Distance-Based Repairs of Databases

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. We 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 various 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 and Motivation

Inconsistency of constraint data-sources is a widespread phenomenon. Restoring information consistency (or *repairing* the database) 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 [33], 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 [12] and [35]. In the context of database systems, this notion was introduced by [1], and then considered by many others, including [2–4, 6, 7, 13, 22, 21, 29, 34]. Some implementations of these methods are reported in [3, 18, 19, 28]. 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 close as possible' to the original (inconsistent) data. This implies that database repairing can be specified in terms of distance semantics, using appropriate metrics.

In this paper, we identify distance-based semantics at the heart of a vast amount of repairing methods, and introduce a corresponding framework for data

repair. In this respect, we follow Bertossi's remark [5], 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*". Indeed, the representation of different repairing methods as distance-based formalisms provides a common ground for relating them. Moreover, we show that the same distance-based considerations are not only the essence of database repairing and consistent query answering, but are also the nucleus of many approaches for belief revision and data integration. In this respect, this work is not restricted to databases only.

The rest of this paper 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 existing repairing methods and defining some new ones. In Section 4 we relate database repairing to different methods of merging independent data-sources. In Section 5 we conclude.

2 Database Repair as a Distance Minimization Problem

Let $\mathcal{L}$ be a first-order language with $\mathcal{P}$ its underlying set of predicates.

Definition 1. A *database* $\mathcal{DB}$ is a pair $(\mathcal{D}, \mathcal{IC})$, where $\mathcal{D}$ is a finite set of ground atomic facts (i.e., atomic formulas without variables) whose predicate names are in $\mathcal{P}$, and $\mathcal{IC}$ is a finite and consistent set of formulae in $\mathcal{L}$.

The set $\mathcal{D}$ in the definition above is called *database instance* and its elements are called *facts*. The meaning of $\mathcal{D}$ is usually determined by the conjunction of its facts augmented with Reiter's closed world assumption [33]. The formulas in $\mathcal{IC}$ are called *integrity constraints*. These formulas specify conditions that should be satisfied, with respect to some underlying semantics $\mathcal{S}$, by all the database facts. We denote this by $\mathcal{D} \models^{\mathcal{S}} \mathcal{IC}$. Common definitions for $\mathcal{S}$ are the standard two-valued semantics, the minimal Herbrand model semantics, the stable model semantics [20], Kleene's three-valued semantics [24], or any other multiple-valued semantics [23]. A semantics $\mathcal{S}$ defines, for a set Γ of formulas in $\mathcal{L}$, the $\mathcal{S}$-*models* of Γ, i.e., a set $mod^{\mathcal{S}}(\Gamma)$ of valuations that satisfy all the formulas in Γ. In this respect $\mathcal{D} \models^{\mathcal{S}} \mathcal{IC}$ means that every element of $mod^{\mathcal{S}}(\mathcal{D})$ is also an $\mathcal{S}$-model of every integrity constraint in $\mathcal{IC}$.

Definition 2. A database $(\mathcal{D}, \mathcal{IC})$ is *consistent* with respect to a semantics $\mathcal{S}$ ($\mathcal{S}$-consistent, for short), if $\mathcal{D} \models^{\mathcal{S}} \mathcal{IC}$.

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. For this, given a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$, we denote by $\mathsf{HU}(\mathcal{DB})$ the ground terms in the Herbrand universe of $\mathcal{DB}$, θ_i denotes substitutions of variables by elements in $\mathsf{HU}(\mathcal{DB})$, and $\mathsf{Atoms}(\mathcal{DB})$ denotes the atomic formulas

that appear in the formulas of $\mathcal{D} \cup \mathcal{IC}$. For each $P(c_1, \ldots, c_n) \in \mathsf{Atoms}(\mathcal{DB})$, let

$$\mathsf{Upd}\big(P(c_1, \ldots, c_n)\big) = \Big\{ P(d_1, \ldots, d_n) \mid \exists 1 \leq j \leq n \; d_j \in \mathsf{HU}(\mathcal{DB}) \text{ and } \exists\, \theta_1, \theta_2 \text{ such that } \theta_1\big(P(d_1, \ldots, d_n)\big) = \theta_2\big(P(c_1, \ldots, c_n)\big) \Big\}.$$

An *update* $\mathcal{U}$ of $\mathcal{D}$ is a set that consists of zero or more elements from $\mathsf{Upd}(A)$ for each $A \in \mathsf{Atoms}(\mathcal{DB})$. The set of all the updates of $\mathcal{D}$ is denoted by $\mathsf{Upd}(\mathcal{D})$.

