (for instance, because of differences in reliability of the sources). Our framework supports such cases as well by a proper choice of the components of the merging context. We demonstrate this in the next example.

Example 12. Consider the distributed system described in Examples 10 and 11, but this time in the context of speculations on the stock exchange. An investor (represented by the mediator system) consults four financial experts about their opinion regarding four different shares, denoted t, p, s, r. The opinion of expert i is represented by $\mathcal{D}_i$ (see Example 10). For instance, in our case expert 4 suggests to buy shares t and p, doesn't recommend to buy share r, and doesn't have a particular opinion about share s. The integrity constraint in our case may be interpreted as the investor's own policy of buying shares. (For instance, the integrity constraint in Example 10 may be intuitively understood by the risk of buying share r that should be 'balanced' by purchasing at least two out of the three shares t, p, s). Clearly, the experts could have different reputations, and this may affect the investor's decision, which is embodied in the distance function. This is expressible by a weighted average distance function, in which the distance to each source is multiplied by a different certainty factor (smaller factors are attached to more reliable sources). This yields distance functions such as

$$d\big(\mathcal{M}, \{D_1, \ldots, D_4\}\big) = \sum_{i=1}^{4} c_i \cdot d^u_\Sigma(\mathcal{M}, D_i).$$

Now, the investment policy among the eight possible policies that are shown in the table of Example 11 is determined by the minimal value of the distance function, and this depends on the actual values of the preference factors c_i.

Note 6 (Schema integration of multiple sources). In the context of integration systems of multiple sources, inconsistencies may occur not only because of contradictory information among the sources, but also by the need to relate different terminologies and concepts used by the sources. This is done by *mediator-based systems* [32] that integrate independent sources

[11] The decision which facility to choose requires further preference criteria. Summation of distances, e.g., prefers $\mathcal{M}_6$ and $\mathcal{M}_7$ over $\mathcal{M}_4$, thus t and p are preferred over s.

containing information about a common domain using different schemas. Such mediators consist of an alphabet, called the global schema (representing the global information), and a set of rules that link the information of the sources with the global schema. This frequently requires appropriate definitions, in which the relations of one schema are expressed in terms of another schema. There are two common methods to define these relations: One, called *global-as-view* [45], expresses the relations of the global schema in terms of those of the sources. The other method, called *local-as-view* [34], defines each source relation in terms of the relations of the global schema. This process of *schema integration* (or *ontology integration*) is also vital for data exchange, where data is shipped from a source database in order to populate a target schema.

A great deal of work is done in the context of schema integration (see [32]). As the next example shows, distance-based considerations may be incorporated for this purpose as well.

Example 13. Suppose that the database considered in Section 2.1 is distributed over two sites that store data about employees in different locations (say, New-York and New-Jersey). In this case, the information may be divided as follows:

$$\begin{array}{ll} \mathsf{source1}: & \mathtt{empNY}(Alice),\ \mathtt{sal}(Alice, 1000),\ \mathtt{director}(Alice) \\ \mathsf{source2}: & \mathtt{empNJ}(Bob),\quad \mathtt{sal}(Bob, 1000) \end{array}$$

Also, using a global-as-view approach, we have a rule for relating the local vocabularies:

$$\forall x\ (\mathtt{empNY}(x) \vee \mathtt{empNJ}(x) \rightarrow \mathtt{emp}(x))$$

and the same integrity constraint as before (specified in the global language):

$$\forall x \forall y \forall z_1 \forall z_2\ \big(\mathtt{sal}(x, z_1) \wedge \mathtt{sal}(y, z_2) \wedge \mathtt{director}(x) \wedge \neg \mathtt{director}(y)\big) \rightarrow z_1 > z_2$$

Taking the last two rules as the set of global integrity constraints ($\mathcal{IC}$) together with d^2 as the distance function and Σ as the aggregation function, we get – in terms of the global language – the same optimal repairs as those in Example 8. Thus, for instance, conservative answers to the queries $\mathtt{emp}(x)$ and $\mathtt{sal}(Alice, 1000)$ are $\{Alice, Bob\}$ and 'no', respectively.

Merging strategies of constraint belief-bases like those mentioned in this section, as well as some related complexity results, are discussed in detail in [29, 30].

