Theory Comput Syst (2017) 61:191–232
DOI 10.1007/s00224-016-9718-9

From Causes for Database Queries to Repairs and Model-Based Diagnosis and Back

Leopoldo Bertossi[1] · **Babak Salimi**[2]

Published online: 25 November 2016
© Springer Science+Business Media New York 2016

Abstract In this work we establish and investigate connections between causes for query answers in databases, database repairs with respect to denial constraints, and consistency-based diagnosis. The first two are relatively new research areas in databases, and the third one is an established subject in knowledge representation. We show how to obtain database repairs from causes, and the other way around. Causality problems are formulated as diagnosis problems, and the diagnoses provide causes and their responsibilities. The vast body of research on database repairs can be applied to the newer problems of computing actual causes for query answers and their responsibilities. These connections are interesting *per se*. They also allow us, after a transition inspired by consistency-based diagnosis to computational problems on hitting-sets and vertex covers in hypergraphs, to obtain several new algorithmic and complexity results for database causality.

Keywords Causality · Diagnosis · Repairs · Consistent query answering · Integrity constraints

✉ Leopoldo Bertossi
bertossi@scs.carleton.ca

Babak Salimi
bsalimi@cs.washington.edu

[1] School of Computer Science,
Carleton University, Ottawa, Canada

[2] Computer Science and Engineering,
University of Washington, Seattle, USA

1 Introduction

When querying a database, a user may not always obtain the expected results, and the system could provide some explanations. They could be useful to further understand the data or check if the query is the intended one. Actually, the notion of explanation for a query result was introduced in [47], on the basis of the deeper concept of *actual causation*.[1]

A tuple t is an *actual cause* for an answer $\bar{a}$ to a conjunctive query $\mathcal{Q}$ from a relational database instance D if there is a *contingent* set of tuples Γ, such that, after removing Γ from D, $\bar{a}$ is still an answer, but after further removing t from $D \setminus \Gamma$, $\bar{a}$ is not an answer anymore (cf. Section 2.1 for a precise definition). Here, Γ is a set of tuples that has to accompany t so that the latter becomes a counterfactual cause for answer $\bar{a}$. Actual causes and contingent tuples are restricted to be among a pre-specified set of *endogenous tuples*, which are admissible, possible candidates for causes, as opposed to *exogenous tuples*, which may also be present in the database. In rest of this paper, whenever we simply say "cause", we mean "actual cause".

In applications involving large data sets, it is crucial to rank potential causes by their *responsibilities* [47, 48], which reflect the relative (quantitative) degrees of their causality for a query result. The responsibility measure for a cause is based on its *contingency sets*: the smallest (one of) its contingency sets, the strongest it is as a cause.

Actual causation, as used in [47], can be traced back to [32, 33], which provides a model-based account of causation on the basis of *counterfactual dependence*. *Causal responsibility* was introduced in [19], to provide a graded, quantitative notion of causality when multiple causes may over-determine an outcome.

Apart from the explicit use of causality, research on explanations for query results has focused mainly, and rather implicitly, on provenance [13–15, 22, 38, 40, 61]. A close connection between causality and provenance has been established in [47]. However, causality is a more refined notion that identifies causes for query results on the basis of user-defined criteria, and ranks causes according to their responsibilities [48].

Consistency-based diagnosis [53], a form of model-based diagnosis [60, sec. 10.3], is an area of knowledge representation. The problem here is, given the *specification* of a system in some logical formalism and a usually unexpected *observation* about the system, to obtain *explanations* for the observation, in the form of a diagnosis for the unintended behavior (cf. Section 2.3 for a precise definition).

In a different direction, a database instance, D, that is expected to satisfy certain integrity constraints may fail to do so. In this case, a *repair* of D is a database D' that does satisfy the integrity constraints and *minimally departs* from D. Different forms of minimality can be applied and investigated. A *consistent answer* to a query from D and with respect to the integrity constraints is a query answer that is obtained from

[1] In contrast with general causal claims, such as "smoking causes cancer", which refer some sort of related events, actual causation specifies a particular instantiation of a causal relationship, e.g., "Joe's smoking is a cause for his cancer".

all possible repairs, i.e. is invariant or certain under the class of repairs (cf. Section 2.2 for a precise definition). These notions were introduced in [2] (see [7, 9] for surveys).[2]

These three forms of reasoning, namely inferring causes from databases, consistency-based diagnosis, and consistent query answering (and repairs) are all *non-monotonic* [55]. For example, a (most responsible) cause for a query result may not be such anymore after the database is updated. Furthermore, they all reflect some sort of *uncertainty* about the information at hand. In this work we establish natural, precise, useful, and deeper connections between these three reasoning tasks.

More precisely, we unveil a strong connection between computing causes and their responsibilities for conjunctive query answers, on one hand, and computing repairs in databases with respect to *denial constraints*, on the other. These computational problems can be reduced to each other. In order to obtain repairs with respect to *a set* of denial constraints from causes, we investigate causes for queries that are *unions of conjunctive queries*, and develop algorithms to compute causes and responsibilities.

We show that inferring and computing actual causes and their responsibilities in a database setting become diagnosis reasoning problems and tasks. Actually, a causality-based explanation for a conjunctive query answer can be viewed as a diagnosis, where in essence the first-order logical reconstruction of the relational database provides the system description [54], and the observation is the query answer. We obtain causes and their responsibilities -and as a side result, also database repairs- from diagnosis.

Being the causality problems the main focus of this work, we take advantage of algorithms and complexity results both for consistency-based diagnosis on one side; and database repairs and consistent query answering [9], on another. In this way, we obtain new complexity results for the main problems of causality, namely computing actual causes, determining their responsibilities, and obtaining most responsible causes; and also for their decision versions. In particular, we obtain fixed-parameter polynomial-time algorithms for some of them. More precisely, our main results are as follows: (the complexity results are all in data complexity)

1. We characterize actual causes and most responsible actual causes for a Boolean conjunctive query in terms of subset- and cardinality-repairs of the instance with respect to the denial constraint associated to the query (the query being the violation view of the constraint). In this way we can compute causes from repairs.