The intuition behind this definition is simple: the predicates in $\mathsf{Atoms}(\mathcal{DB})$ are those that are 'known' to the database, thus they are the relations that potentially appear in a repaired database. Now, the elements in $\mathsf{Upd}\big(P(c_1, \ldots, c_n)\big)$ represent the possible updates of $P(c_1, \ldots, c_n)$. Note that $\{c_i\}$ and $\{d_i\}$ are atomic terms (constants or variables), so the role of the substitutions in the definition of Upd is twofold: to introduce constants instead of variables in predicate tuples (in case that the values are known) and to replace constants by variables (in case that there are wrong values in a tuple and the correct values are unknown. Here, variables may be intuitively regarded as missing (null) values). By this, only erroneous fragments of tuples are modified.[3] The condition that in every tuple at least one component belongs to $\mathsf{HU}(\mathcal{DB})$ is meant to exclude degenerated cases, in which a tuple consists of null values only.

Note 1. Unless all the arguments of an atom A are variables, $A \in \mathsf{Upd}(A)$. Thus, if for some $A \in \mathcal{D}$ and an update $\mathcal{U}$, $\mathcal{U} \cap \mathsf{Upd}(A) = \{A\}$, A remains unchanged. Also, if $\mathcal{U} \cap \mathsf{Upd}(A) = \emptyset$, A is deleted from $\mathcal{D}$, and if $\mathcal{U} \cap \mathsf{Upd}(A) = \{A_1, \ldots, A_n\}$, A is replaced by $n \geq 1$ new facts A_i. Insertions to the database also occur when $\mathcal{U} \cap \mathsf{Upd}(A)$ is not empty for some atom A not in the database instance.

A *potential repair* $\mathcal{R}$ of $\mathcal{DB}$ is an update of $\mathcal{D}$ that preserves $\mathcal{IC}$ with respect to $\mathcal{S}$ ($\mathcal{R} \models^{\mathcal{S}} \mathcal{IC}$). The set of all the potential repairs of $\mathcal{DB}$ is denoted by $\mathsf{Rep}(\mathcal{DB})$.

Example 2. [4] Given the following database instance

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

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. In this case (using some abbreviations with obvious meanings), we have that the updates of the ground facts in the database are the following:

$\mathsf{Upd}(emp(Alice)) = \{emp(Alice)\}$,
$\mathsf{Upd}(emp(Bob)) = \{emp(Bob)\}$,
$\mathsf{Upd}(dir(Alice)) = \{dir(Alice)\}$,
$\mathsf{Upd}(sal(Alice, 1000)) = \{sal(Alice, 1000), sal(x_A, 1000), sal(Alice, val_1)\}$,
$\mathsf{Upd}(sal(Bob, 1000)) = \{sal(Bob, 1000), sal(x_B, 1000), sal(Bob, val_2)\}$. [4]

[3] See also Wijsen's notion of homomorphisms of tableaux [34].

[4] For the other atoms in $\mathsf{Atoms}(\mathcal{DB})$ we have that $\mathsf{Upd}(emp(x)) = \mathsf{Upd}(emp(Alice)) \cup \mathsf{Upd}(emp(Bob))$ and $\mathsf{Upd}(sal(x, y)) = \mathsf{Upd}(sal(Alice, 1000)) \cup \mathsf{Upd}(sal(Bob, 1000))$.

Among the possible updates of $\mathcal{D}$ we therefore have the following sets:

$\mathcal{U}_1 = \{emp(Alice), emp(Bob), sal(Alice, 1000), sal(Bob, 1000)\}$,
$\mathcal{U}_2 = \{emp(Alice), emp(Bob), dir(Alice), sal(Alice, val_1), sal(Bob, 1000)\}$,
$\mathcal{U}_3 = \{emp(Alice), emp(Bob), dir(Alice), sal(Alice, 1000), sal(Bob, val_2)\}$.

Note that $\mathcal{U}_1$ is obtained by retracting the fact that Alice is a director (so in this case $\mathsf{Upd}(dir(Alice))$ has no representatives), while $\mathcal{U}_2$ and $\mathcal{U}_3$ cause modifications in the salary of Alice and Bob (respectively). Note also that all these updates are also potential repairs, provided that $val_1 > 1000$ and $val_2 < 1000$.

