Proceedings of the 22nd International Conference on Principles of Knowledge Representation and Reasoning
Main Track

A Logic-Based Framework for Database Repairs

Nicolas Fröhlich[1] , **Arne Meier**[1] , **Nina Pardal**[2] , **Jonni Virtema**[3]
[1]Institut für Theoretische Informatik, Leibniz Universität Hannover, Germany
[2]Department of Computer Science, University of Huddersfield, UK
[3]School of Computer Science, University of Sheffield, UK.
{nicolas.froehlich, meier}@thi.uni-hannover.de, n.pardal@hud.ac.uk, j.t.virtema@sheffield.ac.uk

Abstract

We introduce a general abstract framework for database repairs, where the repair notions are defined using formal logic. We distinguish between integrity constraints and so-called query constraints. The former are used to model consistency and desirable properties of the data (such as functional dependencies and independencies), while the latter relate two database instances according to their answers to the query constraints. The framework allows for a distinction between hard and soft queries, allowing the answers to a core set of queries to be preserved, as well as defining a distance between instances based on query answers. We illustrate how different repair notions from the literature can be modelled in our framework. The framework generalises both set-based and cardinality based repairs to semiring annotated databases. Finally, we initiate a complexity-theoretic analysis of consistent query answering and checking existence of a repair in our setting.

1 Introduction

Inconsistency is a common phenomenon when dealing with large collections of data. In real-world applications, data is often made available from non-trustworthy sources resulting in very diverse quality of data and leading to problems related to the integrity of databases and repositories. *Database repairing*, one of the main approaches for dealing with inconsistency, focuses on frameworks that allow inconsistencies to be identified in order to then obtain a database that satisfies the constraints imposed. The usual approach is to search for a 'similar' database that satisfies the constraints. This new database is called a *repair*, and in order to define it properly one must determine the meaning of 'similar'. Repairs have been studied from different perspectives and several frameworks have been presented, including the introduction of preference criteria represented by weights (Staworko, Chomicki, and Marcinkowski 2012, Lukasiewicz, Malizia, and Molinaro 2023), as well as both hard and soft constraints (Carmeli et al. 2024). In some cases, such as when dealing with inconsistent knowledge bases (KBs), inconsistency-tolerant semantics are employed to extract meaningful answers from the facts in the KB without altering the underlying data (Bienvenu and Bourgaux 2020, Lukasiewicz et al. 2022). Another approach to deal with inconsistency is to use an inconsistency measure, which can be either a function that counts the number of integrity constraints violated, an abstract non-negative mapping to some partial order, or even a numerical measure based on an abstract repair semantics (Decker 2017, Parisi and Grant 2023, Livshits et al. 2021, Bertossi 2019). In this setting, a repair need not satisfy all specified integrity constraints but instead repairs are tolerant for a certain amount of inconsistency.

A database may admit multiple (a priori) incomparable repairs in the sense that it is not always clear what is the best way to repair an inconsistent database. *Consistent Query Answering* (CQA) aims to generalise the notion of cautious reasoning (or certain answers, in database parlance) in the presence of inconsistent knowledge. In the setting of CQA, a "valid" answer to a query is one that can be found in every possible repair. This problem has been analysed for different data models and different types of integrity constraints, under the most prominent repair semantics.

Data provenance provides means to describe the origins of data, allowing to give information about the witnesses to a query, or determining how a certain output is derived. Provenance semirings were introduced by Green et al. (2007) to devise a general framework that allows to uniformly treat extensions of positive relational algebra, where the tuples have annotations that reflect very diverse information. Some motivating examples of said relations come from incomplete and probabilistic databases, and bag semantics. The framework captures a notion of data provenance called *how-provenance*, where the semiring operations capture how each output is produced from the source. Subsequently semiring semantics for full first-order logic (FO) were developed by Grädel and Tannen (2017). The semantics refines the classical Boolean semantics by allowing formulae to be evaluated as values from a semiring. If K is a semiring, then a K-relation is a relation whose records are annotated with elements from K.

This paper considers repairs and consistent query answering for K-databases (i.e, relational databases whose tables are K-relations). Our general approach encompasses relational databases under set and bag semantics, as well as repair semantics over K-relations encoding some provenance data.

Our contribution. We present an abstract framework for defining database repairs that allows us to unify and simultaneously incorporate diverse notions that have been used in the literature to deal with inconsistency. We allow a distinction between integrity constraints classifying properties of data as either necessary or merely desirable to preserve in a repair. The latter are used in our framework to define a measure

for the inconsistency. The repair notions in our framework are expressed through the preservation of a given core set of query answers; together with integrity constraints, this yields the space for possible repairs. The distance between databases is computed using distances between the answers of specified queries in each instance. The technical definitions are presented in Section 3. We show examples of how well-known repair notions from literature can be expressed in our framework and exemplify the flexibility of the framework by generating novel repair notions.

Our framework enables us to simultaneously prove complexity results for a wide family of repair notions. We exemplify this in the simpler setting of set-based databases and obtain results for the complexity of the most important computational problems. As we initiate the complexity analysis of our framework, we focus on two widely adopted and well-understood constraint classes in data integration (LAV and GAV) as a natural starting point due to their simplicity and practical relevance. Although more expressive settings have been considered (e.g., Arming. et al. (2016)), they typically entail significantly higher complexity. Our results extend and generalise existing results on the data complexity of repairing, and pinpoint problems that are complete for the first and second level of the polynomial hierarchy (see Sec. 4.1).

Related work. Most of the previous research on repairs has been conducted within the data management community, while the problem of measuring inconsistency has been approached mostly by the knowledge representation community. Some works use a logic-based approach to explore reasoning under inconsistencies in knowledge bases and ontologies as a means to compute consistent answers and repairs (see, e.g., (Greco et al. 2003, Subrahmanian and Amgoud 2007, Zhang et al. 2017, Lukasiewicz et al. 2023)). Burdick et al. (2019) address the problem of repairing using entity linking, weights and consistent answers to determine the strength of the links, thus allowing the repair to be performed only on a given part of the schema. In the repair literature, a set of integrity constraints and a distance between instances is always presented. In some cases, the distance is represented by an inconsistency measure. However, while there are approaches to repairs that reflect the evaluation of important queries as criteria for determining preference, these usually focus on how to resolve conflicts and provide meaningful query answers despite the inconsistency (Calautti et al. 2022, Staworko et al. 2012, Yun et al. 2018). On the other hand, the idea of having a core set of query answers that need to be preserved is similar to Belief Revision (BR) concepts. The AGM theory for Belief Change, defined by Alchourron et al. (1985), represents *belief sets* as sets of formulae closed under a consequence operator, and a list of postulates describes how the revision operator incorporates new information, even if this information is inconsistent with what is believed. Guerra and Wassermann (2019) give a characterisation of model repair in terms of BR by introducing a new postulate that preserves the core of a belief set in the repair model. We are not aware of existing repair frameworks that simultaneously incorporate the impositions given by a set of integrity constraints and preserve the answers of a given set of queries, as well as allow the introduction of shades of inconsistency or truth as part of an inconsistency measure or minimality criteria.

Ten Cate et al. (2015) gave a systematic study of the data complexity of CQA for set-based repairs. The restrictions imposed to the integrity constraints arise from classes of tuple-generated dependencies, vital in data exchange and integration. Their results can be framed as results concerning particular instances of our framework; by refining the components of our framework, our results generalise theirs.

2 Preliminaries

We write $\vec{a}$ to denote a finite tuple $(a_1, \ldots, a_n)$ of elements from some set A. A *multiset* is a generalisation of a set that keeps track of the multiplicities of its elements. We write $\{\!\{\cdots\}\!\}$ to denote a multiset whose elements are written between the double curly brackets. E.g., the multiset $\{\!\{a, a, b\}\!\}$ has two copies of a and a single copy of b. The *support* $\mathrm{Supp}(A)$ of a multiset A is the underlying set of A. E.g., $\mathrm{Supp}(\{\!\{a, a, b\}\!\}) = \{a, b\}$. A set or a multiset A is finite if its cardinality $|A|$ is a natural number.

Definition 1. *A* semiring *is a tuple* $(K, +, \cdot, 0, 1)$*, where* $+$ *and* $\cdot$ *are binary operations on a set* K*,* $(K, +, 0)$ *is a commutative monoid with identity element* 0*,* $(K, \cdot, 1)$ *is a monoid with identity element* 1*, '*$\cdot$*' is distributive over '*$+$*', and* $x{\cdot}0 = 0 = 0{\cdot}x$ *for all* $x \in K$*. A semiring is* commutative *if* $(K, \cdot, 1)$ *is a commutative monoid.*

A semiring K has *divisors of* 0, if $ab = 0$ for some non-zero $a, b \in K$. It is $+$*-positive* if $a + b = 0$ implies that $a = b = 0$. It is *positive* if it is both $+$-positive and has no divisors of 0. E.g., the modulo two integer semiring $\mathbb{Z}_2$ is not positive since it is not $+$-positive (even though it has no divisors of 0), while $\mathbb{Z}_4$ has divisors of 0. Throughout the paper, we consider only partially ordered commutative semirings which are positive and have 0 as their minimum element (e.g., all naturally ordered positive semirings satisfy this). We write $<$ for the order relation. The *Boolean semiring* $(\mathbb{B}, \vee, \wedge, 0, 1)$ models logical truth or set-based data, and is formed from the two-element Boolean algebra. It is the simplest example of a semiring that is not a ring. The *semiring of natural numbers* $(\mathbb{N}, +, \cdot, 0, 1)$ consists of natural numbers with their usual operations and can be used, e.g., to model multisets of data. The *probability semiring* $(\mathbb{R}_{\geq 0}, +, \cdot, 0, 1)$ consists of the non-negative reals with standard addition and multiplication.

An *aggregate function* σ (for a semiring K) is a function that maps multisets of elements of K into an element of K. For instance, the sum and product of the elements in a multiset are aggregate functions.

Definition 2. *Let* K *be a semiring,* A *a set, and* $n \in \mathbb{N}$*. An* n-ary K-relation *is a function* $R\colon B \to K$*, where* $B \subseteq A^n$*. The* support *of* R *is* $\mathrm{Supp}(R) \coloneqq \{b \in B \mid R(b) \neq 0\}$*, and* B *is often identified with* A^n *via* $R(b) = 0$ *for* $b \in A^n \setminus B$*.*

Note that K-relations, for $K = \mathbb{B}$ or $K = \mathbb{N}$, are essentially sets and multisets, respectively.