   In the other direction, we obtain repairs of databases with respect to *sets of denial constraints* from causes for query results. For this, we extend the treatment of causality to *unions of conjunctive queries* (to represent multiple denial constraints). We characterize an actual cause's responsibility in terms of cardinality-repairs. Along the way we provide PTIME algorithms to compute causes and their (minimal) contingency sets for unions of conjunctive queries.

[2] Although not in the context of repairs, consistency-based diagnosis has been applied to consistency restoration of a database with respect to integrity constraints [30].

2. We reduce causes for a Boolean conjunctive query to consistency-based diagnosis for the query being unexpectedly true according to a system description. In particular, we show how to compute actual causes, their contingency sets, and responsibilities using the diagnosis characterization. As a side result, we obtain database repairs from diagnosis.

   Hitting-set-based algorithmic approaches to diagnosis [53] inspire our algorithmic/complexity approaches to causality. In particular, we reformulate the causality problems as hitting-set problems and vertex cover problems on hypergraphs, which allows us to apply results and techniques for the latter to causality.
3. We obtain several new computational complexity results:
   (a) Checking minimal contingency sets can be done in PTIME.
   (b) The *responsibility decision problem* for conjunctive queries, which is about deciding if a tuple's responsibility is greater that a bound v (that is part of the input) is *NP*-complete. However, this problem becomes fixed-parameter tractable, with the parameter being $\frac{1}{v}$.
   (c) The problem of computing responsibilities of causes is $FP^{NP(log(n))}$-complete. Deciding most responsible causes is $P^{NP(log(n))}$-complete.
   (d) The structure of the resulting hitting-set problem allows us to obtain efficient parameterized algorithms and good approximation algorithms for computing causes and minimal contingency sets.
   (e) From the repair connection we obtain that, for consistency based-diagnosis with specifications given by positive implications with disjunctive consequents, the problems of computing minimum-cardinality diagnoses and computing minimum-cardinality diagnoses that contain a given atom are both $FP^{NP(log(n))}$-hard in the size of their underlying Herbrand structure.
4. We define notions of preferred causes; in particular one based on prioritized repairs [59]. We also propose an approach to causality based on interventions that are repair actions that replace attribute values by null values.

The paper is structured as follows. Section 2 introduces technical preliminaries for relational databases, causality in databases, database repairs and consistent query answering, consistency-based diagnosis, and relevant complexity classes. Section 3 characterizes actual causes and responsibilities in terms of database repairs. Section 4 characterizes repairs and consistent query answers in terms of causes and contingency sets for queries that are unions of conjunctive queries, and presents an algorithm for computing both of the latter. Section 5 formulates causality and repair problems as consistency-based diagnosis problems. Section 6 shows complexity and algorithmic results; in particular a fixed-parameter tractability result for causes' responsibilities, and also about consistency based-diagnosis. Section 7 deals with preferred causes. Section 8 discusses several relevant issues, connections and open problems around causality in databases. It also draws some final conclusions. We provide proofs for all the results except for those that are rather straightforward. This is an extended version of [58]. It contains proofs, many improvements in the presentation, and also new developments and results, mainly in Sections 6.2 and 7.

2 Preliminaries

We consider relational database schemas of the form $\mathcal{S} = (U, \mathcal{P})$, where U is the possibly infinite database domain of *constants* and $\mathcal{P}$ is a finite set of *database predicates*[3] of fixed arities. A database instance D compatible with $\mathcal{S}$ can be seen as a finite set of ground atomic formulas (in databases aka. atoms or tuples), of the form $P(c_1, ..., c_n)$, where $P \in \mathcal{P}$ has arity n, and $c_1, \ldots, c_n \in U$.

A *conjunctive query* (CQ) is a formula $\mathcal{Q}(\bar{x})$ of the first-order (FO) logic language, $\mathcal{L}(\mathcal{S})$, associated to $\mathcal{S}$ of the form $\exists \bar{y}(P_1(\bar{s}_1) \wedge \cdots \wedge P_m(\bar{s}_m))$, where the $P_i(\bar{s}_i)$ are atomic formulas, i.e. $P_i \in \mathcal{P}$, and the $\bar{s}_i$ are sequences of terms, i.e. variables or constants.[4] The $\bar{x}$ in $\mathcal{Q}(\bar{x})$ shows all the free variables in the formula, i.e. those not appearing in $\bar{y}$. If $\bar{x}$ is non-empty, the query is *open*. If $\bar{x}$ is empty, the query is *Boolean* (a BCQ), i.e. the query is a sentence, in which case, it is true or false in a database, denoted by $D \models \mathcal{Q}$ and $D \not\models \mathcal{Q}$, respectively. A sequence $\bar{c}$ of constants is an answer to an open query $\mathcal{Q}(\bar{x})$ if $D \models \mathcal{Q}[\bar{c}]$, i.e. the query becomes true in D when the free variables are replaced by the corresponding constants in $\bar{c}$.

An *integrity constraint* is a sentence of language $\mathcal{L}(\mathcal{S})$, and then, may be true or false in an instance for schema $\mathcal{S}$. Given a set IC of integrity constraints for schema $\mathcal{S}$, a database instance D is *consistent* with IC if $D \models IC$; otherwise it is said to be *inconsistent*. In this work we assume that sets of integrity constraints are always finite and logically consistent.