For selecting the best potential repairs we require that the repaired information should be 'as close as possible' to the original one. Implicitly, then, this criterion involves distance-based considerations and a corresponding metric.

Definition 3. A total function $d : U \times U \to \mathbb{R}^+$ is called *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 ($\forall u, v, w \in U\ d(u, v) \leq d(u, w) + d(w, v)$).

Definition 4. A *repair context* for a language $\mathcal{L}$ is a pair $\mathfrak{R} = \langle \models^{\mathcal{S}}, d \rangle$, where $\models^{\mathcal{S}}$ is the entailment relation induced by the underlying semantics $\mathcal{S}$ and d is a pseudo distance on the power set $2^{\mathcal{L}}$ of the well-formed formulae in $\mathcal{L}$.

Repair contexts are parametrized descriptions on how to repair databases. Given a repair context $\mathfrak{R} = \langle \models^{\mathcal{S}}, d \rangle$, the *repairs* of a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ are the instances that $\mathcal{S}$-satisfy $\mathcal{IC}$ and that are d-closest to $\mathcal{D}$. Formally:

Definition 5. The *repairs* of a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ with respect to a repair context $\mathfrak{R} = \langle \models^{\mathcal{S}}, d \rangle$, are the elements of the following set:

$$\Delta_{\mathfrak{R}}(\mathcal{DB}) = \big\{ \mathcal{R} \in \mathsf{Rep}(\mathcal{DB}) \mid \forall \mathcal{R}' \in \mathsf{Rep}(\mathcal{DB})\ d(\mathcal{R}, \mathcal{D}) \leq d(\mathcal{R}', \mathcal{D}) \big\}.$$

Database repairs induce corresponding notions of query answering:

Definition 6. 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 $\mathfrak{R} = \langle \models^{\mathcal{S}}, d \rangle$ be a repair context, 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 $\mathcal{R} \in \Delta_{\mathfrak{R}}(\mathcal{DB})$ s.t. $\mathcal{R} \models^{\mathcal{S}} \mathcal{Q}[c_1/x_1, \ldots, c_n/x_n])$.
- A tuple $\langle c_1, \ldots, c_n \rangle$ is a *conservative answer* (or a *consistent query answer*) for $\mathcal{Q}$ if $\mathcal{R} \models^{\mathcal{S}} \mathcal{Q}[c_1/x_1, \ldots, c_n/x_n])$ for every $\mathcal{R} \in \Delta_{\mathfrak{R}}(\mathcal{DB})$.

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:

Definition 7. Let d be a distance function on $\mathcal{L}$. For $A, B \in 2^{\mathcal{L}}$, define:

– *The Hausdorff distance* [15]:

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

– *Eiter and Mannila's distance* [17]:

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

The following proposition recalls some known facts about these distances:

Proposition 1. *The Hausdorff distance is a distance function on $2^{\mathcal{L}}$ and Eiter–Mannila's distance is a pseudo distance on $2^{\mathcal{L}}$.*

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 8. A *numeric aggregation function* f is a total function that accepts multisets of real numbers and returns a real number. Also, f is non-decreasing in the values of its argument,[5] $f(\{x_1,\ldots,x_n\}) = 0$ if $x_1 = \ldots = x_n = 0$, and $\forall x \in \mathbb{R}\ f(\{x\}) = x$.

Definition 9. Let $\mathcal{DB}$ be a database, $A, B \subseteq \mathsf{Atoms}(\mathcal{DB})$, d a (pseudo) distance on the formulae of $\mathcal{L}$, and f a numeric aggregation function.

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

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

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

where, for every set S, $d(x,S) = \frac{1}{2}\max\{d(y,z) \mid y,z \in \mathsf{Atoms}(\mathcal{DB})\}$.

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

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

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

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

The aggregation function in Definition 8 may be, 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).

Proposition 2. *The function d_f in Definition 9(d) is a pseudo distance on $2^{\mathcal{L}}$.*[7]

3.2 Aggregation-based repairs

Definition 10. An *aggregation-based repair context* is a triple $\mathfrak{R} = \langle \models^{\mathcal{S}}, d, f \rangle$, where $\models^{\mathcal{S}}$ is the entailment relation induced by $\mathcal{S}$, d is a pseudo distance on $\mathcal{L}$, and f is a numeric aggregation function.