2.1 Relational structures and databases

A *finite purely relational vocabulary* is a finite set $\{R_1, \ldots, R_n\}$ of relation symbols; each with a fixed *arity*

$\mathrm{ar}(R_i) \in \mathbb{N}$. We consider relational vocabularies τ extending finite purely relational vocabularies with a countable set of constant symbols c_i, for $i \in \mathbb{N}$. A *τ-interpretation* I maps each n-ary relation symbol to an n-ary relation $I(R_i) \subseteq A^n$ and each constant symbol c to some element $I(c) \in A$, over some *domain* set A. If instead, the interpretations of n-ary relation symbols are K-relations with support from A^n, we call I a *(τ, K)-interpretation*. A *finite τ-structure* (or finite *(τ, K)-structure*, resp.) $\mathfrak{A}$ consists of a finite domain set A and the interpretations of the symbols in τ, which in the case of (τ, K)-structures are K-relations. We write $R^{\mathfrak{A}}$ and $c^{\mathfrak{A}}$ to denote the interpretations $I(R)$ and $I(c)$ of R and c in $\mathfrak{A}$.

Formally, a *database schema* $\mathcal{S}$ extends a relational vocabulary τ with a finite set Att of attributes and countable sets of possible values $\mathrm{Ran}(x)$, for each $x \in \mathrm{Att}$. The *domain* of the schema $\mathrm{Dom}(\mathcal{S})$ consists of the union of $\mathrm{Ran}(x)$ for $x \in \mathrm{Att}$. Sometimes we write Dom instead of $\mathrm{Dom}(\mathcal{S})$. Furthermore, $\mathcal{S}$ fixes the type $\mathrm{Type}(R_i) \in \mathrm{Att}^n$ of each n-ary relation symbol R_i and the type $\mathrm{Type}(c) \in \mathrm{Att}$ of each constant symbol c. A *database* $\mathfrak{D}$ is obtained via an interpretation that maps each constant symbol $c \in \tau$ to an element $c^{\mathfrak{D}} \in \mathrm{Ran}(\mathrm{Type}(c))$ and each n-ary relation symbol $R \in \tau$ to a finite relation $R^{\mathfrak{D}} \subseteq \mathrm{Ran}(\mathrm{Type}(R))$, where $\mathrm{Ran}(x_1, \ldots, x_n)$ is defined as $\mathrm{Ran}(x_1) \times \cdots \times \mathrm{Ran}(x_n)$. The *active domain* $\mathrm{Adom}(\mathfrak{D})$ of a database $\mathfrak{D}$ is the smallest finite set such that $R^{\mathfrak{D}} \subseteq \mathrm{Adom}(\mathfrak{D})^{\mathrm{ar}(R)}$ and $c^{\mathfrak{D}} \in \mathrm{Adom}(\mathfrak{D})$ for each relation symbol R and constant symbol c of the schema. We denote the set of all databases over $\mathcal{S}$ by $\mathrm{DBS}(\mathcal{S})$ and write DBS when the context allows. For a semiring K and database schema $\mathcal{S}$, a *K-database* is obtained from a database of schema $\mathcal{S}$ by reinterpreting its relation symbols by K-relations of the same support. Instances $\mathfrak{D}$ of a database schema can naturally be interpreted as finite structures over the underlying schema vocabulary, extended by unary relation symbols R_x, for each $x \in \mathrm{Att}$, interpreted as $\mathrm{Ran}(x) \cap \mathrm{Adom}(\mathfrak{D})$. The domain of the finite structure corresponding to a database $\mathfrak{D}$ is $\mathrm{Adom}(\mathfrak{D})$. From now on, we identify finite structures with databases.

Let $\mathfrak{D}$ be a database (K-database, resp.) and R be a relation (K-relation, resp.) of the database. An atomic formula $R(\vec{a})$ is called a *fact*, if $\vec{a} \in R^{\mathfrak{D}}$ (resp., $\vec{a} \in \mathrm{Supp}(R^{\mathfrak{D}})$). Similarly, $\neg R(\vec{a})$ is a *negated fact*, if $\vec{a} \notin R^{\mathfrak{D}}$ ($\vec{a} \notin \mathrm{Supp}(R^{\mathfrak{D}})$, resp.). If $R(\vec{a})$ is a fact, then $\vec{a}$ is a *record* of R and $\mathfrak{D}$. In the set-based environment, we sometimes define databases by listing all the facts in the database tables. We use set comparison symbols and operations, e.g., *symmetric difference* $\oplus$ and $\subseteq$ on databases and sets. Table 1 serves as an illustration of an annotated database.

Table STOCK

ID	Product	Warehouse	#
112	potato	A	4
112	cabbage	A	6
113	carrot	B	7

Table BUILDINGS

Name	Address	#
A	5 Regent St.	1
C	2 Broad Ln.	1
D	14 Mappin St.	1

Table 1: An example of a product $\mathbb{N}$-database $\mathfrak{D}$ with quantities modelled via annotations in the semiring of natural numbers, where STOCK is a ternary and BUILDINGS a binary $\mathbb{N}$-relation. As this will serve as a running example, the duplicated IDs are intended.

2.2 Logics and query languages

We consider logics that are syntactic fragments and extensions of first-order logic and whose satisfaction relations are defined over databases or K-databases. More importantly, formulae of these logics have well-defined notions of *free* and *bound variables*. *Sentences* are formulae without free variables. Logic formulae are naturally interpreted as database queries; sentences are *Boolean queries* and formulae with k free variables are *k-ary queries*. We write Var for the set of first-order variables. If ϕ is a formula with free variables in $\vec{x}$, and $\vec{a}$ is a tuple of domain elements of the same length, we write $\phi(\vec{a}/\vec{x})$ to denote that the variables $\vec{x}$ are interpreted as $\vec{a}$. If $s\colon \mathrm{Var} \to \mathrm{Adom}(\mathfrak{D})$ is a variable assignment, $s[\vec{a}/\vec{x}]$ is the assignment that agrees otherwise with s but maps $\vec{x} \mapsto \vec{a}$.

The two main approaches to defining logics for K-databases are the logics for *meta-finite structures* of Grädel and Gurevitch (1998) and the use of semiring semantics as defined by Grädel and Tannen (2017). We adopt an approach close to the latter, where in addition to having K-relations, "truth" values of sentences are also elements of a semiring. K-databases can be seen as a special kind of model-defining K-interpretations of Grädel and Tannen.

Definition 3. *Let $(K, +, \cdot, 0, 1)$ be a semiring, $\mathfrak{D}$ be a K-database, and $s\colon \mathrm{Var} \to D$ be an assignment. The value $[\![\phi]\!]_{\mathfrak{D},s}$ of a formula $\phi \in \mathrm{FO}$ under s is defined as follows:*

$$[\![R(\vec{x})]\!]_{\mathfrak{D},s} = R^{\mathfrak{D}}(s(\vec{x})) \qquad [\![\phi \wedge \psi]\!]_{\mathfrak{D},s} = [\![\phi]\!]_{\mathfrak{D},s} \cdot [\![\psi]\!]_{\mathfrak{D},s}$$

$$[\![\neg\phi]\!]_{\mathfrak{D},s} = [\![\mathrm{nnf}(\neg\phi)]\!]_{\mathfrak{D},s} \quad [\![\phi \vee \psi]\!]_{\mathfrak{D},s} = [\![\phi]\!]_{\mathfrak{D},s} + [\![\psi]\!]_{\mathfrak{D},s}$$

$$[\![\forall x\phi]\!]_{\mathfrak{D},s} = \prod_{a \in \mathrm{Adom}(\mathfrak{D})} [\![\phi]\!]_{\mathfrak{D},s[a/x]} \quad [\![\exists x\phi]\!]_{\mathfrak{D},s} = \sum_{a \in \mathrm{Adom}(\mathfrak{D})} [\![\phi]\!]_{\mathfrak{D},s[a/x]}$$

$$[\![\neg R(\vec{x})]\!]_{\mathfrak{D},s} = 1, \text{ if } [\![R(\vec{x})]\!]_{\mathfrak{D},s} = 0, \text{ and } 0 \text{ otherwise,}$$

$$[\![x \sim y]\!]_{\mathfrak{D},s} = 1, \text{ if } s(x) \sim s(y), \text{ and } 0 \text{ otherwise,}$$

where $\sim \in \{=, \neq\}$ and $\mathrm{nnf}(\neg\phi)$ is the formula obtained from $\neg\phi$ by pushing all the negations to atomic level using the usual dualities. We use the standard shorthands $\phi \to \psi$ and $\phi \leftrightarrow \psi$. For sentences ϕ, we write $[\![\phi]\!]_{\mathfrak{D}}$ as a shorthand for $[\![\phi]\!]_{\mathfrak{D},s_\emptyset}$, where $s_\emptyset$ is the empty assignment. We write $\mathfrak{D}, s \models \phi$ ($\mathfrak{D} \models \phi$, resp.), if $[\![\phi]\!]_{\mathfrak{D},s} \neq 0$ ($[\![\phi]\!]_{\mathfrak{D}} \neq 0$, resp.). We define $[\![\phi]\!]^B_{\mathfrak{D},s} = 1$, if $\mathfrak{D}, s \models \phi$, and $[\![\phi]\!]^B_{\mathfrak{D},s} = 0$ otherwise.

If ϕ is a query, then a *query answer* over a database $\mathfrak{D}$ is a tuple of elements $\vec{a}$ from $\mathrm{Adom}(\mathfrak{D})$ such that $\mathfrak{D} \models \phi(\vec{a}/\vec{x})$. We write $\mathrm{Ans}(\mathfrak{D}, \phi) := \{\vec{a} \in \mathrm{Adom}(\mathfrak{D})^n \mid \mathfrak{D} \models \phi(\vec{a}/\vec{x})\}$ to denote the set of answers to the query ϕ in the database $\mathfrak{D}$. We set $w_{\mathfrak{D},\phi}(\vec{a}) := [\![\phi(\vec{a}/\vec{x})]\!]_{\mathfrak{D}}$ to indicate the annotated answers to the query in the database.