5 Related Works

Distance minimization is a primary principle behind many systems for information handling. Back in the 1980's distance-based approaches have been considered e.g., by Dalal [16], Winslett [49], and others, in the context of belief revision. In the database systems point of view, a notion coinciding with Winslett's revision model was first introduced in the seminal paper of Arenas, Bertossi, and Chomicki [1], who presented a model theoretic definition of consistent answers to a query posed to an inconsistent database. Following this work, many other proposals for database repair and consistent query answering have emerged. Most of the proposals are based on the idea that the set of database tuples either inserted to- or deleted from the database instance in order to restore its consistency has to be made minimal under set inclusion [1, 2, 9, 10, 14, 18, 27] or cardinality [2, 4, 5, 35]. Clearly, these are domain independent considerations, in the sense that they are applied to every database regardless of the nature of its information. Representing cardinality-based considerations in our framework is discussed in Section 3.3. Corresponding computations by disjunctive logic programs and stable model semantics are reported in [2, 5].

In contrast to cardinality-based distance semantics, the following definition and proposition show that even in the propositional case, set inclusion considerations deviate from our framework:

Definition 17. Let $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ be a propositional database, and let $A, B \in \mathsf{Potential}(\mathcal{DB})$. As $\mathcal{DB}$ is propositional, the constraint components of A and B are empty, so we identify A and B with their (variable-free) update components.

- A is *inclusion-preferred* over B ($A \prec_{\mathcal{D}} B$), if $\big((A \setminus \mathcal{D}) \cup (\mathcal{D} \setminus A)\big) \subset \big((B \setminus \mathcal{D}) \cup (\mathcal{D} \setminus B)\big)$.
- The $\prec_{\mathcal{D}}$-minimal elements of $\mathsf{Potential}(\mathcal{DB})$ are called *inclusion-optimal*.

As a distance function d induces a total order on $\mathsf{Potential}(\mathcal{DB})$ (determined for each $A \in \mathsf{Potential}(\mathcal{DB})$ by $d(A, \mathcal{D})$), while $\prec_D$ is only a partial order on $\mathsf{Potential}(\mathcal{DB})$, it is quite obvious that preferences based on set inclusion cannot be simulated by distance functions. Moreover,

Proposition 10. *There is no distance function of the form d_f, obtained by a pseudo distance d and an aggregation function f as in Definition 13(d), such that the inclusion-optimal solutions of a propositional database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ are those that are d_f-closest to $\mathcal{D}$.*

Proof. Consider the database $\mathcal{DB} = (\{p, q, r\}, \{p \rightarrow \neg q,\ p \rightarrow \neg r\})$. Here, $\mathsf{Potential}(\mathcal{DB}) = \{\{p\}, \{q\}, \{r\}, \{q, r\}\}$, and so the inclusion-optimal potential repairs of $\mathcal{DB}$ are $\{p\}$ and $\{q, r\}$. Now, let $m = \max\{d(p, q) \mid p, q \in \mathsf{GAtoms}\}$. Then: $d_f(\{p\}, \{p, q, r\}) = d_f(\{r\}, \{p, q, r\}) = f(\{\frac{1}{2}m, \frac{1}{2}m\})$, thus $\{p\}$ and $\{r\}$ are equally distant from the original database $\mathcal{D} = \{p, q, r\}$. It follows that either both $\{p\}$ and $\{r\}$ are d_f-closest to $\mathcal{D}$ (in which case they are repairs of $\mathcal{DB}$) or neither of them is d_f-closest to $\mathcal{D}$. But $\{r\}$ is not inclusion-optimal while $\{p\}$ is. □

Domain independent repairs may be extended in various ways to make them domain dependent. For instance, the distance function d^u_Σ, considered in Section 3.3 in the context of pairwise (cardinality-based) repairs, may by generalized by attaching different weights to different predicates, expressing the idea that for restoring consistency it may be more costly to insert or remove tuples of a certain predicate than to change tuples of other predicates. This yields a generalized Hamming distance, defined by the sum of the weights of the elements in the symmetric difference of the relevant sets. This kind of distance-based repairing is considered, e.g., in [36].