Note 2. If $\mathfrak{R} = \langle \models^{\mathcal{S}}, d, f \rangle$ is an aggregation-based repair context in the sense of Definition 10, then $\mathfrak{R} = \langle \models^{\mathcal{S}}, d_f \rangle$, where d_f is obtained from d and f by Definition 9(d), is a repair context in the sense of Definition 4. This is so, since d_f is a pseudo distance on $2^{\mathcal{L}}$ (by Proposition 2).

Definition 11. The *repairs* of a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ with respect to an aggregation-based repair context $\mathfrak{R} = \langle \models^{\mathcal{S}}, d, f \rangle$ are the elements of the set

$$\Delta_{\mathfrak{R}}(\mathcal{DB}) = \big\{ \mathcal{R} \in \mathsf{Rep}(\mathcal{DB}) \mid \forall \mathcal{R}' \in \mathsf{Rep}(\mathcal{DB})\ d_f(\mathcal{R}, \mathcal{D}) \leq d_f(\mathcal{R}', \mathcal{D}) \big\}.$$

By Note 2, Definition 11 is a particular case of Definition 5 for aggregation-based distance functions (and aggregation-based repair contexts).

Example 3. Consider again the database $\mathcal{DB} = (\{p, r\}, \{p \rightarrow q\})$ of Example 1, and let $\mathfrak{R} = \langle \models, d^u, \Sigma \rangle$ be an aggregation-based repair context, where d^u is a distance function on the atomic formulas of $\mathcal{L}$, defined by $d^u(s_1, s_2) = 0$ if $s_1 = s_2$, and $d^u(s_1, s_2) = 1$ otherwise. The six potential repairs of $\mathcal{DB}$ and their distances from $\mathcal{D} = \{p, r\}$ are given in the table below.

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 $\mathcal{R}_1 = \{p, q, r\}$ and $\mathcal{R}_5 = \{r\}$. Among the potential repairs, these repairs require a minimal amount of modifications in $\mathcal{D}$.[8] Thus, e.g., r conservatively (and so credulously) follows from $\mathcal{DB}$, and q credulously (but not conservatively) follows from $\mathcal{DB}$.

[7] Due to lack of space proofs are omitted. Full proofs will appear in an extended version of this paper.

[8] As Proposition 4 below shows, this is not a coincidence.

Definition 12. An aggregation function f such that $f(\{x_1, \ldots, x_n\}) = 0$ only if $x_1 = \ldots = x_n = 0$, is called *strict*. An aggregation-based repair context $\mathfrak{R} = \langle \models^{\mathcal{S}}, d, f \rangle$ is strict if f is strict.

Note that as distances are non-negative, all the aggregation functions on sets of distances considered above (summation, maximum, median, etc.) are strict.[9]

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

Proposition 3. For every strict aggregation-based repair context $\mathfrak{R}$, if $\mathcal{DB}$ is a consistent database, then $\Delta_{\mathfrak{R}}(\mathcal{DB}) = \{\mathcal{D}\}$.

3.3 Domain independent repairs

A common definition of an aggregation-based repair context is $\mathfrak{R} = \langle \models, d^u, \Sigma \rangle$, where the underlying distance-aggregation function, d^u_Σ, is obtained by a summation of the drastic distances d^u between matched elements (see also Example 3).

Definition 13. The *drastic distance* is $d^u(x, y) = 0$ if $x = y$, else $d^u(x, y) = 1$.

It is easy to verify that d^u and d^u_Σ are distance functions, and both of them are 'blind' to the domain of discourse at hand. Next we show that the metric obtained by d^u_Σ corresponds to the Hamming distance between sets of formulae.[10]

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

The repair context $\mathfrak{R} = \langle \models, d^u, \Sigma \rangle$ corresponds to the repair method introduced in [1], which inspires many other works on (domain independent) database repair (see, e.g., [2, 4, 6, 7, 22, 21, 28]).

Definition 14. [1] A *pairwise*[11] *repair* of $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ is a pair $(\mathsf{Insert}, \mathsf{Retract})$, such that: 1. $\mathsf{Insert} \cap \mathcal{D} = \emptyset$, 2. $\mathsf{Retract} \subseteq \mathcal{D}$, 3. $(\mathcal{D} \cup \mathsf{Insert} \setminus \mathsf{Retract}, \mathcal{IC})$ is a consistent database, 4. $(\mathsf{Insert}, \mathsf{Retract})$ is minimal:[12] there is no pair $(\mathsf{Insert}', \mathsf{Retract}')$ that satisfies conditions 1–3 and for which $|\mathsf{Insert}' \cup \mathsf{Retract}'| < |\mathsf{Insert} \cup \mathsf{Retract}|$.