2.3 Repairs and consistent query answering

Integrity constraints (ICs) are sentences in some logic that describe the necessary properties the data should comply. Let $\mathcal{S}$ be a database schema. Given a set of ICs $\mathcal{C}$, we say that $\mathfrak{D} \in \mathrm{DBS}(\mathcal{S})$ is *consistent (w.r.t. $\mathcal{C}$)* if $\mathfrak{D}$ satisfies $\mathcal{C}$, that is, $\mathfrak{D} \models \phi$ for every $\phi \in \mathcal{C}$. Otherwise, $\mathfrak{D}$ is *inconsistent*. A set of ICs is *consistent* if there exists a database instance

$\mathfrak{D}$ that makes the ICs true. Given a distance d between instances and $\mathfrak{D}_1, \mathfrak{D}_2, \mathfrak{D}_3 \in \mathrm{DBS}$, we write $\mathfrak{D}_2 \leq_{d,\mathfrak{D}_1} \mathfrak{D}_3$ if $d(\mathfrak{D}_1, \mathfrak{D}_2) \leq d(\mathfrak{D}_1, \mathfrak{D}_3)$. A database $\mathfrak{D}_2$ is a *repair* of $\mathfrak{D}_1$, if $\mathfrak{D}_2$ is $\leq_{d,\mathfrak{D}_1}$-minimal in the set $\{\mathfrak{D} \in \mathrm{DBS}(\mathcal{S}) \mid \mathfrak{D} \models \mathcal{C}\}$.

Set-based repairs are one of the most prominent types of repairs. The goal is to find a consistent database instance, with the same schema as the original one, that satisfies the repair semantics (e.g., *subset*, *superset* and *symmetric difference*) and differs from the original by a minimal set of records under set inclusion. Given a set of ICs $\mathcal{C}$, the instance $\mathfrak{D}'$ is a *symmetric difference repair* of $\mathfrak{D}$ w.r.t. $\mathcal{C}$ if $\mathfrak{D}' \models \mathcal{C}$ and there is no instance $\mathfrak{D}''$ such that $\mathfrak{D}'' \models \mathcal{C}$ and $\mathfrak{D} \oplus \mathfrak{D}'' \subsetneq \mathfrak{D} \oplus \mathfrak{D}'$. The instance $\mathfrak{D}'$ is a *subset repair* of $\mathfrak{D}$ w.r.t. $\mathcal{C}$ if $\mathfrak{D}' \subseteq \mathfrak{D}$ and $\mathfrak{D}'$ is a symmetric difference repair of $\mathfrak{D}$. Superset repair is defined analogously. Cardinality-based repairs as defined in (Lopatenko and Bertossi 2007) aim to find repairs of the original database that minimise the cardinality of the symmetric difference between instances.

Related to the problem of repairing, *consistent query answering* (CQA) proposes that a valid query answer is one that holds across all possible database repairs. More formally, given a database $\mathfrak{D}$, a set of ICs $\mathcal{C}$, and a query $q(\vec{x})$, the consistent answers to q w.r.t. $\mathfrak{D}$ and $\mathcal{C}$, denoted by $CQA(q, \mathfrak{D}, \mathcal{C})$, is the set of tuples $\vec{a}$ such that $\vec{a} \in \mathrm{Ans}(\mathfrak{D}', q)$ for each repair $\mathfrak{D}'$ of $\mathfrak{D}$ w.r.t. $\mathcal{C}$. For Boolean queries q computing the consistent answers of q is equivalent to determining if $\mathfrak{D}' \models q$ for every repair $\mathfrak{D}'$. Hence, $CQA(q, \mathfrak{D}, \mathcal{C})$ contains the empty tuple if $\mathfrak{D}' \models q$ for all repairs $\mathfrak{D}'$ of $\mathfrak{D}$. CQA has been studied across various data models, repair semantics, and IC types, especially where repairs are well-explored.

Repairs are instances that eliminate all inconsistency of the given database. This notion can be relaxed to simply look for a "reduction" of the inconsistency to some acceptable level, giving way to the notion of *inconsistency-tolerant repairs*. In this context, it suffices to find a *nearly* consistent instance without adding constraint violations. To this end, one formalises measuring inconsistency via *inconsistency measures*. These are typically $\mathbb{R}$-valued functions that meet certain postulates that vary with chosen semantics. This concept has been explored for both relational and graph databases. Inconsistency can be measured using a general definition taking values in an ordered positive semiring, not just $\mathbb{R}_{\geq 0}$. Inconsistency measures should meet certain postulates, including the *consistency postulate* "$\mathfrak{D} \models \mathcal{C}$ implies $\mathcal{M}(\mathcal{C}, \mathfrak{D}) = 0$".

Definition 4. *Given a set of ICs $\mathcal{C}$ and an instance $\mathfrak{D}$, an* inconsistency measure (IM) *$\mathcal{M}$ is a function that maps $(\mathcal{C}, \mathfrak{D})$ to a value in a semiring K.*

3 Unified Framework for Repairs

Database repairing always incurs a cost, reflected by the number of record changes needed for consistency. Some changes may be costlier, and some facts may need to remain unchanged. Sometimes, tolerating inconsistency is preferable to the repair cost, thus requiring various repair approaches. Inconsistency measures, usually based on database ICs, can vary in flexibility. We divide ICs into those that *must* be satisfied (*hard-constraints* $\mathcal{C}_h$), and those for which a degree of inconsistency is allowed (*soft-constraints* $\mathcal{C}_s$). Hard-constraints represent those properties usually referred to as "integrity constraints". We suggest a fine-grained repair framework utilising both hard and soft constraints, and including hard-queries ($\mathcal{Q}_h$) for further constraining the repair space and soft-queries ($\mathcal{Q}_s$) for defining the minimality criteria.

Hard-Queries ($\mathcal{Q}_h = (\mathcal{Q}_h^+, \mathcal{Q}_h^-)$). These yield a core set of answers that we want to preserve in a repair, both in the positive and negative sense. For any Boolean query $\phi \in \mathcal{Q}_h^+$, if $\mathfrak{D}'$ is a repair of $\mathfrak{D}$, we want $\mathfrak{D} \models \phi$ to imply $\mathfrak{D}' \models \phi$. For non-Boolean queries, we want the answers to a hard-query in $\mathfrak{D}$ to be also retrieved in $\mathfrak{D}'$. Formally (in the setting of Boolean annotations), we require $Ans(\mathfrak{D}, \phi) \subseteq Ans(\mathfrak{D}', \phi)$. If the annotations are non-Boolean, the above notion is generalised to reflect the annotations. Moreover, we require that for all Boolean queries $\psi \in \mathcal{Q}_h^-$, $\mathfrak{D} \not\models \psi$ implies $\mathfrak{D}' \not\models \psi$, and for non-Boolean queries, we want that all answers to a hard-query in $\mathfrak{D}'$ are already answers in $\mathfrak{D}$. Formally, for every $\phi \in \mathcal{Q}_h^+$, $\psi \in \mathcal{Q}_h^-$ and $\vec{a} \in \mathrm{Adom}(\mathfrak{D})^n$, we require that $w_{\mathfrak{D},\phi}(\vec{a}) \leq w_{\mathfrak{D}',\phi}(\vec{a})$ and $w_{\mathfrak{D},\psi}(\vec{a}) \geq w_{\mathfrak{D}',\psi}(\vec{a})$. E.g., if annotations reflect multiplicities in a multiset, the corresponding multiplicities for answers of $\mathcal{Q}_h^+$ (resp. $\mathcal{Q}_h^-$) can only increase (decrease) in repairs.

Soft-Queries ($\mathcal{Q}_s$). These reflect answers which are deemed important, but not necessary to maintain. Given a database $\mathfrak{D}$ they define a partial order $\leq_{\mathcal{Q}_s,\mathfrak{D}}$ between database instances reflecting how close the instances are from $\mathfrak{D}$ with respect to the answers to queries in $\mathcal{Q}_s$. To simplify the presentation, we define the notion for Boolean queries, but will later extend this to formulae as well. Let $\mathcal{Q}_s = \{\varphi_1, \ldots, \varphi_n\}$ be the set of soft-queries, and let $\mathfrak{D}$ be a database. We define $\mathcal{Q}_s[\mathfrak{D}] := (\llbracket\varphi_1\rrbracket_{\mathfrak{D}}, \ldots, \llbracket\varphi_n\rrbracket_{\mathfrak{D}})$. From the order of the semiring K, we obtain the canonical partial order $\leq_K^n$ for K^n, which gives rise to a partial order $\leq_{\mathcal{Q}_s,\mathfrak{D}}$ between databases by setting $\mathfrak{D}' \leq_{\mathcal{Q}_s,\mathfrak{D}} \mathfrak{D}''$ if $\mathcal{Q}_s[\mathfrak{D}] \leq_K^n \mathcal{Q}_s[\mathfrak{D}'] \leq_K^n \mathcal{Q}_s[\mathfrak{D}'']$. This notion resembles standard set-based repairs.

As an alternative, we define a notion similar to cardinality based repairs. Let $\Delta\colon K^n \times K^n \to K^n$ be a function that intuitively computes a distance between tuples of semiring values, and let σ be an n-ary aggregate function taking values in K. Define a distance between instances in terms of $\mathcal{Q}_s$ by setting $d_{\mathcal{Q}_s}(\mathfrak{D}, \mathfrak{D}') := \sigma\big(\Delta(\mathcal{Q}_s[\mathfrak{D}], \mathcal{Q}_s[\mathfrak{D}'])\big)$.

Soft-Constraints ($\mathcal{C}_s$). Modelled using sentences, these are used to define an inconsistency measure $\mathcal{M}$ that allows us to obtain degrees of tolerance for the repair. Let $f_{sc}\colon \mathrm{DBS} \times \mathcal{L} \to K$ be a function, and let σ be an aggregate function taking values in K. The value $f_{sc}(\mathfrak{D}, \phi)$ indicates how inconsistent $\mathfrak{D}$ is according to ϕ (e.g., how far the property defined by ϕ is of being true in $\mathfrak{D}$) and σ then aggregates the levels of inconsistency. We stipulate that $f_{sc}(\mathfrak{D}, \phi) = 0$ iff $\mathfrak{D} \models \phi$. The inconsistency measure is defined as $\mathcal{M}(\mathfrak{D}, \mathcal{C}_s) := \sigma\{\!\{f_{sc}(\mathfrak{D}, \phi) \mid \phi \in \mathcal{C}_s\}\!\}$.

Next we exemplify the use of soft constraints with a σ that is the aggregate sum defined by the $+$ of the semiring of natural numbers and with two examples of the function f_{sc}.