As noted in Section 3.4, our framework can also capture other domain dependent techniques for database repair, recently considered, e.g., in [8, 24, 46, 47]. Below, we describe the relations to some of these methods in greater details. For this, we briefly recall the basic definitions behind Wijsen's approach [46, 47] of repairing by value modifications. For simplicity, assume that the language $\mathcal{L}$ consists of one predicate with arity n, i.e., a fixed schema $\langle A_1, \ldots, A_n \rangle$ of distinct attributes. A *tableaux* is a finite set T of tuples $\langle t_1, \ldots, t_m \rangle$, where each t_i is a term.[12] A tableau is called *linear* if no variable occurs in it more than once; it is called a *relation* if all its terms are ground (i.e., without variables). Wijsen's approach is based on *tableaux homomorphisms*. Given two tableaux T_1 and T_2, a homomorphism from T_1 to T_2 is a variable substitution θ on the variables of T_1, such that $\theta(T_1) \subseteq T_2$. Such a homomorphism is called one-to-one if it does not identify distinct tuples of T_1. Now, given a relation P and a set $\mathcal{IC}$ of integrity constraints, a *fix* of P under $\mathcal{IC}$ is a maximal tableaux P' for which (1) there is a one-to-one homomorphism to P, and (2) there is a (not necessarily one-to-one) homomorphism to some relation R that satisfies $\mathcal{IC}$. If such a relation R is minimal under set inclusion, it is a repair of P in the sense of Wijsen (an uprepair, in his terms).

The correspondence between Wijsen's approach and ours is straightforward: In our sense, a tableau T is the first component of a database update (referring to a single relation), and the second component of an update, namely the constraint set, represents tableau

[12] As the language is function-free, these terms must be atomic, i.e., constants or variables.

homomorphisms on T. If a tableau is homomorphic to a consistent relation then, in our terminology, it is a potential repair. In this respect, Wijsen's fixes may be viewed as compact representations of potential repairs.

The main difference between the two methods is that Wijsen uses set inclusion as the underlying preference criterion, while in our framework distance functions are used for choosing the optimal repairs. A variant of Wijsen's approach, which defines uprepairs by cardinality minimization rather than by set inclusion minimization, is easily simulated in our framework by taking, e.g., d^1 and Σ as the underlying distance and aggregation functions, respectively.

Incidentally, domain-dependent repairs with minimal cardinality and set-minimal repairs have also been considered in the closely related construct introduced in [24] (see [48] for a discussion on the similarities and differences between the repair methods). In this case as well, our methods offer a considerable flexibility in the determination of the repair strategy, by allowing to incorporate domain-specific considerations (such as those in the definition of d^2 above).

Another domain dependent approach for restoring database consistency, based on tuple updates, is presented in [8]. This time, the tableaux homomorphisms approach is traded by a quantitative attitude that is based on the *values* of the attributes. We demonstrate this by the following example:

Example 14. [8, Examples 1,4] Consider the following database of traffic network.

`Traffic`

`Time`	`LinkName`	`Type`	`Flow`
9:00am	a	0	1100
9:00am	b	1	900
9:30am	b	1	850

Suppose that the maximum capacities of links of type 0 and 1 are 1000 and 1500, respectively. The database above is not consistent with respect to these constraints, as they are not obeyed by the first tuple.

According to [8], the distance between two tuples is determined by the (weighted sum of the squares of the) differences between their numerical values. Thus, e.g., the difference between the tuples (9:00am,a,0,1100) and (9:00am,a,1,1000) is $c_1 \cdot 1^2 + c_2 \cdot 100^2$ for some fixed coefficients c_1 and c_2. Using our terminology, the distance between the original database instance and a potential repair of it is equal to the weighted sum of the differences between the tuples in those sets, having the same key values. Now, repairs are the potential repairs with minimal distance to the original database. In Example 14, then, two potential repairs are:

`Traffic`$'$

`Time`	`LinkName`	`Type`	`Flow`
9:00am	a	1	1100
9:00am	b	1	900
9:30am	b	1	850

`Traffic`$''$

`Time`	`LinkName`	`Type`	`Flow`
9:00am	a	0	1000
9:00am	b	1	900
9:30am	b	1	850

where `Traffic`$'$ changes the type of link 'a' to 1, and `Traffic`$''$ reduces its flow capacity to 1000. For $c_1 = 1$ and $c_2 = 10^{-5}$ the distances from `Traffic` are $1 \cdot 1^2$ and $10^{-5} \cdot 100^2$, respectively, so in this case `Traffic`$''$ is preferred over `Traffic`$'$.