Proposition 5. Consider a database $\mathcal{DB} = (\mathcal{D}, \mathcal{IC})$ and the repair context $\mathfrak{R} = \langle \models, d^u, \Sigma \rangle$. Then $(\mathsf{Insert}, \mathsf{Retract})$ is a pairwise repair of $\mathcal{DB}$ iff there is a repair $\mathcal{R} \in \Delta_{\mathfrak{R}}(\mathcal{DB})$ s.t. $\mathsf{Insert} = \mathcal{R} \setminus \mathcal{D}$ and $\mathsf{Retract} = \mathcal{D} \setminus \mathcal{R}$.

It is also interesting to check the distance-based functions of Definition 7 when the domain independent 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's distance, on the other hand, is

[9] The minimum function is *not* strict, but it is not useful for repair contexts.
[10] Also known as the symmetric distance, or the Dalal distance [12].
[11] This adjective is added to distinguish this kind of repairs from repairs in our sense.
[12] A different condition may be defined by set inclusion instead of minimal cardinality.

more appropriate in this case, and as in Proposition 5, is it related to pairwise repairing. Indeed, given d^u, the Eiter–Mannila's distance between the original database $\mathcal{D}$ and its repair $\mathcal{R} = \mathcal{D} \cup \mathsf{Insert} \setminus \mathsf{Retract}$ is equal to $\frac{1}{2}(\mathsf{Insert} + \mathsf{Retract})$. In this case we get the Ramon–Bruynooghe matching-based distance [32], which is a distance function (and not only a pseudo distance, cf. Proposition 1).

3.4 Domain-dependent repairs

Consider again the database of Example 2. There are several potential repairs in this case. Let's consider two of them:

$\mathcal{R}_1$: remove all the information about Bob from the database,

$\mathcal{R}_2$: change the information about the salary of Bob.

Note that if we use a domain independent repair context with e.g. d^u as the underlying distance function, each potential repair above is as good as the other one, since the cost of the optimal matching between the original database and the repaired database that is obtained by $\mathcal{R}_1$ is the cost of the two retracted elements ($employee(Bob)$ and $salary(Bob, 1000)$) that cannot be matched to an element in the repaired database, which is $\frac{1}{2} + \frac{1}{2} = 1$. Likewise, the optimal matching between the original database and the repaired database that is obtained by $\mathcal{R}_2$ links $employee(Alice)$, $employee(Bob)$, $salary(Alice, 1000)$ and $director(Alice)$ to the same facts in the repaired database, and relates $salary(Bob, 1000)$ to $salary(Bob, x)$ (for some $x < 1000$). The resulting distance is therefore $0 + 0 + 0{+}0{+}1 = 1$. According to the repair context $\mathfrak{R} = \langle \models, d^u, \Sigma \rangle$, then, both potential repairs have the same priority. However, in this case, the second repair (salary changes) seems more plausible than the first one (employee removal), as it is more realistic here to assume that the problem is due to a typographic error in the salary information. Moreover, $\mathcal{R}_1$ is more drastic, as it causes information loss (Bob is no longer a reported employee). It is clear, then, that simple cardinality considerations are not useful here, and more delicate considerations, that would yield the preference of $\mathcal{R}_2$ over $\mathcal{R}_1$, are required. [13]

A more subtle preference criterion is obtained by the distance function in [30]:

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

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.

[13] The need to rectify an error within a tuple without deleting the whole tuple has been acknowledged in [4] (see Example 6.2 of that paper), and is also the main motivation behind the work of Wijsen on database repairing by updates [34].