A particular class of integrity constraints is formed by *denial constraints* (DCs), which are sentences κ of the form: $\forall \bar{s} \neg(A_1(\bar{s}_1) \wedge \cdots \wedge A_n(\bar{s}_n))$, where $\bar{s} = \bigcup \bar{s}_i$ and each $A_i(\bar{s}_i)$ is a database atom, i.e. predicate $A_i \in \mathcal{P}$. So as with conjunctive queries, the atoms may contain constants. Denial constraints are exactly the negations of BCQs. Sometimes we use the common representation of DCs as "negative rules" of the form: $\leftarrow \quad A_1(\bar{s}_1), \ldots, A_n(\bar{s}_n)$. We will also consider *functional dependencies* (FDs) as DCs. They are represented by negative rules of the form: $\leftarrow \quad A(\bar{x}_1, \bar{x}_2, y), A(\bar{x}_1, \bar{x}_3, z), y \neq z$, saying that the last attribute of relation A functionally depends upon the attributes holding variables $\bar{x}_1$. They do not contain constants, and correspond to BCQs with inequality.

2.1 Causality and responsibility

Assume that the database instance is split in two, i.e. $D = D^n \cup D^x$, where D^n and D^x denote the disjoint sets of *endogenous* and *exogenous* tuples, respectively.

Actual causes and contingent tuples are usually restricted to be among a prespecified set of endogenous tuples, which are admissible, possible candidates for

[3] As opposed to built-in predicates (e.g. $\neq$) that we assume do not appear, unless explicitly stated otherwise.
[4] In this work, we will assume, unless otherwise explicitly said, that CQs may contain inequality atoms (equality atoms are not an issue, because they can always be eliminated).

causes, as opposed to the exogenous tuples. Actually, the latter provide the context or the background for the problem, and are considered as external factors that are not of interest to the current problem statement or beyond our control. Since no *intervention* (or update, in database parlance) is conceivable on exogenous tuples, they can not be included in any contingency set or be an actual cause. They are assumed to be included in all conceivable hypothetical states of a database.

The endogenous/exogenous partition is application-dependent and captures predetermined factors, such as users preferences that may affect QA-causal analysis. For example, certain tuples or full tables might be identified as irrelevant (or exogenous) in relation to a particular query at hand, or decided to be exogenous or endogenous a priori, independently from the query.

A tuple $t \in D^n$ is called a *counterfactual cause* for a BCQ $\mathcal{Q}$, if $D \models \mathcal{Q}$ and $D \smallsetminus \{t\} \not\models \mathcal{Q}$. A tuple $t \in D^n$ is an *actual cause* for $\mathcal{Q}$ if there exists $\Gamma \subseteq D^n$, called a *contingency set*, such that t is a counterfactual cause for $\mathcal{Q}$ in $D \smallsetminus \Gamma$ [47].

We will concentrate mostly on CQs. However, the definitions of actual cause and contingency set can be applied without a change to *monotone queries* in general [47], in particular to *unions of BCQs* (UBCQs), with or without built-ins.

The *responsibility* of an actual cause t for $\mathcal{Q}$, denoted by $\rho_D(t)$, is the numerical value $\frac{1}{|\Gamma|+1}$, where $|\Gamma|$ is the size of the smallest contingency set for t. We can extend responsibility to all the other tuples in D^n by setting their value to 0. Those tuples are not actual causes for $\mathcal{Q}$.

Example 1 Consider $D = D^n = \{R(a_4, a_3), R(a_2, a_1), R(a_3, a_3), S(a_4), S(a_2), S(a_3)\}$, and the query $\mathcal{Q} : \exists x \exists y (S(x) \wedge R(x, y) \wedge S(y))$. It holds: $D \models \mathcal{Q}$.

Tuple $S(a_3)$ is a counterfactual cause for $\mathcal{Q}$. If $S(a_3)$ is removed from D, $\mathcal{Q}$ is not true anymore. Therefore, the responsibility of $S(a_3)$ is 1. Besides, $R(a_4, a_3)$ is an actual cause for $\mathcal{Q}$ with contingency set $\{R(a_3, a_3)\}$. If $R(a_3, a_3)$ is removed from D, $\mathcal{Q}$ is still true, but further removing $R(a_4, a_3)$ makes $\mathcal{Q}$ false. The responsibility of $R(a_4, a_3)$ is $\frac{1}{2}$, because its smallest contingency sets have size 1. Likewise, $R(a_3, a_3)$ and $S(a_4)$ are actual causes for $\mathcal{Q}$ with responsibility $\frac{1}{2}$.

For the same $\mathcal{Q}$, but with $D = \{S(a_3), S(a_4), R(a_4, a_3)\}$, and the partition $D^n = \{S(a_4), S(a_3)\}$ and $D^x = \{R(a_4, a_3)\}$, it turns out that both $S(a_3)$ and $S(a_4)$ are counterfactual causes for $\mathcal{Q}$.

Remark 1 In the rest of this paper, we will assume in the context of causality that database instances D are partitioned as $D = D^n \cup D^x$, into a subset of endogenous and a set of exogenous tuples, respectively. We will denote with $Causes(D, \mathcal{Q})$ the set of actual causes for the BCQ $\mathcal{Q}$ (being true) from instance D.

2.2 Database repairs

Given a set IC of integrity constraints, a *subset repair* (simply, S-repair) of a possibly inconsistent instance D for schema $\mathcal{S}$ is an instance D' for $\mathcal{S}$ that satisfies IC and

makes $\Delta(D, D') = (D \setminus D') \cup (D' \setminus D)$ minimal under set inclusion.[5] $Srep(D, IC)$ denotes the set of S-repairs of D with respect to IC [2]. Similarly, D' is a *cardinality repair* (simply C-repair) of D if D' satisfies IC and minimizes $|\Delta(D, D')|$. $Crep(D, IC)$ denotes the class of C-repairs of D with respect to IC. C-repairs are always S-repairs. For DCs, S-repairs and C-repairs are obtained from the original instance by deleting an S-minimal, resp. C-minimal, set of tuples. In other words, S- and C-repairs under DCs become maximal (under set inclusion), resp. maximum (in cardinality), consistent subsets of the given instance.