Example 5. *Consider $\mathfrak{D}$ in Table 1 and the soft-constraint that warehouses A and B may contain at most one type of product. This is expressed as $\mathcal{C}_s := \{\phi_A, \phi_B\}$, where $\phi_c = \forall x \forall y \forall x' \forall y' \, (\mathtt{STOCK}(x, y, c) \wedge \mathtt{STOCK}(x', y', c) \rightarrow y = y')$,*

for $c \in \{\text{A}, \text{B}\}$. In the first example below, an inconsistency measure is defined using the Boolean truth value of a negated formula $\neg\phi$ as a measure of the inconsistency of $\mathfrak{D}$:

$$\mathcal{M}(\mathfrak{D}, \mathcal{C}_s) := \sum \{\!\{ [\![\neg\phi]\!]^B_{\mathfrak{D}} \mid \phi \in \mathcal{C}_s \}\!\} = 1 + 0 = 1.$$

In the second, we directly use the semiring value $[\![\neg\phi]\!]_{\mathfrak{D}}$:

$$\mathcal{M}(\mathfrak{D}_1, \mathcal{C}_s) := \sum \{\!\{ [\![\exists x \exists y \exists x' \exists y' \, (\texttt{STOCK}(x, y, c) \wedge \texttt{STOCK}(x', y', c) \wedge y \neq y')]\!]_{\mathfrak{D}} \mid c \in \{\text{A}, \text{B}\} \}\!\} = 48 + 0 = 48.$$

We are now ready to formally define our repair framework that incorporates the concepts discussed.

Definition 6 (repair framework)**.** *Fix a database schema $\mathcal{S}$. For each part of the framework, we may use a different logic $\mathcal{L}$ over the schema. Fix an inconsistency measure $\mathcal{M}$ defined in terms of $\mathcal{L}$-sentences as described above. A* repair framework *$\mathcal{R} = (\mathcal{C}_h, \mathcal{C}_s, \mathcal{Q}_h, \mathcal{Q}_s, \mathcal{M}, m_{\mathcal{Q}_s})$ is a tuple, where $\mathcal{C}_h$ and $\mathcal{C}_s$ are finite sets of $\mathcal{L}$-sentences representing* hard *and* soft-constraints*, $\mathcal{Q}_h = (\mathcal{Q}_h^+, \mathcal{Q}_h^-)$ and $\mathcal{Q}_s$ are finite sets of $\mathcal{L}$-formulae representing* hard *and* soft-queries*, and $m_{\mathcal{Q}_s} \in \{d_{\mathcal{Q}_s}, \leq_{\mathcal{Q}_s, \mathfrak{D}}\}$ is a method to compare databases as described above. We require $\mathcal{C}_h$ to be consistent.*

Given a database $\mathfrak{D}$ and a repair framework, we define the following relativised sets of $\mathcal{L}$-sentences: $\mathcal{Q}_s^{\mathfrak{D}} := \{\varphi(\vec{a}/\vec{x}) \mid \varphi(\vec{x}) \in \mathcal{Q}_s, \vec{a} \subseteq \text{Adom}(\mathfrak{D}) \text{ of suitable type}\}$, and $\mathcal{Q}_h^{\star,\mathfrak{D}} := \{\varphi(\vec{a}/\vec{x}) \mid \varphi(\vec{x}) \in \mathcal{Q}_h^{\star}, \vec{a} \subseteq \text{Adom}(\mathfrak{D}) \text{ of suitable type}\}$ for $\star \in \{+, -\}$. This shift from sets of formulae to sets of sentences is not crucial but does make the presentation of the following definition slightly lighter. Note that if $\vec{b}$ contains an element that is not in the active domain of $\mathfrak{D}$ then the interpretations of atoms $R(\vec{b})$ (and their negations) are computed as if the elements belonged to the active domain of $\mathfrak{D}$. That is,

$$\left[\!\left[R(\vec{b})\right]\!\right]_{\mathfrak{D}} = \left[\!\left[R(\vec{b})\right]\!\right]^B_{\mathfrak{D}} = 0, \; \left[\!\left[\neg R(\vec{b})\right]\!\right]_{\mathfrak{D}} = \left[\!\left[\neg R(\vec{b})\right]\!\right]^B_{\mathfrak{D}} = 1.$$

In the following definition (item 1.), we choose to limit the active domains of repairs. With this restriction, it is possible to use more expressive logics in the different parts of the framework without increasing the computational complexity too much; see Section 4.3 for our upper bounds. Note that any inconsistent database instance can be provided with sets of fresh data values, which can then be used as fresh data values for the repairs without affecting our complexity results.

Definition 7 ($\mathcal{R}$-repair)**.** *Given a repair framework $\mathcal{R} = (\mathcal{C}_h, \mathcal{C}_s, \mathcal{Q}_h, \mathcal{Q}_s, \mathcal{M}, m_{\mathcal{Q}_s})$, a K-database $\mathfrak{D}$, and a threshold $\epsilon \in K$ s.t. $\varepsilon \geq 0$, we say that $\mathfrak{D}'$ is an ε-$\mathcal{R}$-*repair of $\mathfrak{D}$ *if the following six items are fulfilled:*

(1) $\text{Adom}(\mathfrak{D}') \subseteq \text{Adom}(\mathfrak{D})$,
(2) $\mathfrak{D}' \models \mathcal{C}_h$,
(3) $[\![\phi]\!]_{\mathfrak{D}} \leq [\![\phi]\!]_{\mathfrak{D}'}$, *for all* $\phi \in \mathcal{Q}_h^+$,
(4) $[\![\psi]\!]_{\mathfrak{D}} \geq [\![\psi]\!]_{\mathfrak{D}'}$, *for all* $\psi \in \mathcal{Q}_h^-$,
(5) $\mathcal{M}(\mathfrak{D}', \mathcal{C}_s) \leq \varepsilon$,
(6) $\mathfrak{D}'$ *is minimal with respect to* $\leq_{\mathcal{Q}_s, \mathfrak{D}}$, *if* $m_{\mathcal{Q}_s} = \leq_{\mathcal{Q}_s, \mathfrak{D}}$, *and* $d_{\mathcal{Q}_s}(\mathfrak{D}, \mathfrak{D}')$ *is minimised, if* $m_{\mathcal{Q}_s} = d_{\mathcal{Q}_s}$.

We say that $\mathfrak{D}'$ is an annotation unaware repair *if in* (3) *and* (4) *all refereces to $[\![-]\!]_{\mathfrak{D}}$ are replaced with $[\![-]\!]^B_{\mathfrak{D}}$. Notice that, $[\![\phi]\!]^B_{\mathfrak{D}} \leq [\![\phi]\!]^B_{\mathfrak{D}'}$ if and only if $[\![\neg\phi]\!]^B_{\mathfrak{D}'} \leq [\![\neg\phi]\!]^B_{\mathfrak{D}}$. Hence, in the annotation unaware case, we may omit* (4) *and write $\mathcal{Q}_h$ for $\mathcal{Q}_h^+$. We drop ε from ε-$\mathcal{R}$-*repair*, if $\varepsilon = 0$ or $\mathcal{C}_s = \emptyset$.*

Note that, if $\varepsilon > 0$, the above notions are meaningful, even if $\mathcal{C}_s$ is inconsistent (recall that we consider only partially ordered positive semirings whose minimum element is 0). The framework facilitates the creation of diverse repair notions. For instance, by using (3) and (4) it is straightforward to specify a repair notion which is a subset repair with respect to some relation R (put $R(\vec{x})$ in $\mathcal{Q}_h^-$), a superset repair with respect to some other relation S (put $S(\vec{x})$ in $\mathcal{Q}_h^+$), and where the interpretation of a third relation T must remain unchanged (put $T(\vec{x})$ in both $\mathcal{Q}_h^+$ and $\mathcal{Q}_h^-$). Moreover, putting $\neg R(\vec{x})$ in $\mathcal{Q}_h^+$ and $\neg S(\vec{x})$ in $\mathcal{Q}_h^-$ leads to a repair notion that allows annotations to be changed freely as long as, with respect to supports of relations, the repair notion is a subset repair with respect to R and a superset repair with respect S. The notions of minimality facilitated by (6) are also diverse. The repair notions obtained by using $\leq_{\mathcal{Q}_s, \mathfrak{D}}$ resemble standard set-based repairs, while notions given by $d_{\mathcal{Q}_s}$ are similar to cardinality based repairs. This is due to $d_{\mathcal{Q}_s}$ being in a sense 1-dimensional, as it aggregates distances between interpretations of formulae in $\mathcal{Q}_s$ into a single semiring value, which is then mimimised.

If $\mathcal{C}_s = \emptyset$ or $\epsilon = 0$, Definition 7 yields a classical definition of a repair, where the desired minimality and repair criteria is defined through $\mathcal{Q}_h$ and $\mathcal{Q}_s$. If instead $0 < \epsilon \leq \mathcal{M}(\mathcal{C}_s, \mathfrak{D})$, it resembles the definition of inconsistency-tolerant repair given in (Decker 2017). The following example shows how the standard superset and subset repairs, and their cardinality based variants, are implemented in our framework.

Example 8. *Consider the database in Tab. 1 restricting attention to the* STOCK *table, and notice that the hard constraint $\mathcal{C}_h := \{$"ID is a key"$\}$ is violated. Setting $\mathcal{Q}_h^- := \{\texttt{STOCK}(x, y, z)\}$ as negative hard queries, yields that no tuples can be added to* STOCK *nor any annotations can be increased in the repairs. Setting $\mathcal{Q}_s := \{\texttt{STOCK}(x, y, z)\}$ as soft queries and using the partial order $\leq_{\mathcal{Q}_s, \mathfrak{D}}$ as the minimality notion, together with the hard queries, yields the standard subset repair notion (in bag semantics). In this case, the database in Tab. 1 has two repairs, obtained by removing one of the records of* STOCK *with* ID 112.