Clearly, the approach of [8] is represented in our framework, where minimal distances correspond to least square values. The tuple matching according to [8] is done by key values, and this is a plausible matching criterion, as in [8] key attributes are not updateable. Note,

however, that in our framework updates are not limited to numerical values only. More substantially, our framework permits more general repairs. In Example 14, for instance, the d^2_Σ-semantics (in which changes of key attributes are more expensive than other changes) will change (9:00am,a,0,1100) to either (9:00am,a,x,1100) for some $x \neq 0$ (which corresponds to $\texttt{Traffic}'$ above), or to (9:00am,a,0,y) for some $y \leq 1000$. The latter is clearly a more general repair than $\texttt{Traffic}''$, as it is not committed to a specific flow capacity of link 'a' (i.e., 1000 in case of $\texttt{Traffic}''$). [13]

6 Conclusion and Future Work

Data processing by distance considerations is not a new idea, and it has been used mainly in the context of query answering [2, 6, 37] integration of constraint belief-sets [29, 30] and operators for belief revision [19, 31, 39, 44]. In this paper, we introduced a uniform framework for representing, comparing and implementing different approaches for these contexts. Another advantage of our approach is that it opens the door to many new methods that are induced by known distance definitions. This is particularly useful in the context of database repairing, where so far most of the formalisms in the literature that involve distance-based semantics are domain independent, while in many practical cases domain dependent repairs are more adequate. Typical cases for this are census, demographic, and experimental data, where faulty information need to be altered (rather than removed altogether) in order to meet certain integrity constraints. The new forms of repairs offered by our framework provide a step forward towards more intuitive solutions to such cases, mainly as the notion of closeness can be captured in more subtle ways, and erroneous components of the data can be detected and updated without violating the valid fragment of the information.

The message of this paper is, therefore, that it is useful to think in terms of distances to express preferences among repairs and that different choices of distances lead to different preferences that can be applied in different scenarios. Having this said, it seems that distance semantics per-se is not strong enough for handling many practical cases. This is so not only because of the considerable amount of repairs that are usually induced by it, but also due to the fact that, in many cases, distance considerations cannot completely capture every aspect of the underlying database information, and so different repairs may have different likelihood. To see this, suppose that the following information is part of a household data, associated to a census:

Person

Name	Age	Marital Status
David	71	Married
Ann	6	Married
Tom	20	Bachelor

Here, the second tuple violates a constraint that people under 16 cannot be married. The common domain independent repairs, applying set inclusion or minimal cardinality of insertions/retractions, will remove the problematic tuple. However, this does not seem to be an appropriate solution in this case, as all the information about Ann will be lost. By using the distance-based techniques for repairing by attribute modifications, we will get, according to d^1_Σ (as well as by d^2_Σ), the following two repairs:

[13] See also the discussion in the paragraph below Definition 7.

Person′

Name	Age	Marital Status
David	71	Married
Ann	x	Married
Tom	20	Bachelor

Person″

Name	Age	Marital Status
David	71	Married
Ann	6	y
Tom	20	Bachelor

In the repair on the left-hand side the problem is fixed by changing the age of Ann to some $x \geq 16$, and according to the repair in the right-hand side, Ann's marital status is changed to some y other than 'Married' (or any other status that requires marriage, such as 'Divorced' or 'Widower').

While the distance functions considered above imply that Person′ and Person″ are equally good repairs, additional information may help to conclude that one repair is more plausible than the other. For instance, information about assets or hobbies usually provides some indication whether the underlying person is an adult or a child. As this kind of information cannot make an exclusive discrimination between the two options, integrity constraints are not useful here. Extending the distance functions by probability factors looks somewhat 'ad-hoc' and cumbersome. More promising approaches for resolving this problem are to adopt learning techniques for 'pulling out' the most plausible repairs, or to incorporate declarative theories that give further indications on 'how to repair'. Using such methods in our framework is a subject for a future research.