In more general terms, we say that a set is S-minimal in a class of sets $\mathcal{C}$ if it is minimal under set inclusion in $\mathcal{C}$. Similarly, a set is C-minimal (or minimum) if it is minimal in cardinality within $\mathcal{C}$. S-maximality and C-maximality are defined similarly.

Example 2 (ex. 1 cont.) Consider the denial constraint $\kappa : \leftarrow S(x), R(x, y), S(y)$, whose body corresponds to the CQ in Example 1, and is violated by the given instance D.

Here, $Srep(D, \kappa) = \{D_1, D_2, D_3\}$ with $D_1 = \{R(a_4, a_3), R(a_2, a_1), R(a_3, a_3), S(a_4), S(a_2)\}$, $D_2 = \{R(a_2, a_1), S(a_4), S(a_2), S(a_3)\}$, $D_3 = \{R(a_4, a_3), R(a_2, a_1), S(a_2), S(a_3)\}$. The only C-repair is D_1, i.e. $Crep(D, \kappa) = \{D_1\}$.

More generally, different *repair semantics* may be considered to restore consistency with respect to general integrity constraints. They depend on the kind of allowed updates on the database (i.e. tuple insertions/deletions, changes of attribute values), and the minimality conditions on repairs, e.g. subset-minimality, cardinality-minimality, etc.

Given D and IC, a *repair semantics*, S, defines a class $Rep^{\mathsf{S}}(D, IC)$ of S-repairs, which are the intended repairs [9, Sec. 2.5]. All the elements of $Rep^{\mathsf{S}}(D, IC)$ are instances over the same schema of D, and consistent with respect to IC. If D is already consistent, $Rep^{\mathsf{S}}(D, IC)$ contains D as its only member.

Given a repair semantics, S, $\bar{c}$ is a S-*consistent answer* to an open query $\mathcal{Q}(\bar{x})$ if $D' \models \mathcal{Q}[\bar{c}]$ for every $D' \in Rep^{\mathsf{S}}(D, IC)$. A BCQ is S-*consistently true* if it is true in every $D' \in Rep^{\mathsf{S}}(D, IC)$. In particular, if $\bar{c}$ is a consistent answer to $\mathcal{Q}(\bar{x})$ with respect to S-repairs, we say it is an *S-consistent answer*. Similarly for *C-consistent answers*. Consistent query answering for DCs under S-repairs was investigated in detail [18]. C-repairs and consistent query answering under them were investigated in detail in [43]. (Cf. [9] for more references.)

2.3 Consistency-based diagnosis

Consistency-based diagnosis, a form of model-based diagnosis [60, Sec. 10.4], considers problems $\mathcal{M} = (SD, COMPS, OBS)$, where SD is the description in logic of the intended properties of a system under the *explicit* assumption that all the *components*

[5] In general, in the context of repairs, partitions on instances are not considered. However, in Section 7.3 we will bring them into the repair scene.

in *COMPS* are working normally. *OBS* is a FO sentence that represents the observations. If the system does not behave as expected (as shown by the observations), then the logical theory obtained from $SD \cup OBS$ plus the explicit assumption, say $\bigwedge_{c \in COMPS} \neg Ab(c)$, that the components are indeed behaving normally, becomes inconsistent. *Ab* is an *abnormality* predicate.[6]

The inconsistency is captured via the, i.e. those minimal subsets $COMPS'$ of *COMPS*, such that $SD \cup OBS \cup \{\bigwedge_{c \in COMPS'} \neg Ab(c)\}$ is inconsistent. As expected, different notions of minimality can be used at this point.

A *minimal diagnosis* for $\mathcal{M}$ is a minimal subset Δ of *COMPS*, such that $SD \cup OBS \cup \{\neg Ab(c) \mid c \in COMPS \setminus \Delta\} \cup \{Ab(c) \mid c \in \Delta\}$ is consistent. That is, consistency is restored by flipping the normality assumption to abnormality for a minimal set of components, and those are the ones considered to be (jointly) faulty. The notion of minimality commonly used is S-minimality, i.e. a diagnosis that does not have a proper subset that is a diagnosis. We will use this kind of minimality in relation to diagnosis. Diagnosis can be obtained from conflict sets [53].

Example 3 Consider a simple logical gate *Or*, denoted with *o* (the only system component in this case), that receives two digits, x, y, as inputs and outputs a digit $val(x, y)$.

This simple system can be specified in terms of *normal behavior* by the logical formula $\sigma : \neg AB(o) \longrightarrow \forall x \forall y val(x, y) = 0 \longleftrightarrow x = y = 0))$, saying that, when the gate is not abnormal, the output is 0 iff the inputs are both 0.

The logical theory $\{\sigma,\ val(0, 1) = 0\}$ is logically consistent (it can be made true) despite the unexpected observation (namely, output 0 with inputs 0, 1). This is because the system's model allows for abnormal behaviors. However, this theory together with the extra assumption $\neg Ab(o)$, i.e. that the gate is normal, form the theory $\{\sigma,\ val(0, 1) = 0,\ \neg Ab(o)\}$ that is inconsistent in the sense that it can not be made true (in technical terms, it has not models).

2.4 Complexity classes

We recall some complexity classes [52] used in this paper. *FP* is the class of functional problems associated to decision problem in the class *PTIME*, i.e. that are solvable in polynomial time. P^{NP} (or Δ_2^P) is the class of decision problems solvable in polynomial time by a machine that makes calls to an *NP* oracle. For $P^{NP(log(n))}$ the number of calls is logarithmic. It is not known if $P^{NP(log(n))}$ is strictly contained in P^{NP}. $FP^{NP(log(n))}$ is similarly defined.