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

According to d^1, the distance between the database instance $\mathcal{D}$ of Example 2 and $\mathcal{R}_1$ is still 1, while the distance between $\mathcal{D}$ and $\mathcal{R}_2$ is the same as the distance between $salary(Bob, 1000)$ and $salary(Bob, x)$, which is $\frac{1}{4}(0+1) = \frac{1}{4}$. It follows, then, that now $\mathcal{R}_2$ is preferred over $\mathcal{R}_1$, 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 15. Below we denote primary key values by underscores, and assume, without loss of generality, that they precede the non-key values. Define:

$$d^2\big(P(\underline{t_1}, \ldots, \underline{t_k}, t_{k+1}, \ldots, t_m), Q(\underline{s_1}, \ldots, \underline{s_l}, t_{l+1}, \ldots, t_n)\big) =$$

$$\begin{cases} 1 & \text{if } P \neq Q \text{ or } \exists 1 \leq i \leq k \text{ s.t. } \underline{t_i} \neq \underline{s_i}, \\ \dfrac{1}{2(m-k)} \displaystyle\sum_{i=k+1}^{m} d^u(t_i, s_i) & \text{otherwise.} \end{cases}$$

Example 4. As noted in Example 2,

$$\mathsf{Upd}(sal(Alice, 1000)) = \{sal(Alice, 1000), sal(x, 1000), sal(Alice, y)\},$$

which means that there are four options regarding the fact $salary(Alice, 1000)$: keeping it unchanged, changing the first argument (employee-name), changing the second argument (salary), or deleting it altogether. Assuming that employee-name is the primary key for the salary relation, according to d^2, the costs of these options are 0, 1, $\frac{1}{2}$ and 1, respectively. Note, also, that in this case, according to the repair context $\mathfrak{R} = \langle \models, d^2, \Sigma \rangle$, the two repairs of the database are:
$\{emp(Alice), emp(Bob), dir(Alice), sal(Alice, v_1), sal(Bob, 1000)\}$ for $v_1 > 1000$,
$\{emp(Alice), emp(Bob), dir(Alice), sal(Alice, 1000), sal(Bob, v_2)\}$ for $v_2 < 1000$.
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 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 9.

Example 5. Consider a database instance $\mathcal{D} = \{teaches(John, DB)\}$ 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. A repair in this case would be $\mathcal{R} = \{teaches(John, x_1), teaches(John, x_2)\}$ for some $x_1 \neq x_2 \neq DB$. Each one of the two optimal matchings in this case relates the database fact to one of the two elements of $\mathcal{R}$, leaving the other one unmatched. In the notations of the previous section, then, $d^1_\Sigma(\mathcal{D}, \mathcal{R}) = \frac{1}{2} + \frac{1}{4}$. If linking is used instead of matching, there is only one optimal linking between $\mathcal{D}$ and $\mathcal{R}$, which associates the two new facts in $\mathcal{R}$ with the old one in $\mathcal{D}$, hence in this case $d^1_\Sigma(\mathcal{D}, \mathcal{R}) = \frac{1}{4} + \frac{1}{4}$.

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 with respect to $\mathfrak{R} = \langle \models, d^u, \Sigma \rangle$. 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, 16, 19, 21, 22], (hyper-)graph computations [10, 11], and proof theoretic methods, such as analytic tableaux [7]. Tractability in such cases is usually reached only for restricted syntactical forms of the integrity constraints. For instance, the technique in [11] is polynomial only for denial integrity constraints[14], and the rewriting technique in [1], which is also tractable, is limited to binary universal constraints. Computational considerations regarding database repairs is beyond the scope of this paper, which is concentrated on the representational aspects of the problem. We note, however, that generally, the distance functions themselves do not add extra computational complexity to the problem. This is demonstrated, for instance, by the following results:

Proposition 7. [32] *Computing $d^u_\Sigma(A, B)$ is polynomial in the size of A and B.*

Proposition 8. *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.*

The main computational difficulty 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 [8–10] (see also [34] for complexity results regarding update-based repairing).

4 Integration of Constraint Data-Sources

Integration of autonomous data-sources under global integrity constraints (see [26]) 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.

Example 6. [26] 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 opinions of the owners 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

[14] That is, 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.

rent may be represented by the constraint $\mathcal{IC} = \{r \leftrightarrow \big((s \wedge t) \vee (s \wedge p) \vee (t \wedge p)\big)\}$. 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 6 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.