Using the same soft queries $\mathcal{Q}_s := \{\texttt{STOCK}(x, y, z)\}$, we can instead define a distance between instances using the modulus $|a - b|$ and the aggregate sum of the natural numbers as $d_{\mathcal{Q}_s}(\mathfrak{D}, \mathfrak{D}') := \sum_{\phi \in \mathcal{Q}_s} |[\![\phi]\!]_{\mathfrak{D}} - [\![\phi]\!]_{\mathfrak{D}'}|$. In this case, the database in Tab. 1 has only one repair. The instance that complies with the key constraint and the hard queries, and minimises this distance with respect to $\mathfrak{D}$ is the one that keeps tuples STOCK(112, cabbage, A) *and* STOCK(113, carrot, B)*. Considering $\mathcal{Q}_h^+ := \{\exists x \exists y \exists z \, \texttt{STOCK}(x, y, z)\}$ as the hard queries for an annotation-aware repair, yields that repairs maintain at least the same quantity of product units as $\mathfrak{D}$, or more. Note that this restriction allows annotations to change and does not prevent tuples from being deleted or added.*

Example 9. *Let $\mathfrak{D}$ be the two-table database of Table 1 and $\mathcal{R} = (\mathcal{C}_h, \mathcal{C}_s, \mathcal{Q}_h, \mathcal{Q}_s, \mathcal{M}, m_{\mathcal{Q}_s})$ be a repair framework, where $\mathcal{C}_h$:= {"Warehouse is a foreign key"} are the hard-constraints, the soft-constraints are $\mathcal{C}_s := \{\forall x\, \neg\texttt{STOCK}(x, \text{cabbage}, \text{A}), \forall x\, \neg\texttt{STOCK}(x, \text{potato}, \text{B})\}$, the positive and negative hard-queries are defined as $\mathcal{Q}_h^+ := \{\neg\texttt{STOCK}(x,y,z), \texttt{BUILDINGS}(u,v)\}$ and $\mathcal{Q}_h^- := \{\texttt{STOCK}(x,y,z)\}$, and the soft-queries are set $\mathcal{Q}_s := \{\texttt{STOCK}(x,y,z), \texttt{BUILDINGS}(u,v)\}$. Let $\mathcal{M}$ be the annotation-aware inconsistency measure from Example 5, $m_{\mathcal{Q}_s}$ be $\leq_{\mathcal{Q}_s,\mathfrak{D}}$, and $\epsilon := 5$.*

Now, the hard-queries imply that any repair $\mathfrak{D}'$ of $\mathfrak{D}$ must be such that no deletions to BUILDINGS *nor additions to* STOCK *have been done and the support of the table* STOCK *has remained unchanged. The soft-queries together with $\leq_{\mathcal{Q}_s,\mathfrak{D}}$ imply that the repair should coordinate-wise minimise the changes made to the tables. The soft-constraints imply that not too many cabbages (potatos, resp.) are stored in warehouse* A *(*B*, resp.). Hence, the only ϵ-$\mathcal{R}$-repairs of $\mathfrak{D}$ are databases that insert a tuple (B, x) with annotation 1 to* BUILDINGS *for some address x and change the annotation of the second record of* STOCK *in $\mathfrak{D}$ from 6 to 5.*

Some properties are expected when dealing with repairs, for example, a consistent $\mathfrak{D}$ should not need to be repaired. Indeed, if $\mathfrak{D}$ satisfies $\mathcal{C}_h$ and $\mathcal{C}_s$, then in particular $\mathcal{M}(\mathcal{C}_s, \mathfrak{D}) = 0$ and it follows from the $\leq_{d_{\mathcal{Q}_s},\mathfrak{D}}$-minimality of $\mathfrak{D}$ (minimality of $d_{\mathcal{Q}_s}(\mathfrak{D}, \mathfrak{D})$, resp.) that $\mathfrak{D}$ is a repair. It is often desirable that any $\mathfrak{D}$ can always be repaired. However, our framework can express both ICs and critical properties of databases that should be kept unchanged (expressed using $\mathcal{Q}_h$). Assuming that the set of ICs is consistent, there will always be some $\mathfrak{D}'$ which satisfies the ICs and thus $\mathfrak{D}' \models \mathcal{C}_h$ and $\mathcal{M}(\mathcal{C}_s, \mathfrak{D}') = 0$. However, this is not sufficient to ensure that $Ans(\mathfrak{D}, \mathcal{Q}_h^+) \subseteq Ans(\mathfrak{D}', \mathcal{Q}_h^+)$ and $Ans(\mathfrak{D}, \mathcal{Q}_h^-) \supseteq Ans(\mathfrak{D}', \mathcal{Q}_h^-)$. Furthermore, if we want to find an inconsistency-tolerant repair and only assume $\mathcal{C}_h$ to be consistent, this is not sufficient to ensure that $\mathcal{M}(\mathcal{C}_s, \mathfrak{D}') \leq \epsilon$ for an instance $\mathfrak{D}'$ that satisfies $\mathcal{C}_h$. Hence, deciding the *existence of a repair* results a meaningful problem in our framework.

We conclude with worked out examples illustrating how our repair framework can generate new repair notions and exemplify its flexibility.

Example 10. *Consider a database schema with two relations indicating teacher allocations in a department; $T(x, y)$ and $C(z)$ indicate that a lecturer x is assigned to course y, and that z is a course. Consider an annotation unaware repair framework $(\mathcal{C}_h, \mathcal{C}_s, \mathcal{Q}_h, \mathcal{Q}_s, \mathcal{M}, d_{\mathcal{Q}_s})$, where $\mathcal{C}_s := \{\}$,*

$$\mathcal{C}_h := \{\forall y\,(C(y) \to \exists x T(x,y)), \neg T(t_4, c), \neg C(a)\},$$
$$\mathcal{Q}_h^+ := \{T(t_1,c), T(t_2,d), C(b), C(c), C(d)\}, \quad \mathcal{Q}_h^- := \emptyset,$$
$$\mathcal{Q}_s := \{\phi_i = \forall x \exists^{\leq i} y\,\big(C(y) \wedge T(x,y)\big) \mid 1 \leq i \leq 10\},$$

where $t_1, \ldots, t_4, a, b, c$, and d are constants. The sentence ϕ_i (written using counting quantifiers as the usual shorthand) expresses that every teacher is assigned to at most i courses. From $\mathcal{Q}_s$, we define a distance between instances to describe a minimality concept that is neither set-based nor cardinality-based (below $\Delta(x_1, x_2) := |x_1 - x_2|$):

$$d_{\mathcal{Q}_s}(\mathfrak{D}, \mathfrak{D}') := \sum_{\phi_i \in \mathcal{Q}_s} \Delta\Big([\![\phi_i]\!]^B_{\mathfrak{D}}, [\![\phi_i]\!]^B_{\mathfrak{D}'}\Big).$$

The distance prioritises instances that have a similar maximum allocation per teacher. Consider an instance $\mathfrak{D} := \{T(t_1,c), T(t_2,d), T(t_1,b), T(t_4,c), C(a), C(b), C(c), C(d), C(e)\}$. Here, we have $[\![\phi_1]\!]^B_{\mathfrak{D}} = 0$ and $[\![\phi_i]\!]^B_{\mathfrak{D}} = 1$ for every $i \geq 2$. Now, $\mathfrak{D}' := \{T(t_1,c), T(t_2,d), T(t_1,b), T(t_4,e), C(b), C(c), C(d), C(e)\}$ is a repair satisfying that every course in $C(y)$ has at least one teacher assigned, the facts $T(t_4, c)$ and $C(a)$ are no longer in the database, and prioritises the criteria that maximum allocation per teacher is as similar as possible to $\mathfrak{D}$. The framework allows us to express prioritised repairs in terms of formulae, which differs from the approach of using predefined priority criteria studied in the repair literature.

Next, we give two examples of a repair framework for a graph databases. For simplicity, we consider simple digraphs.

Example 11. *Consider a simple directed labelled graph (V^G, E^G, T^G) representing a railway network; nodes are stations and edges are direct rail links. The predicate $T(x)$ indicates that there is a taxi rank at station x. We set $\mathcal{C}_h :=$ $\{\forall x \exists y\, E(x,y)\}$, $\mathcal{C}_s := \{\forall x \forall y\,(E(x,y) \to T(y))\}$,*

$$\mathcal{Q}_h^+ := \{E(\textit{London}, \textit{Cardiff}), E(\textit{Cardiff}, \textit{London})$$
$$E(\textit{London}, \textit{Edinburgh}), E(\textit{Edinburgh}, \textit{London})\} \cup \{\neg T(x)\},$$

and $\mathcal{Q}_s := \{E(x,y)\}$. The distance used to determine a repair minimises the cardinality of the symmetric difference of $\{E(a,b) \mid G \models E(a,b), a, b \in V^G\}$ and $\{E(a,b) \mid H \models E(a,b), a, b \in V^H\}$. We use the same inconsistency measure as in Example 5, with the semiring value of the formula $\exists x \exists y (E(x,y) \wedge \neg T(y))$, and set $\epsilon := 5$. That is, the measure counts the number of violations to the rule that stipulates "if there is a direct connection between x and y, then there is a taxi rank in station y". The space of repairs are those graphs (V^H, E^H, T^H) with minimum degree at least 1, bidirectional connections between London and Cardiff, and London and Edinburgh, and have at most 5 connections to a station with no taxi ranks, no new taxi ranks added.

Example 12. *Consider graph databases that are relational structures of vocabulary $\{E, P_1, P_2\}$, where E is binary, and P_1 and P_2 are unary. For simplicity, we restrict to structures with Boolean annotations. Let $(\mathcal{C}_h, \mathcal{C}_s, \mathcal{Q}_h, \mathcal{Q}_s, \mathcal{M}, \leq_{d_{\mathcal{Q}_s}})$ be a repair framework over the semiring of natural numbers, where $\mathcal{C}_h$ specifies some ICs, $\mathcal{C}_s := \emptyset$, and $\mathcal{Q}_h^+ := \{P_1(x), \neg P_2(x)\} \cup \{\psi_i(\vec{x}_i) \mid 1 \leq i \leq 10\}$, where the formula $\psi_i(\vec{x}_i) := x_0 = x_i \wedge \bigwedge_{0 \leq j < k \leq i} x_j \neq x_k \wedge \bigwedge_{0 \leq j < i} E(x_j, x_{j+1})$ expresses that $\vec{x}_i = (x_0, \ldots, x_i)$ induces a cycle of length i, and $\mathcal{Q}_h^- := \emptyset$. Finally, we set $\mathcal{Q}_s := \{\exists x (x = x)\}$. The distance $d_{\mathcal{Q}_s}$ is the annotation aware distance given in Example 8. Given a graph database G, the space of repairs of G described by $\mathcal{C}_h$ and $\mathcal{Q}_h$ are those labelled graphs G' that satisfy the ICs in $\mathcal{C}_h$, include all vertices labelled with P_1 with labels intact, include every short cycle of G with possibly different labels, and does not label new vertices with P_2. The graphs with minimal $d_{\mathcal{Q}_s}(G, G')$ are those G' that are closest to G in cardinality.*