References

1. M. Arenas, L. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *Proc. PODS'99*, pages 68–79. ACM Press, 1999.
2. M. Arenas, L. Bertossi, and J. Chomicki. Answer sets for consistent query answering in inconsistent databases. *Theory and Practice of Logic Programming.*, 3(4–5):393–424, 2003.
3. O. Arieli, M. Denecker, and M. Bruynooghe. Distance-based repairs of databasese. In M.Fisher et al., editor, *Proc. JELIA'06*, number 4160 in LNAI, pages 43–55. Springer, 2006.
4. O. Arieli, M. Denecker, B. Van Nuffelen, and M. Bruynooghe. Coherent integration of databases by abductive logic programming. *Artificial Intelligence Research*, 21:245–286, 2004.
5. O. Arieli, M. Denecker, B. Van Nuffelen, and M. Bruynooghe. Computational methods for database repair by signed formulae. *Annals of Mathematics and Artificial Intelligence*, 46(1–2):4–37, 2006.
6. L. Bertossi. Consistent query answering in databases. *SIGMOD Records*, 35(2):68–76, 2006.
7. L. Bertossi. Some research directions in consistent query answering: A vision. In J. Chomicki and J. Wijsen, editors, *Pre-proceedings of the EDBT'06 Workshop on Inconsistency and Incompleteness in Databases*, pages 109–113, 2006.
8. L. Bertossi, L. Bravo, E. Franconi, and A. Lopatenko. Complexity and approximation of fixing numerical attributes in databases under integrity constraints. In *Proc. DBLP'05*, number 3774 in LNCS, pages 262–278. Springer, 2005.
9. L. Bertossi, J. Chomicki, A. Cortés, and C. Gutierrez. Consistent answers from integrated data sources. In *Proc. FQAS'2002*, number 2522 in LNCS, pages 71–85. Springer, 2002.
10. L. Bertossi and C. Schwind. Database repairs and analytic tableau. *Annals of Mathematics and Artificial Intelligence*, 40(1–2):5–35, 2004.
11. A. Cali, D. Lembo, and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. PODS'03*, pages 260–271. ACM Press, 2003.
12. J Chomicki. Consistent query answering: Five easy pieces. In *Proc. ICDT'07*, number 4353 in LNCS, pages 1–17. Springer, 2007.
13. J. Chomicki and J. Marchinkowski. On the computational complexity of minimal-change integrity maintenance in relational databases. In L. Bertossi, A. Hunter, and T. Schaub, editors, *Inconsistency Tolerance*, number 3300 in LNCS, pages 119–150. Springer, 2004.
14. J. Chomicki and J. Marchinkowski. Minimal-change integrity maintenance using tuple deletion. *Information and Computation*, 197(1–2):90–121, 2005.