- Instead of database instances, which are sets of atomic facts, data sources are sets of *literals*. Denote by $\mathfrak{D}$ the set of these sources. So, instead of the set of atomic formulas, the following set is considered:
  $$\mathsf{Lit}(\mathfrak{D}, \mathcal{IC}) = \bigcup_{\mathcal{D}_i \in \mathfrak{D}} \mathsf{Atoms}(\mathcal{D}_i, \mathcal{IC}) \cup \bigcup_{\mathcal{D}_i \in \mathfrak{D}} \{\neg a \mid a \in \mathsf{Atoms}(\mathcal{D}_i, \mathcal{IC})\}.$$
  As before, an update $\mathcal{U}$ of $\mathfrak{D}$ is a consistent set[15] that consists of zero or more elements from $\mathsf{Upd}(L)$ for each $L \in \mathsf{Lit}(\mathfrak{D}, \mathcal{IC})$, and the set $\mathsf{Merge}(\mathfrak{D}, \mathcal{IC})$ of the potential merging of $\mathfrak{D}$ under $\mathcal{IC}$ consists of the updates that satisfy $\mathcal{IC}$.
- 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 10 and 11):
  - A *merging context* is a quadruple $\mathfrak{M} = \langle \models^{\mathcal{S}}, d, f, g \rangle$, where $\models^{\mathcal{S}}$ is the entailment relation induced by the underlying semantics $\mathcal{S}$, 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 $\mathfrak{M} = \langle \models^{\mathcal{S}}, d, f, g \rangle$, 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 *mergings* of the data-sources in $\mathfrak{D}$ under $\mathcal{IC}$, and with respect to the merging context $\mathfrak{M} = \langle \models^{\mathcal{S}}, d, f, g \rangle$, are the elements of the set
    $$\Delta_{\mathfrak{M}}(\mathcal{DB}) = \big\{\mathcal{M} \in \mathsf{Merge}(\mathfrak{D}, \mathcal{IC}) \mid \forall \mathcal{M}' \in \mathsf{Merge}(\mathfrak{D}, \mathcal{IC})\ d_{g,f}(\mathcal{M}, \mathfrak{D}) \leq d_{g,f}(\mathcal{M}', \mathfrak{D})\ \big\}.$$

Example 7. Consider again Example 6 and two contexts: $\mathfrak{M}_1 = \langle \models, d^u, \Sigma, \Sigma \rangle$, $\mathfrak{M}_2 = \langle \models, d^u, \Sigma, \max \rangle$. 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 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	$2\frac{1}{2}$	$9\frac{1}{2}$	$3\frac{1}{2}$

[15] I.e., without complementary literals.

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 of the facilities will be built, and so the rent will remain the same.[16] Thus, e.g., r is a consistent answer w.r.t. $\mathfrak{M}_1$ while $\neg r$ is a consistent answer w.r.t. $\mathfrak{M}_2$.

5 Conclusion

Data processing by distance considerations is not a new idea, and it has been used mainly in the context of query answering [1, 2] integration of constraint belief-sets [25, 26] and operators for belief revision [14, 27, 31]. 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. The new forms of repairs offered by our framework provide intuitive solutions to such cases, mainly as the notion of closeness can be captured in more subtle ways (most of them are domain dependent), and erroneous components of the data can be detected and updated without violating the valid fragment of the information.

References

1. M. Arenas, L. Bertossi, J. Chomicki. Consistent query answers in inconsistent databases. *Proc. PODS'99*, pp.68–79, 1999.
2. M. Arenas, L. Bertossi, J. Chomicki. Answer sets for consistent query answering in inconsistent databases. *Theory and Practice of Log. Prog.*, 3(4–5):393–424, 2003.
3. O. Arieli, M. Denecker, B. Van Nuffelen, M. Bruynooghe. Coherent integration of databases by abductive logic programming. *Artif. Intell. Res.*, 21:245–286, 2004.
4. O. Arieli, M. Denecker, B. Van Nuffelen, M. Bruynooghe. Computational methods for database repair by signed formulae. *Annals Math. Artif. Intell.*, 46:4–37, 2006.
5. L. Bertossi. Some research directions in consistent query answering: a vision. *Preproc. of EDBT'06 Workshop on Inconsistency in Databases*, pp.109–113, 2006.
6. L. Bertossi, J. Chomicki, A. Cortés, C. Gutierrez. Consistent answers from integrated data sources. *Proc. FQAS'2002*, LNCS 2522, pp.71–85, 2002.
7. L. Bertossi, C. Schwind. Database repairs and analytic tableau. *Annals of Mathematics and Artificial Intelligence*, 40(1–2):5–35, 2004.
8. A. Cali, D. Lembo, R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. *Proc. PODS'03*, 260–271, 2003.
9. J. Chomicki, J. Marchinkowski. A on the computational complexity of minimal-change integrity maintenance in relational databases. In L. Bertossi, A. Hunter, and T. Schaub, editors, *Inconsistency Tolerance*, LNCS 3300, pp.119–150. 2005.