4 Annotation Unaware Case

First we recall some important definitions, and stipulate some conventions used in this section. We consider repair frameworks $(\mathcal{C}_h, \mathcal{C}_s, \mathcal{Q}_h, \mathcal{Q}_s, \mathcal{M}, d_{\mathcal{Q}_s})$, where $\mathcal{C}_h$ and $\mathcal{C}_s$ are finite sets of first-order sentences of which $\mathcal{C}_h$ is assumed to be consistent, and $\mathcal{Q}_h = (\mathcal{Q}_h^+, \mathcal{Q}_h^-)$ and $\mathcal{Q}_s$ are finite sets of first-order formulae. In this section, we fix the following canonical inconsistency measure and distance between databases and simply write $(\mathcal{C}_h, \mathcal{C}_s, \mathcal{Q}_h, \mathcal{Q}_s)$ to denote repair frameworks: the inconsistency measure $\mathcal{M}$ is defined as $\mathcal{M}(\mathfrak{D}, \mathcal{C}_s) \coloneqq \sum\{\!\!\{ [\![\neg\phi]\!]^B_{\mathfrak{D}} \mid \phi \in \mathcal{C}_s \}\!\!\}$, and the distance between databases $d_{\mathcal{Q}_s}$ is defined as $d_{\mathcal{Q}_s}(\mathfrak{D}, \mathfrak{D}') \coloneqq \sum\{\!\!\{ \left| [\![\phi]\!]^B_{\mathfrak{D}} - [\![\phi]\!]^B_{\mathfrak{D}'} \right| \mid \phi \in \mathcal{Q}_s^{\mathfrak{D}} \}\!\!\}$. In this section, we focus on annotation unaware repairs as defined in Definition 7 using the semiring of natural numbers. Hence, we essentially consider set-based databases (that is $\mathbb{B}$-databases) that are encoded as $\mathbb{N}$-databases to obtain richer repair frameworks. Since the inconsistency measure and distance are defined to be annotation unaware, and we restrict our attention to repairs of set-based databases encoded as bag-databases, we may stipulate that all repairs obtained will have Boolean annotations. Technically the sets of repairs may also contain non-Boolean annotations, but the set will be invariant under collapsing the annotations to Booleans.

4.1 Classical results on complexity of repair

The complexity of database repairs is often characterised for ICs specified in fragments of tuple-generating dependencies (tgds). Queries used for consistent query answering are often conjunctive queries. Next, we define the fragments of first-order logic of interest.

We write Atoms for the set of *atomic formulae* $R(\vec{x})$, where R is a relation and $\vec{x}$ is a variable tuple. The set Lit of *literals* contain atomic formulae and their negations. A *tgd* is a first-order sentence of the form $\forall\vec{x}\big(\varphi(\vec{x}) \to \exists\vec{y}\psi(\vec{x},\vec{y})\big)$, where φ, ψ are conjunctions of atomic formulae, $\vec{x} = (x_1, \ldots, x_n)$ and $\vec{y} = (y_1, \ldots, y_m)$ are variable tuples, and every universally quantified variable x_i occurs in φ. A *local-as-view (*LAV*)* tgd is a tgd in which φ is a single atomic formula. A *global-as-view (*GAV*)* tgd is a tgd $\forall\vec{x}\big(\varphi(\vec{x}) \to \psi(\vec{x}')\big)$ in which ψ is a single atomic formula such that the variables in $\bar{x}'$ are among the variables of $\bar{x}$. A conjunctive query is a first-order formula of the form $\exists x_1 \ldots \exists x_n(\phi_1 \wedge \cdots \wedge \phi_m)$, where each ϕ_i is an atomic formula. We write CQ for the set of conjunctive queries.

We are interested in the following computational problems. Let $\mathcal{R}$ be a repair framework and q a query.

Problem: Consistent query answering ($\mathcal{R}$-$\mathrm{CQA}(q)$).
Instance: Database $\mathfrak{D}$, a tuple $\vec{t} \in \mathrm{Adom}(\mathfrak{D})^*$. **Question:** Is $\vec{t} \in \mathrm{Ans}(\mathfrak{D}', q)$ for every $\mathcal{R}$-repair of $\mathfrak{D}$?
Problem: Existence of repair ($\exists\mathcal{R}$-repair).
Instance: Database $\mathfrak{D}$. **Question:** Does $\mathfrak{D}$ have an $\mathcal{R}$-repair?

If $\mathcal{R}$ is replaced with a classical repair notion in the definitions above, we obtain the usual decision problems.

ICs		repair notions		$\mathcal{R}$-CQA
$\mathcal{C}_h$	$\mathcal{C}_s$	$\mathcal{Q}_h^+/\mathcal{Q}_h^-$	$\mathcal{Q}_s$	Complexity
LAV	$\emptyset$	Atoms	Atoms	coNP-h [Prop 14]
GAV	$\emptyset$	Atoms	Atoms	coNP-h †
GAV	$\emptyset$	CQ/Atoms	$\emptyset$	coNP-c [Thm. 18]
FO	$\emptyset$	FO	$\emptyset$	coNP-c [Thm. 18]
tgds	$\emptyset$	Atoms	Atoms	Θ_2^p-c [Thm. 19]
FO	$\emptyset$	FO	FO	Θ_2^p-c [Thm. 19]

Table 2: Data complexity overview for consistent query answering. Hardness results hold already for conjunctive queries, while inclusions are proven for first-order queries. Upper/lower bounds transfer to respective classes, e.g., the membership result for FO applies also to subclasses like Lit or Atoms. In the $\mathcal{Q}_h^+/\mathcal{Q}_h^-$-column, if we do not use a slash, then the specification refers to both $\mathcal{Q}_h^+$ and $\mathcal{Q}_h^-$. †: (ten Cate, Fontaine, and Kolaitis 2015, Theorem 5.5)

Ten Cate et al. (2015) present a thorough study on the data complexity of CQA and repair checking problems for set-based repairs. As this section deals with so-called cardinality-based repairs, their results are not directly applicable. We generalise cardinality-based variants of these repair notions.

4.2 Complexity of simple repair notions

For simplicity, we focus on notions where $\mathcal{C}_s = \emptyset$. Table 2 summarises our results on the complexity of CQA.

Remark 13. *For $\mathcal{R} = (\mathsf{GAV}, \emptyset, \mathrm{Atoms}, \mathrm{Atoms})$, the existence of a conjunctive query q such that $\mathcal{R}$-$\mathrm{CQA}(q)$ is* coNP*-hard follows from the proof by Ten Cate et al. 2015, Thm. 5.5 in a straightforward manner. A proof sketch can be found in the full arXiv version (Fröhlich et al. 2025).*

Proposition 14. *Let $\mathcal{R} = (\mathsf{LAV}, \emptyset, \mathrm{Atoms}, \mathrm{Atoms})$. There is a conjunctive query q such that $\mathcal{R}$-$\mathrm{CQA}(q)$ is* coNP*-hard.*

Proof. We reduce from the complement of 3-COLOURABILITY. The components of the repair framework are:

$$\mathcal{C}_h \coloneqq \{\forall xy\, E(x,y) \to \exists jk(C(x,j) \wedge C(y,k) \wedge P(j,k))\},$$
$$\mathcal{Q}_h^+ = \mathcal{Q}_h^- \coloneqq \{E(x,y), P(j,k)\}, \quad \mathcal{Q}_s \coloneqq \{C(x,j)\}.$$

Let $G = (V, E)$ be a graph. The instance $\mathfrak{D}$ is defined as:

$$E^{\mathfrak{D}} \coloneqq \{\,(x,y) \in E\,\}, \quad C^{\mathfrak{D}} \coloneqq \{\},$$
$$P^{\mathfrak{D}} \coloneqq \{1,2,3\}^2 \setminus \{(1,1),(2,2),(3,3)\}.$$

Further define $q \coloneqq \exists x\, j\, k\, \big(C(x,j) \wedge C(x,k) \wedge P(j,k)\big)$.

The intuition of the relational symbols is as follows: E encodes the edges of the graph G, C assigns to each vertex a colour and P encodes inequality between the three possible colours. Now, we claim that the following is true for all $\mathcal{R}$-repairs $\mathfrak{D}'$ of $\mathfrak{D}$: $\mathrm{Ans}(\mathfrak{D}', q)$ is nonempty if and only if $G \notin$ 3-COLOURABILITY.

If $G \notin$ 3-COLOURABILITY, then no valid colouring exists. It follows that in all repairs there must be a vertex x which is assigned two colours to satisfy the hard constraint, thereby satisfying the query q. If $G \in$ 3-COLOURABILITY, then

there exists a valid colouring f for G. Let $\mathfrak{D}_f$ be the following instance:

$$E^{\mathfrak{D}_f} \coloneqq E^{\mathfrak{D}}, \quad C^{\mathfrak{D}_f} \coloneqq \{(x, f(x)) \mid x \in V\}, \quad P^{\mathfrak{D}_f} \coloneqq P^{\mathfrak{D}}.$$

It is easy to see, that $\mathfrak{D}_f$ is a repair of $\mathfrak{D}$ and $q(\mathfrak{D}_f)$ is false, so the consistent answers to q cannot be nonempty. Since 3-COLOURABILITY is NP-complete, the claim follows. □

Proposition 15. *Let $\mathcal{R} = (\mathsf{GAV}, \emptyset, \mathsf{CQ} \cup \mathrm{Atoms}, \emptyset)$. Then $\exists\mathcal{R}$-repair is NP-complete.*

Proof. Inclusion to NP is due to Theorem 16; NP-hardness is shown in the full arXiv version (Fröhlich et al. 2025). □

4.3 Existence of a repair and second-order logic

We now show how the existence of a repair can be reduced to model checking of existential second-order logic, and obtain general upper bounds for existence of repair and CQA.