15. J. Chomicki, J. Marchinkowski, and S. Staworko. Computing consistent query answers using conflict hypergraphs. In *Proc. CIKM'04*, pages 417–426, 2004.
16. M. Dalal. Investigations into a theory of knowledge base revision. In *Proc. AAAI'88*, pages 475–479. AAAI Press, 1988.
17. E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.
18. S. de Amo, W. A. Carnielli, and J. Marcos. A logical framework for integrating inconsistent information in multiple databases. In *Proc. FoIKS'02*, number 2284 in LNCS, pages 67–84. Springer, 2002.
19. J. Delgrande. Preliminary considerations on the modelling of belief change operators by metric spaces. In *Proc. NMR'04*, pages 118–125, 2004.
20. J. Dieudonné, editor. *Foundations of Modern Analysis*. Academic Press, 1969.
21. T. Eiter. Data integration and answer set programming. In *Proc. LPNMR'05*, number 3662 in LNCS, pages 13–25. Springer, 2005.
22. T. Eiter and H. Mannila. Distance measure for point sets and their computation. *Acta Informatica*, 34:109–133, 1997.
23. B. Fazzinga, S. Flesca, F. Furfaro, and F. Parisi. DART: A data acquisition and repairing tool. In T. Grust et al., editor, *Current Trends in Database Technology – Proc. of the EDBT'06 Workshops*, number 4254 in LNCS, pages 297–317. Springer, 2006.
24. S. Flesca, F. Furfaro, and F. Parisi. Consistent query answers on numerical databases under aggregate constraints. In *Proc. 10th Int. Symp. on Database Programming Languages* (DBLP'05), number 3774 in LNCS, pages 279–294. Springer, 2005.
25. E. Franconi, A. L. Palma, N. Leone, D. Perri, and F. Scarcello. Census data repair: A challenging application of disjunctive logic programming. In *Proc. LPAR'01*, number 2250 in LNCS, pages 561–578. Springer, 2001.
26. G. Greco, S. Greco, and E. Zumpano. A logic programming approach to the integration, repairing and querying of inconsistent databases. In *Proc. ICLP'01*, number 2237 in LNCS, pages 348–363. Springer, 2001.
27. S. Greco and E. Zumpano. Querying inconsistent databases. In *Proc. LPAR'2000*, number 1955 in LNAI, pages 308–325. Springer, 2000.
28. A. Grove. Two modellings for theory change. *Journal of Philosophical Logic*, 17:157–180, 1988.
29. S. Konieczny, J. Lang, and P. Marquis. Distance-based merging: A general framework and some complexity results. In *Proc KR'02*, pages 97–108. Morgan Kaufmann, 2002.
30. S. Konieczny and R. Pino Pérez. Merging information under constraints: A logical framework. *Logic and Computation*, 12(5):773–808, 2002.
31. D. Lehmann, M. Magidor, and K. Schlechta. Distance semantics for belief revision. *Journal of Symbolic Logic*, 66(1):295–317, 2001.
32. M. Lenzerini. Data integration: A theoretical perspective. In *Proc. PODS'02*, pages 233–246. ACM Press, 2002.
33. N. Leone, T. Eiter, W. Faber, M. Fink, G. Gottlob, and G. Greco. Boosting information integration: The INFOMIX system. In *Proc. SEBD'05*, pages 55–66, 2005.
34. A. Levy, A. Rajaraman, and Ordille J.J. Querying heterogeneous information sources using source descriptions. In *Proc. VLDB-96*, pages 251–262. Morgan Kaufmann, 1996.
35. P. Liberatore and M. Schaerf. BReLS: A system for the integration of knowledge bases. In *Proc. KR'2000*, pages 145–152. Morgan Kaufmann, 2000.
36. A. Lopatenko and L. Bertossi. Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. An extended version of [37]; Corr Archiv paper cs.DB/0604002.
37. A. Lopatenko and L. Bertossi. Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. In *Proc. ICDT'07*, number 4353 in LNCS, pages 179–193. Springer, 2007.
38. S.H. Nienhuys-Cheng. Distance between Herbrand interpretations: A measure for approximations to a target concept. In *Proc. ILP'97*, number 1297 in LNCS, pages 213–226. Springer, 1997.
39. P. Peppas, S. Chopra, and N. Foo. Distance semantics for relevance-sensitive belief revision. In *Proc. KR'04*, pages 319–328. AAAI Press, 2004.

40. J. Ramon. Clustering and instance based learning in first order logic, 2002. Ph.D. Thesis, K.U.Leuven.
41. J. Ramon and M. Bruynooghe. A polynomial time computable metric between point sets. *Acta Informatica*, 37(10):765–780, 2001.
42. R. Reiter. On closed world databases. In *Logic and Databases*, pages 55–76. Plenum Press, 1978.
43. R. Reiter. Towards a logical reconstruction of relational database theory. In *On Conceptual Modelling (Intervale)*, pages 191–233, 1982.
44. W. Spohn. Ordinal conditional functions: a dynamic theory of epistemic states. In W. L. Harper and B. Skyrms, editors, *Belief Change and Statistics*, volume II, pages 105–134. Kluwer, 1988.
45. J. Ullman. Information integration using logical views. *Theoretical Computer Science*, 239(2):189–210, 2000.
46. J. Wijsen. Making more out of an inconsistent database. In *Proc. ADBIS'04*, number 3255 in LNCS, pages 291–305. Springer, 2004.
47. J. Wijsen. Database repairing using updates. *ACM Trans. on Database Systems*, 30(3):722–768, 2005.
48. J. Wijsen. A note on database repairing by value modification. In J. Chomicki and J. Wijsen, editors, *Pre-proceedings of the EDBT'06 Workshop on Inconsistency and Incompleteness in Databases*, pages 104–107, 2006.
49. M. Winslett. Reasoning about action using a possible models approach. In *Proc. AAAI'88*, pages 89–93. AAAI press, 1988.