[6] Here, and as usual, the atom $Ab(c)$ expresses that component c is (behaving) abnormal(ly).

3 Actual Causes From Database Repairs

In this section we characterize actual causes for a BCQ $\mathcal{Q}$ being true in a database instance D in terms of the repairs of D with respect to a denial constraint whose violation view is $\mathcal{Q}$, i.e. the latter asks if the constraint is violated. In essence, the actual causes will become the tuples outside an S-repair. The complement of the latter contains the cause plus a contingency set for the cause. In order to capture responsibility, C-repairs are considered.

Let D be an instance for schema $\mathcal{S}$, and $\mathcal{Q}: \exists \bar{x}(P_1(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m))$ a BCQ. $\mathcal{Q}$ may be unexpectedly true, i.e. $D \models \mathcal{Q}$. Now, $\neg\mathcal{Q}$ is logically equivalent to the DC $\kappa(\mathcal{Q}): \forall \bar{x} \neg(P_1(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m))$. The requirement that $\neg\mathcal{Q}$ holds can be captured by imposing $\kappa(\mathcal{Q})$ on D. Due to $D \models \mathcal{Q}$, it holds $D \not\models \kappa(\mathcal{Q})$. So, D is inconsistent with respect to $\kappa(\mathcal{Q})$, and could be repaired.

Repairs for (violations of) DCs are obtained by tuple deletions. Intuitively, a tuple that participates in a violation of $\kappa(\mathcal{Q})$ in D is an actual cause for $\mathcal{Q}$. S-minimal sets of tuples like this are expected to correspond to S-repairs for D with respect to $\kappa(\mathcal{Q})$.

More precisely, given an instance D, a BCQ $\mathcal{Q}$, and a tuple $t \in D^n$, we consider:

– The class containing the sets of differences between D and those S-repairs that do not contain t, and are obtained by removing a subset of D^n:

$$\mathit{Diff}^{\mathrm{s}}(D, \kappa(\mathcal{Q}), t) = \{D \smallsetminus D' \mid D' \in \mathit{Srep}(D, \kappa(\mathcal{Q})), \\ t \in (D \smallsetminus D') \subseteq D^n\}. \tag{1}$$

– The class containing the sets of differences between D and those C-repairs that do not contain t, and are obtained by removing a subset of D^n:

$$\mathit{Diff}^{\mathrm{c}}(D, \kappa(\mathcal{Q}), t) = \{D \smallsetminus D' \mid D' \in \mathit{Crep}(D, \kappa(\mathcal{Q})), \\ t \in (D \smallsetminus D') \subseteq D^n\}. \tag{2}$$

It holds $\mathit{Diff}^{\mathrm{c}}(D, \kappa(\mathcal{Q}), t) \subseteq \mathit{Diff}^{\mathrm{s}}(D, \kappa(\mathcal{Q}), t)$.

Now, any $\Lambda \in \mathit{Diff}^{\mathrm{s}}(D, \kappa(\mathcal{Q}), t)$ can be written as $\Lambda = \Lambda' \cup \{t\}$. From the S-minimality of S-repairs, it follows that $D \smallsetminus (\Lambda' \cup \{t\}) \models \kappa(\mathcal{Q})$, but $D \smallsetminus \Lambda' \models \neg\kappa(\mathcal{Q})$. That is, $D \smallsetminus (\Lambda' \cup \{t\}) \not\models \mathcal{Q}$, but $D \smallsetminus \Lambda' \models \mathcal{Q}$. As a consequence, t is an actual cause for $\mathcal{Q}$ with contingency set Λ'. We have obtained the following result.

Proposition 1 *Given an instance D and a BCQ $\mathcal{Q}$, $t \in D^n$ is an actual cause for $\mathcal{Q}$ iff $\mathit{Diff}^{\mathrm{s}}(D, \kappa(\mathcal{Q}), t) \neq \emptyset$. Furthermore, if $D \smallsetminus D' \in \mathit{Diff}^{\mathrm{s}}(D, \kappa(\mathcal{Q}), t)$, then $D \smallsetminus (D' \cup \{t\})$ is a minimal contingency set for t.*

Proposition 2 *Given an instance D, a BCQ $\mathcal{Q}$, and $t \in D^n$:*

(a) *If $\mathit{Diff}^{\mathrm{s}}(D, \kappa(\mathcal{Q}), t) = \emptyset$, then $\rho_D(t) = 0$.*
(b) *Otherwise, $\rho_D(t) = \frac{1}{|\Lambda|}$, where $\Lambda \in \mathit{Diff}^{\mathrm{s}}(D, \kappa(\mathcal{Q}), t)$ and there is no $\Lambda' \in \mathit{Diff}^{\mathrm{s}}(D, \kappa(\mathcal{Q}), t)$ such that $|\Lambda'| < |\Lambda|$.*

Corollary 1 *Given an instance D and a BCQ $\mathcal{Q}$: $t \in D^n$ is a most responsible actual cause for $\mathcal{Q}$ iff $Diff^c(D, \kappa(\mathcal{Q}), t) \neq \emptyset$.*

Example 4 (ex. 1 and 2 cont.) Consider the same instance D and query $\mathcal{Q}$. The associated DC is $\kappa(\mathcal{Q})$: $\leftarrow S(x), R(x, y), S(y)$ that we considered in Example 2, where we obtained $Srep(D, \kappa(\mathcal{Q})) = \{D_1, D_2, D_3\}$ and $Crep(D, \kappa(\mathcal{Q})) = \{D_1\}$.