Theorem 16. *Let $\mathcal{R} = (\mathcal{C}_h, \emptyset, \mathcal{Q}_h, \mathcal{Q}_s)$, where $\mathcal{Q}_s$ is arbitrary and $\mathcal{C}_h, \mathcal{Q}_h^+, \mathcal{Q}_h^- \subseteq \mathrm{FO}$. Then $\exists\mathcal{R}$-repair is in NP. Moreover, the problem is NP-complete for some $\mathcal{C}_h, \mathcal{Q}_h^+, \mathcal{Q}_h^- \subseteq \mathrm{FO}$.*

Proof. The NP-hardness was shown already in Proposition 15. Here we establish inclusion to NP. Recall that, in the annotation unaware case, we may omit $\mathcal{Q}_h^-$ and write $\mathcal{Q}_h$ for $\mathcal{Q}_h^+ \cup \{\neg\varphi \mid \varphi \in \mathcal{Q}_h^-\}$. Fix a finite relational vocabulary $\tau_1 = \{R_1, \ldots, R_n\}$ of some schema $\mathcal{S}$ and let $\tau_2 = \{S_1, \ldots, S_n\}$ denote a disjoint copy of τ_1. We consider databases and repair notions with schema $\mathcal{S}$. Let $\mathcal{R} \coloneqq (\mathcal{C}_h, \emptyset, \mathcal{Q}_h, \mathcal{Q}_s)$ be a repair notion as described in the theorem. Note first that $\mathcal{Q}_s$ does not affect the existence of a repair, so we may assume that $\mathcal{Q}_s = \emptyset$. Given a database $\mathfrak{D}$ over $\mathcal{S}$, let $(\mathcal{C}_h, \emptyset, \mathcal{Q}_h^{\mathfrak{D}}, \emptyset)$ be the repair notion, where $\mathcal{Q}_h^{\mathfrak{D}}$ is the relativisation of $\mathcal{Q}_h$ to $\mathfrak{D}$ computed from $\mathcal{Q}_h$ and $\mathfrak{D}$ in polynomial time. The existence of an $\mathcal{R}$-repair of $\mathfrak{D}$ can be reduced to model checking as follows: The database $\mathfrak{D}$ has an $\mathcal{R}$-repair if and only if $\mathfrak{D}$ satisfies the formula

$$\exists\vec{S}\Big(\bigwedge_{\varphi\in\mathcal{Q}_h^{\mathfrak{D}}} \forall\vec{x}\big(\varphi(\vec{x}) \to \varphi^*[\vec{S}/\vec{R}](\vec{x})\big) \wedge \bigwedge_{\varphi\in\mathcal{C}_h} \varphi^*[\vec{S}/\vec{R}]\Big),$$

where $\varphi^*[\vec{S}/\vec{R}]$ denotes the formula obtained from φ by substituting $\vec{R}$ by $\vec{S}$ and bounding its first-order qualifications to the active domain of the repair obtained from the interpretations of $\exists S_1 \ldots \exists S_n$. The fact that the above model checking can be decided in NP with respect to the size of $\mathfrak{D}$ follows essentially from Fagin's theorem; the fact that data complexity of existential second-order logic is in NP suffices. Note first that $S_1, \ldots S_n$ are of polynomial size, since n and their arities are constant. The fact that the satisfaction of the subsequent formula can be checked in NP follows from the fact that $\mathcal{Q}_h^{\mathfrak{D}}$ and $\mathcal{C}_h$ are polynomial size sets of FO-formulae of constant size. Recall that $\mathcal{R}$ is fixed and $\mathfrak{D}$ is the input, and thus the size of φ^* is constant as well. □

Note that the above model checking problem could easily be transformed into an FO satisfiability problem. Simply append the constructed formula above to the FO description of $\mathfrak{D}$ and remove all (existential) second-order quantifiers that are interpreted existentially in the satisfiability problem.

Lemma 17. *Let $\mathcal{R} = (\mathcal{C}_h, \mathcal{C}_s, \mathcal{Q}_h, \mathcal{Q}_s)$. Then $\exists\mathcal{R}$-repair is reducible in logarithmic space to complement problem of $\mathcal{R}$-CQA(q), where $q \in \mathsf{CQ}$.*

Proof. By ten Cate et al. 2015, Thm. 7.1 the query $q = \exists x P(x)$ with a fresh P suffices. That is, given an instance $\mathfrak{D}$: $\top \in \mathrm{CQA}(q, \mathfrak{D}, \mathcal{R})$ if and only if $\mathfrak{D}$ has no repair. □

Theorem 18. *Let $\mathcal{R} = (\mathcal{C}_h, \emptyset, \mathcal{Q}_h, \emptyset)$, where $\mathcal{C}_h, \mathcal{Q}_h^+, \mathcal{Q}_h^- \subseteq \mathrm{FO}$. Then $\mathcal{R}$-CQA(q) is in coNP for any $q \in \mathrm{FO}$. Moreover, the problem is coNP-complete for some $\mathcal{C}_h \subseteq \mathsf{GAV}$, $\mathcal{Q}_h^+ \subseteq \mathsf{CQ} \cup \mathrm{Atoms}$, $\mathcal{Q}_h^- \subseteq \mathrm{Atoms}$, and $q \in \mathsf{CQ}$.*

Proof. coNP-hardness follows from Prop. 15 and Lemma 17; for inclusion, see (Fröhlich et al. 2025) in arXiv. □

The (less known) complexity class Θ_2^p is defined as $\mathsf{P}^{\mathsf{NP}[\log]}$ meaning a restriction to logarithmic many calls to the NP oracle. By definition, we then have the containment $\Theta_2^p \subseteq \Delta_2^p$. Also it can be characterised by $\mathsf{P}^{||\mathsf{NP}}$ which is having non-adaptive but unrestricted many parallel NP oracle calls (Buss and Hay 1991, Hemachandra 1989).

Theorem 19. *Let $\mathcal{R} = (\mathcal{C}_h, \emptyset, \mathcal{Q}_h, \mathcal{Q}_s)$, with $\mathcal{C}_h$, $\mathcal{Q}_h^+$, $\mathcal{Q}_h^-$, $\mathcal{Q}_s \subseteq \mathrm{FO}$. Then $\mathcal{R}$-CQA$(q) \in \Theta_2^p$ for all $q \in \mathrm{FO}$. Moreover, $\mathcal{R}$-CQA(q) is Θ_2^p-hard for $\mathcal{R} = (\mathcal{C}_h, \emptyset, \mathcal{Q}_h, \mathcal{Q}_s)$, with $\mathcal{C}_h$ a set of tgds, $\mathcal{Q}_h^+, \mathcal{Q}_h^-, \mathcal{Q}_s \subseteq \mathrm{Atoms}$ and some $q \in \mathsf{CQ}$.*

Proof. First, we show Θ_2^p membership. Fix the repair framework $\mathcal{R}$ and query q. Let $\langle\mathfrak{D}, \bar{t}\rangle$ be an instance of $\mathcal{R}$-CQA(q). Define the following auxiliary problem:

Problem: RCE (REPAIRCANDIDATEEXISTENCE)
Instance: Database $\mathfrak{D}$, $n \in \mathbb{N}$ and $\bar{t} \in \mathrm{Adom}(\mathfrak{D})$.
Question: Does a database $\mathfrak{D}'$ exist such that $\mathfrak{D}' \models \mathcal{C}_h$, $\mathrm{Ans}(\mathfrak{D}, \phi) \subseteq \mathrm{Ans}(\mathfrak{D}', \phi)$ for all $\phi \in \mathcal{Q}_h$, $d_{\mathcal{Q}_s}(\mathfrak{D}, \mathfrak{D}') = n$ and $\mathfrak{D}' \models \neg q(\bar{t})$, if $\bar{t} \neq \emptyset$?

It is easy to see that RCE $\in$ NP.

Now, $\mathcal{R}$-CQA(q) can be decided in polynomial time with an RCE-oracle. First, use binary search over the interval $[0, |\mathfrak{D}|]$ to find the smallest n_0 such that $\langle\mathfrak{D}, n_0, \emptyset\rangle \in$ RCE. Second, $\langle\mathfrak{D}, \bar{t}\rangle \in \mathcal{R}$-CQA$(q)$ if and only if $\langle\mathfrak{D}, n_0, \bar{t}\rangle \notin$ RCE. That is, $\bar{t}$ is a consistent answer of q if and only if $\neg q(\bar{t})$ is not true in any repair $\mathfrak{D}'$ with a minimal distance from $\mathfrak{D}$. The first step clearly needs log-many oracle calls, while the second step only needs one.

For hardness reduce from the complement of MT3SAT$^=$ which is Θ_2^P-complete (Spakowski and Vogel 2000).

Problem: MT3SAT$^=$ (MAX-TRUE-3SAT-EQUALITY)
Instance: Two 3SAT formulas φ_0 and φ_1 having the same number of clauses and variables.
Question: Is the maximum number of 1's in satisfying truth assignments for φ_0 equal to that for φ_1?

Let $\bar{a} = a_1a_2a_3$ and $\mathcal{C}_h = \{\psi_{\bar{a}}^i, \psi_1^i, \psi_2^i, \psi_3^i, \psi_4, \psi_5 \mid i, a_1, a_2, a_3 \in \{0, 1\}\}$, where

$$\psi_{\bar{a}}^i = R_{\bar{a}}^i(x_1, x_2, x_3) \to \exists v_1, v_2, v_3 I^i(x_1, v_1)$$

$$\wedge I^i(x_2, v_2) \wedge I^i(x_3, v_3) \wedge T_{\bar{a}}(v_1, v_2, v_3),$$
$$\psi_1^i = I^i(x, v_1) \wedge I^i(x, v_2) \to E(v_1, v_2),$$
$$\psi_2^i = I^i(x, 0) \to I'(x, i, 1) \wedge I'(x, i, 2),$$
$$\psi_3^i = I^i(x_i, 1) \to \exists x_{1-i} F(x_0, x_1) \wedge I^i(x_{1-i}, 1),$$
$$\psi_4 = F(x_1, x_2) \wedge F(x_1, x_3) \wedge D(x_2, x_3) \to \exists s A(s),$$
$$\psi_5 = F(x_1, x_2) \wedge F(x_3, x_2) \wedge D(x_1, x_3) \to \exists s A(s).$$

Furthermore let $q = \exists s A(s)$ and

$$\mathcal{Q}_h^+ := \{E(x, y), D(x, y), R^i(x, y, z), T(x, y, z) | i \in \{0, 1\}\}$$
$$\mathcal{Q}_h^- := \mathcal{Q}_h^+ \cup \{I^0(x, y), I^1(x, y)\} \quad \mathcal{Q}_s := \{I'(x, y), A(s)\}.$$

Let $\langle \varphi_0, \varphi_1 \rangle$ be an instance of MT3SAT$^=$, then

$$T_{\bar{a}} = \{(v_1, v_2, v_3) \mid (v_1, v_2, v_3) \neq (a_1, a_2, a_3)\}$$
$$R_{\bar{a}}^i = \{(x_1, x_2, x_3) \mid x_1 \vee x_2 \vee x_3 \text{ is a clause of } \varphi_i \text{ and } a_i = 0 \text{ if } x_i \text{ is a positive literal, and 1 otherwise}\}$$
$$I^i = \{(x, 0), (x, 1) \mid x \in \varphi_i\} \quad E = \{(0, 0), (1, 1)\}$$
$$D = \{(x_1, x_2) \mid x_1 \neq x_2, x_1, x_2 \in \varphi_0 \text{ or } x_1, x_2 \in \varphi_1\}$$
$$F = A = I' = \emptyset$$

defines the database $\mathfrak{D}$. The intuition for the relational symbols is as follows: $T_{\bar{a}}$ encodes all satisfying assignments of clauses, $R_{\bar{a}}^i$ contains the clauses of φ_i, I^i are the assignments for φ_i, E is equality between 0 and 1, D encodes inequality between variables, F will contain a surjective map between the 1's in I^0 and I^1, A will contain a value if F is not injective and I' will contain every 0 assignment in I^i twice to simulate an increased "weight" for the soft queries.