[16] 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.

10. J. Chomicki, J. Marchinkowski. Minimal-change integrity maintenance using tuple deletion. *Journal of Information and Computation*, 197(1–2):90–121, 2005.
11. J. Chomicki, J. Marchinkowski, and S. Staworko. Computing consistent query answers using conflict hypergraphs. *Proc. CIKM'04*, pp.417–426, 2004.
12. M. Dalal. Investigations into a theory of knowledge base revision. *Proc. AAAI'98*, pp.475–479. AAAI Press, 1988.
13. S. de Amo, W. Carnielli, J. Marcos. A logical framework for integrating inconsistent information in multiple databases. *Proc. FoIKS'02*, LNCS 2284, pp.67–84, 2002.
14. J. Delgrande. Preliminary considerations on the modelling of belief change operators by metric spaces. *Proc. NMR'04*, pp.118–125, 2004.
15. J. Dieudonné, editor. *Foundations of Modern Analysis*. Academic Press, 1969.
16. T. Eiter. Data integration and answer set programming. *Proc. LPNMR'05*, LNCS 3662, pp.13–25. Springer, 2005.
17. T. Eiter, H. Mannila. Distance measure for point sets and their computation. *Acta Informatica*, 34:109–133, 1997.
18. B. Fazzinga, S. Flesca, F. Furfaro, F. Parisi. DART: a data acquisition and repairing tool. *EDBT'06 Workshop on Inconsistency in Databases*, pp.2–16, 2006.
19. E. Franconi, A. Palma, N. Leone, D. Perri, F. Scarcello. Census data repair: A challenging application of disjunctive logic programming. *Proc. LPAR'01*, LNCS 2250, pp.561–578. Springer, 2001.
20. N. Gelfond, V. Lifschitz. The stable model semantics for logic programming. *Proc. 5th Logic Programming Symposium*, pp.1070–1080. MIT Press, 1988.
21. G. Greco, S. Greco, E. Zumpano. A logic programming approach to the integration, repairing and querying of inconsistent databases. *Proc. ICLP'01*, LNCS 2237, pp.348–363. Springer, 2001.
22. S. Greco, E. Zumpano. Querying inconsistent databases. *Proc. LPAR'2000*, LNAI 1955, pp.308–325. Springer, 2000.
23. M. Kifer, E. L. Lozinskii. A logic for reasoning with inconsistency. *Journal of Automated Reasoning*, 9(2):179–215, 1992.
24. S. C. Kleene. *Introduction to Metamathematics*. Van Nostrand, 1950.
25. S. Konieczny, J. Lang, P. Marquis. Distance-based merging: A general framework and some complexity results. *Proc KR'02*, pp.97–108, 2002.
26. S. Konieczny, R. Pino Pérez. Merging information under constraints: a logical framework. *Journal of Logic and Computation*, 12(5):773–808, 2002.
27. D. Lehmann, M. Magidor, K. Schlechta. Distance semantics for belief revision. *Journal of Symbolic Logic*, 66(1):295–317, 2001.
28. N. Leone, T. Eiter, W. Faber, M. Fink, G. Gottlob, G. Greco. Boosting information integration: The INFOMIX system. *Proc. SEBD'05*, pp.55–66, 2005.
29. P. Liberatore, M. Schaerf. BReLS: A system for the integration of knowledge bases. *Proc. KR'2000*, pp.145–152. Morgan Kaufmann Publishers, 2000.
30. S.H. Nienhuys-Cheng. Distance between Herbrand interpretations: A measure for approximations to a target concept. *Proc. ILP'97*, LNCS 1297, pp.213–226, 1997.
31. P. Peppas, S. Chopra, N. Foo. Distance semantics for relevance-sensitive belief revision. *Proc. KR'04*, pp.319–328. AAAI Press, 2004.
32. J. Ramon, M. Bruynooghe. A polynomial time computable metric between point sets. *Acta Informatica*, 37(10):765–780, 2001.
33. R. Reiter. On closed world databases. In *Logic and Databases*, pages 55–76. 1978.
34. J. Wijsen. Database repairing using updates. *ACM Transactions on Database Systems*, 30(3):722–768, 2005.
35. M. Winslett. Reasoning about action using a possible models approach. *Proc. AAAI'98*, pp.89–93. AAAI press, 1988.