For tuple $R(a_4, a_3)$, $Diff^s(D, \kappa(\mathcal{Q}), R(a_4, a_3)) = \{D \smallsetminus D_2\} = \{\{R(a_4, a_3), R(a_3, a_3)\}\}$, which, by Propositions 1 and 2, confirms that $R(a_4, a_3)$ is an actual cause, with responsibility $\frac{1}{2}$. The complement of $D \smallsetminus D_2$ contains the actual cause $R(a_3, a_3)$ plus a contingency set of it, namely that formed by tuple $R(a_3, a_3)$, which has to be deleted together with the actual cause $R(a_4, a_3)$ to restore consistency (cf. Example 2).

For tuple $S(a_3)$, $Diff^s(D, \kappa(\mathcal{Q}), S(a_3)) = \{D \smallsetminus D_1\} = \{\{S(a_3)\}\}$. So, $S(a_3)$ is an actual cause with responsibility 1.

Similarly, $R(a_3, a_3)$ is an actual cause with responsibility $\frac{1}{2}$, because $Diff^s(D, \kappa(\mathcal{Q}), R(a_3, a_3)) = \{D \smallsetminus D_2, D \smallsetminus D_3\} = \{\{R(a_4, a_3), R(a_3, a_3)\}, \{R(a_3, a_3), S(a_4)\}\}$.

It holds $Diff^s(D, \kappa(\mathcal{Q}), S(a_2)) = Diff^s(D, \kappa(\mathcal{Q}), R(a_2, a_1)) = \emptyset$, because all repairs contain $S(a_2)$, $R(a_2, a_1)$. This means they do not participate in the violation of $\kappa(\mathcal{Q})$ or contribute to make $\mathcal{Q}$ true. So, they are not actual causes for $\mathcal{Q}$, confirming the result in Example 1.

$Diff^c(D, \kappa(\mathcal{Q}), S(a_3)) = \{\{S(a_3)\}\}$. From Corollary 1, $S(a_3)$ is the most responsible cause. □

Remark 2 The results in this section can be easily extended to unions of BCQs. This can be done by associating a DC to each disjunct of the query, and considering the corresponding problems for database repairs with respect to several DCs (cf. Section 4.1).

4 Database Repairs from Actual Causes

In this section we characterize repairs for inconsistent databases with respect to *a set of* DCs in terms of actual causes with their contingency sets. The reduction of repair-related computations to cause-related computations is particularly relevant, because we can take advantage of known complexity results for repairs to obtain new lower-bound complexity results for causality.

Causality has been investigated so far mainly for single conjunctive queries. However, database repairs appear in the context of *sets* of constraints. We concentrate on sets of DCs, which requires extending the analysis of causality to *unions of conjunctive queries*.

More concretely, in this section we characterize repairs of a database instance D with respect to a set Σ of DCs in terms of the actual causes (with their contingency sets) for the union of the conjunctive queries naturally associated to the (bodies of the) DCs. In essence, an S-repair D' is a maximal subset of D that does not contain

any actual cause t, and the tuples other than t and outside D' form a contingency set for t. As expected, C-repairs require the use of most responsible tuples.

Consider an instance D for schema $\mathcal{S}$, and a set of DCs Σ on $\mathcal{S}$. For each $\kappa \in \Sigma$, say $\kappa : \leftarrow A_1(\bar{x}_1), \ldots, A_n(\bar{x}_n)$, consider its associated *violation view* defined by a BCQ, namely V^κ: $\exists \bar{x}(A_1(\bar{x}_1) \wedge \cdots \wedge A_n(\bar{x}_n))$. The answer *yes* to V^κ shows that κ is violated (i.e. not satisfied) by D.

Next, consider the query that is the union of the individual violation views: $V^\Sigma := \bigvee_{\kappa \in \Sigma} V^\kappa$, a *union of* BCQs (UBCQs). Clearly, D violates (is inconsistent with respect to) Σ iff $D \models V^\Sigma$.

It is easy to verify that D, with $D^x = \emptyset$, is consistent with respect to Σ iff $Causes(D, V^\Sigma) = \emptyset$, i.e. there are no actual causes for V^Σ when all tuples are endogenous.

Now, let us collect all *S-minimal contingency sets* associated with an actual cause t for V^Σ:

Definition 1 For an instance D and a set Σ of DCs:

$$Cont(D, V^\Sigma, t) := \{\Gamma \subseteq D^n \mid D \smallsetminus \Gamma \models V^\Sigma, D \smallsetminus (\Gamma \cup \{t\}) \not\models V^\Sigma, \text{ and } \forall \Gamma' \subsetneq \Gamma,\ D \smallsetminus (\Gamma' \cup \{t\}) \models V^\Sigma\}. \tag{3}$$

Notice that for $\Gamma \in Cont(D, V^\Sigma, t)$, it holds $t \notin \Gamma$. When $D^x = \emptyset$, if $t \in Causes(D, V^\Sigma)$ and $\Gamma \in Cont(D, V^\Sigma, t)$, from the definition of actual cause and the S-minimality of Γ, it holds that $\Gamma'' = \Gamma \cup \{t\}$ is an S-minimal subset of D with $D \smallsetminus \Gamma'' \not\models V^\Sigma$. So, $D \smallsetminus \Gamma''$ is an S-repair for D. Then, the following holds.