Now, the intuition for the hard constrains. First notice, that due to ψ_1^i the relation I^i cannot contain both $(x, 0)$ and $(x, 1)$ for all $x \in \varphi_i$. Furthermore $\psi_{\bar{a}}^i$ forces I^i to contain an assignment satisfying φ_i. Next, ψ_2^i duplicates and doubles the assignments set to 0 in I^i into I'. The constraint ψ_3^i creates a surjective map in F between the 1's in I^0 and I^1. Finally, ψ_4 and ψ_5 create a value in A if F is not injective.

As for the soft queries, since I' and A start empty, a repair aims to minimise adding new facts to the relations. Because I' contains two facts for each variable assigned to zero this leads to repairs maximising the number of 1's in their satisfying assignments. The inclusion of A in the soft queries ensures that A contains a values only if ψ_4 or ψ_5 take effect.

We now show that $\langle \mathfrak{D}, \emptyset \rangle \in \mathcal{R}\text{-CQA}(q)$ if and only if $\langle \varphi_0, \varphi_1 \rangle \notin$ MT3SAT$^=$. Assume $\langle \varphi_0, \varphi_1 \rangle \in$ MT3SAT$^=$ and $\mathfrak{D}$ is constructed as described above. First note, that repairs of $\mathfrak{D}$ maximise the number of 1's in satisfying assignments, by having I' as a soft query. Then, because of $\langle \varphi_0, \varphi_1 \rangle \in$ MT3SAT$^=$, there is a one-to-one mapping between the 1's in the maximum assignments for φ_0 and φ_1. A repair then has this bijective map encoded in F, which means that A can be empty. Since repairs are minimal, A must indeed be empty, so q must be false in all repairs, therefore $\langle \mathfrak{D}, \emptyset \rangle \notin \mathcal{R}\text{-CQA}(q)$.

For the other direction, assume $\langle \varphi_0, \varphi_1 \rangle \notin$ MT3SAT$^=$ and $\mathfrak{D}$ is again constructed as described above. Now maximising the number of 1's in satisfying assignments does not lead to a bijective map, so ψ_4 or ψ_5 takes effect and A is not empty in all repairs. Therefore $\langle \mathfrak{D}, \emptyset \rangle \in \mathcal{R}\text{-CQA}(q)$. □

5 Conclusions and Future Work

Our main contribution is the introduction of a novel abstract framework for defining database repairs. We showcase the flexibility of our framework by giving examples of how the main repair notions from the literature can be expressed in our setting. In addition, we introduce novel repair notions that exemplify further the potential of our framework.

As a technical contribution, we initiate the complexity-theoretic study of our framework. Completing this systematic classification remains an avenue for future work. In particular, exploring the possibilities for the soft constraints would require a deeper investigation into inconsistency measures, identifying suitable properties they should satisfy within this framework and considering potential alternative postulates. We examine the complexity of consistent query answering and existence of a repair in the context of Boolean annotations. Unlike in prior studies, determining whether a repair exists is a meaningful problem in our framework, as it can express both integrity constraints that potentially need to be fixed as well as critical properties of databases that should be preserved. Our complexity results are obtained by reducing known complete problems to questions related to repairs in our framework, and by directly relating problems in our framework to problems concerning logics (see Table 2 on page 7 for an overview of our complexity results). Since our framework is logic-based, the non-emptiness of the consistent answers and the existence of a repair can be formulated as model checking problems in logic.

We conclude with future directions and open questions:

- Our complexity results are mainly negative, as we show intractable cases. Can we pinpoint $\mathcal{R}$-repairs where the related complexities are below NP? Does parameterised complexity (Downey and Fellows 1999) help?
- What characterisations can we obtain for enumeration complexity of repairs, or repair checking?
- Does there exist, for every level of the polynomial hierarchy, a fixed repair framework $\mathcal{R}$ such that the existence of repair is complete for that level of the hierarchy?
- To what extent well-established repair semantics can be captured by our framework, and what are its limitations?

Our complexity considerations focus on relational databases and set semantics. A natural next step is to consider bag semantics and K-databases in general. We decided to focus on the relational setting, since otherwise one would need to also develop the needed complexity theory utilising semirings. Here approaches using BSS-machines (Blum et al. 1997) and variants of arithmetic circuits utilising semirings (Jukna 2023) could be fruitful. Finally note that since our framework is logic-based, it would not be hard to adapt it for repairs in conjunction with data integration, where data can be stored and queried under different data models. The integrated view over the data sources could be implemented using logical interpretations. Then, for instance, integrity constraints and repair notions could be specified over the integrated view, which would abstract away the specifics of the details of the concrete repairs on the source data models.

Acknowledgments

The first and second author acknowledge funding by DFG grant ME 4279/3-1 under the id 511769688. The third and fourth author were partially supported by the DFG grant VI 1045/1-1 under the id 432788559.

Finally, we thank the anonymous reviewers for their valuable comments.

References

Alchourrón, C. E.; Gärdenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *J. Symb. Log.* 50(2):510–530.

Arming, S.; Pichler, R.; and Sallinger, E. 2016. Complexity of repair checking and consistent query answering. In *ICDT*, volume 48 of *LIPIcs*, 21:1–21:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Bertossi, L. E. 2019. Repair-based degrees of database inconsistency. In *LPNMR*, volume 11481 of *Lecture Notes in Computer Science*, 195–209. Springer.

Bienvenu, M., and Bourgaux, C. 2020. Querying and repairing inconsistent prioritized knowledge bases: Complexity analysis and links with abstract argumentation. In *KR*, 141–151.

Blum, L.; Cucker, F.; Shub, M.; and Smale, S. 1997. *Complexity and Real Computation*. Berlin, Heidelberg: Springer-Verlag.

Burdick, D.; Fagin, R.; Kolaitis, P. G.; Popa, L.; and Tan, W. 2019. Expressive power of entity-linking frameworks. *J. Comput. Syst. Sci.* 100:44–69.

Buss, S. R., and Hay, L. 1991. On truth-table reducibility to SAT. *Inf. Comput.* 91(1):86–102.

Calautti, M.; Greco, S.; Molinaro, C.; and Trubitsyna, I. 2022. Preference-based inconsistency-tolerant query answering under existential rules. *Artif. Intell.* 312:103772.

Carmeli, N.; Grohe, M.; Kimelfeld, B.; Livshits, E.; and Tibi, M. 2024. Database repairing with soft functional dependencies. *ACM Trans. Database Syst.* 49(2):8:1–8:34.

Decker, H. 2017. Inconsistency-tolerant database repairs and simplified repair checking by measure-based integrity checking. *Trans. Large Scale Data Knowl. Centered Syst.* 34:153–183.

Downey, R. G., and Fellows, M. R. 1999. *Parameterized Complexity*. Monographs in Computer Science. Springer.

Fröhlich, N.; Meier, A.; Pardal, N.; and Virtema, J. 2025. A logic-based framework for database repairs. *CoRR* abs/2306.15516.

Grädel, E., and Gurevich, Y. 1998. Metafinite model theory. *Inf. Comput.* 140(1):26–81.

Grädel, E., and Tannen, V. 2017. Semiring provenance for first-order model checking. *CoRR* abs/1712.01980.

Greco, G.; Greco, S.; and Zumpano, E. 2003. A logical framework for querying and repairing inconsistent databases. *IEEE Trans. Knowl. Data Eng.* 15(6):1389–1408.

Green, T. J.; Karvounarakis, G.; and Tannen, V. 2007. Provenance semirings. In *PODS*, 31–40. ACM.

Guerra, P. T., and Wassermann, R. 2019. Two agm-style characterizations of model repair. *Ann. Math. Artif. Intell.* 87(3):233–257.

Hemachandra, L. A. 1989. The strong exponential hierarchy collapses. *J. Comput. Syst. Sci.* 39(3):299–322.

Jukna, S. 2023. *Tropical Circuit Complexity: Limits of Pure Dynamic Programming*. SpringerBriefs in Mathematics. Springer International Publishing.

Livshits, E.; Kochirgan, R.; Tsur, S.; Ilyas, I. F.; Kimelfeld, B.; and Roy, S. 2021. Properties of inconsistency measures for databases. In *SIGMOD Conference*, 1182–1194. ACM.

Lopatenko, A., and Bertossi, L. E. 2007. Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. In *ICDT*, volume 4353 of *Lecture Notes in Computer Science*, 179–193. Springer.

Lukasiewicz, T.; Malizia, E.; Martinez, M. V.; Molinaro, C.; Pieris, A.; and Simari, G. I. 2022. Inconsistency-tolerant query answering for existential rules. *Artif. Intell.* 307:103685.

Lukasiewicz, T.; Malizia, E.; and Molinaro, C. 2023. Complexity of inconsistency-tolerant query answering in datalog+/- under preferred repairs. In *KR*, 472–481.

Parisi, F., and Grant, J. 2023. On measuring inconsistency in definite and indefinite databases with denial constraints. *Artif. Intell.* 318:103884.

Spakowski, H., and Vogel, J. 2000. Theta$_2^{\mathrm{p}}$-completeness: A classical approach for new results. In *FSTTCS*, volume 1974 of *Lecture Notes in Computer Science*, 348–360. Springer.

Staworko, S.; Chomicki, J.; and Marcinkowski, J. 2012. Prioritized repairing and consistent query answering in relational databases. *Ann. Math. Artif. Intell.* 64(2-3):209–246.

Subrahmanian, V. S., and Amgoud, L. 2007. A general framework for reasoning about inconsistency. In *IJCAI*, 599–504.

ten Cate, B.; Fontaine, G.; and Kolaitis, P. G. 2015. On the data complexity of consistent query answering. *Theory Comput. Syst.* 57(4):843–891.

Yun, B.; Vesic, S.; Croitoru, M.; and Bisquert, P. 2018. Inconsistency measures for repair semantics in OBDA. In *IJCAI*, 1977–1983. ijcai.org.

Zhang, X.; Wang, K.; Wang, Z.; Ma, Y.; Qi, G.; and Feng, Z. 2017. A distance-based framework for inconsistency-tolerant reasoning and inconsistency measurement in dl-lite. *Int. J. Approx. Reason.* 89:58–79.