Proposition 3 *For an instance D, with $D^x = \emptyset$, and a set DCs Σ: $D' \subseteq D$ is an S-repair for D with respect to Σ iff, for every $t \in D \smallsetminus D'$: $t \in Causes(D, V^\Sigma)$ and $D \smallsetminus (D' \cup \{t\}) \in Cont(D, V^\Sigma, t)$.*

To establish a connection between most responsible actual causes and C-repairs, assume that $D^x = \emptyset$, and collect the *most responsible actual causes* for V^Σ:

Definition 2 For an instance D with $D^x = \emptyset$:

$$MRC(D, V^\Sigma) := \{t \in D \mid t \in Causes(D, V^\Sigma), \nexists t' \in Causes(D, V^\Sigma) \text{ with } \rho_D(t') > \rho_D(t)\}. \tag{4}$$

Proposition 4 *For instance D, with $D^x = \emptyset$, and set of DCs Σ: $D' \subseteq D$ is a C-repair for D with respect to Σ iff, for every $t \in D \smallsetminus D'$: $t \in MRC(D, V^\Sigma)$ and $D \smallsetminus (D' \cup \{t\}) \in Cont(D, V^\Sigma, t)$.*

Actual causes for V^Σ, with their contingency sets, account for the violation of some $\kappa \in \Sigma$. Removing those tuples from D should remove the inconsistency. From Propositions 3 and 4 we obtain:

Corollary 2 *Given an instance D and a set DCs Σ, the instance obtained from D by removing an actual cause, resp. a most responsible actual cause, for V^{Σ} together with any of its S-minimal, resp. C-minimal, contingency sets forms an S-repair, resp. a C-repair, for D with respect to Σ.*

Example 5 Consider $D = \{P(a), P(e), Q(a, b), R(a, c)\}$ and $\Sigma = \{\kappa_1, \kappa_2\}$, with $\kappa_1 : \leftarrow P(x), Q(x, y)$ and $\kappa_2 : \leftarrow P(x), R(x, y)$.

The violation views are $V^{\kappa_1} : \exists xy(P(x) \wedge Q(x, y))$ and $V^{\kappa_2} : \exists xy(P(x) \wedge R(x, y))$. For $V^{\Sigma} := V^{\kappa_1} \vee V^{\kappa_2}$, $D \models V^{\Sigma}$ and D is inconsistent with respect to Σ.

Now assume all tuples are endogenous. It holds $Causes(D, V^{\Sigma}) = \{P(a), Q(a, b), R(a, c)\}$, and its elements are associated with sets of S-minimal contingency sets, as follows: $Cont(D, V^{\Sigma}, Q(a, b)) = \{\{R(a, c)\}\}$, $Cont(D, V^{\Sigma}, R(a, c)) = \{\{Q(a, b)\}\}$, and $Cont(D, V^{\Sigma}, P(a)) = \{\emptyset\}$.

From Corollary 2, and $Cont(D, V^{\Sigma}, R(a, c)) = \{\{Q(a, b)\}\}$, $D_1 = D \setminus (\{R(a, c)\} \cup \{Q(a, b)\}) = \{P(a), P(e)\}$ is an S-repair. So is $D_2 = D \setminus (\{P(a)\} \cup \emptyset) = \{P(e), Q(a, b), R(a, c)\}$. These are the only S-repairs.

Furthermore, $MRC(D, V^{\Sigma}) = \{P(a)\}$. From Corollary 2, D_2 is also a C-repair for D.

Remark 3 An actual cause t with any of its S-minimal contingency sets determines a unique S-repair. The last example shows that, with different combinations of a cause and one of its contingency sets, we may obtain the same repair (e.g. for the first two *Cont* sets). So, we may have more minimal contingency sets than minimal repairs. However, we may still have exponentially many minimal contingency sets, so as we may have exponentially many minimal repairs of an instance with respect to DCs, as the following example shows.[7]

Example 6 Consider $D = \{R(1, 0), R(1, 1), \ldots, R(n, 0), R(n, 1), S(1), S(0)\}$ and the DC $\kappa : \leftarrow R(x, y), R(x, z), S(y), S(z)$. D is inconsistent with respect to κ. There are exponentially many S-repairs of D: $D' = D \setminus \{S(0)\}$, $D'' = D \setminus \{S(1)\}$, $D_1 = D \setminus \{R(1, 0), \ldots, R(n, 0)\}$, ..., $D_{2^n} = D \setminus \{R(1, 1), \ldots, R(n, 1)\}$. The C-repairs are only D' and D''.

For the BCQ V^{κ} associated to κ, $D \models V^{\kappa}$, and $S(1)$ and $S(0)$ are actual causes for V^{κ} (courterfactual causes with responsibility 1). All tuples in R are actual causes, each with exponentially many S-minimal contingency sets. For example, $R(1, 0)$ has the S-minimal contingency set $\{R(2, 0), \ldots, R(n, 0)\}$, among exponentially many others (any set built with just one element from each of the pairs $\{R(2, 0), R(2, 1)\}$, ..., $\{R(n, 0), R(n, 1)\}$ is one).

4.1 Causes for Unions of Conjunctive Queries

If we want to compute repairs with respect to sets of DCs from causes for UBCQs using, say Corollary 2, we first need an algorithm for computing the actual causes

[7] Cf. [4] for an example of the latter that uses key constraints, which are DCs with inequalities (with violation views that contain inequality).

and their (minimal) contingency sets for UBCQs. These algorithms could be used as a first stage of the computation of S-repairs and C-repairs with respect to sets of DCs. However, these algorithms (developed in Section 4.2), are also interesting and useful *per se*.

The PTIME algorithm for computing actual causes in [47] is for single conjunctive queries, but does not compute the actual causes' contingency sets. Actually, doing the latter increases the complexity, because *deciding responsibility*[8] of actual causes is *NP*-hard [47] (which would be tractable if we could efficiently compute all (minimal) contingency sets).[9] In principle, an algorithm for responsibilities can be used to compute C-minimal contingency sets, by iterating over all candidates, but Example 6 shows that there can be exponentially many of them.

We first concentrate on the problem of computing actual causes for UBCQs, without their contingency sets, which requires some notation.

Definition 3 Given $\mathcal{Q} = C_1 \vee \cdots \vee C_k$, where each C_i a BCQ, and an instance D:

(a) $\mathfrak{S}(D)$ is the collection of all S-minimal subsets of D that satisfy a disjunct C_i of $\mathcal{Q}$.
(b) $\mathfrak{S}^n(D)$ consists of the S-minimal subsets Λ of D^n for which there exists a $\Lambda' \in \mathfrak{S}(D)$ with $\Lambda \subseteq \Lambda'$ and $\Lambda \smallsetminus \Lambda' \subseteq D^x$. □

$\mathfrak{S}^n(D)$ contains all S-minimal sets of endogenous tuples that simultaneously (and possibly accompanied by exogenous tuples) make the query true. It is easy to see that $\mathfrak{S}(D)$ and $S^n(D)$ can be computed in polynomial time in the size of D.

Now, generalizing a result for CQs in [47], actual causes for a UBCQs can be computed in PTIME in the size of D without computing contingency sets. We formulate this results in terms of the corresponding *causality decision problem* (CDP).

Proposition 5 *Given an instance D, a UBCQ $\mathcal{Q}$, and $t \in D^n$:*

(a) *t is an actual cause for $\mathcal{Q}$ iff there is $\Lambda \in \mathfrak{S}^n(D)$ with $t \in \Lambda$.*
(b) *The causality decision problem (about membership of)*

$$\mathcal{CDP} := \{(D, t) \mid t \in D^n, \text{ and } t \in \mathit{Causes}(D, \mathcal{Q})\} \tag{5}$$

belongs to PTIME.

Proof

(a) Assume $\mathfrak{S}(D) = \{\Lambda_1, \ldots, \Lambda_m\}$, and there exists a $\Lambda \in \mathfrak{S}^n(D)$ with $t \in \Lambda$. Consider a set $\Gamma \subseteq D^n$ such that, for all $\Lambda_i \in \mathfrak{S}^n(D)$ where $\Lambda_i \neq \Lambda$, $\Gamma \cap \Lambda_i \neq \emptyset$ and $\Gamma \cap \Lambda = \emptyset$. With such a Γ, t is an actual cause for $\mathcal{Q}$ with contingency set Γ. So, it is good enough to prove that such Γ always exists. In fact, since all subsets of $\mathfrak{S}^n(D)$ are S-minimal, then, for each $\Lambda_i \in \mathfrak{S}^n(D)$ with $\Lambda_i \neq \Lambda$,

[8] For a precise formulation, see Definition 5.

[9] Actually, [47] presents a PTIME algorithm for computing responsibilities for a restricted class of CQs.

$\Lambda_i \cap \Lambda = \emptyset$. Therefore, Γ can be obtained from the set of difference between each Λ_i and Λ.

Now, if t is an actual cause for $\mathcal{Q}$, then there exist an S-minimal $\Gamma \in D^n$, such that $D \smallsetminus (\Gamma \cup \{t\}) \not\models \mathcal{Q}$, but $D \smallsetminus \Gamma \models \mathcal{Q}$. This implies that there exists an S-minimal subset Λ of D, such that $t \in \Lambda$ and $\Lambda \models \mathcal{Q}$. Due to the S-minimality of Γ, it is easy to see that t is included in a subset of $\mathfrak{S}^n(D)$.

(b) This is a simple generalization of the proof of the same result for single conjunctive queries found in [47]. □

Example 7 (ex. 5 cont.) Consider the query $\mathcal{Q}$: $\exists xy(P(x) \wedge Q(x, y)) \vee \exists xy(P(x) \wedge R(x, y))$, and assume that for D, $D^n = \{P(a), R(a, c)\}$ and $D^x = \{P(e), Q(a, b)\}$. It holds $\mathfrak{S}(D) = \{\{P(a), Q(a, b)\}, \{P(a), R(a, c)\}\}$. Since $\{P(a)\} \subseteq \{P(a), R(a, c)\}$, $\mathfrak{S}^n(D) = \{\{P(a)\}\}$. So, $P(a)$ is the only actual cause for $\mathcal{Q}$.

4.2 Contingency Sets for Unions of Conjunctive Queries

It is possible to develop a (naive) algorithm that accepts as input an instance D and a UBCQ $\mathcal{Q}$, and returns $Causes(D, \mathcal{Q})$; and also, for each $t \in Causes(D, \mathcal{Q})$, its (set of) S-minimal contingency sets $Cont(D, \mathcal{Q}, t)$.

The basis for the algorithm is a correspondence between the actual causes for $\mathcal{Q}$ with their contingency sets and a *hitting-set problem*.[10] More precisely, for a fixed UBCQ $\mathcal{Q}$, consider the *hitting-set framework*

$$\mathfrak{H}^n(D) = \langle D^n, \mathfrak{S}^n(D) \rangle, \tag{6}$$

with $\mathfrak{S}^n(D)$ as in Definition 3. Different computational and decision problems are based on $\mathfrak{H}^n(D)$, and we will confront some below. Notice that hitting-sets (HSs) are all subsets of D^n.

The S-minimal hitting-sets for $\mathfrak{H}^n(D)$ correspond to actual causes with their S-minimal contingencies for $\mathcal{Q}$. Most responsible causes for $\mathcal{Q}$ are in correspondence with hitting-sets for $\mathfrak{H}^n(D)$. This is formalized as follows:

Proposition 6 *For an instance D, a UBCQ $\mathcal{Q}$, and $t \in D^n$:*

(a) *t is an actual cause for $\mathcal{Q}$ with S-minimal contingency set Γ iff $\Gamma \cup \{t\}$ is an S-minimal hitting-set for $\mathfrak{H}^n(D)$.*

(b) *t is a most responsible actual cause for $\mathcal{Q}$ with C-minimal contingency set Γ iff $\Gamma \cup \{t\}$ is a minimum hitting-set for $\mathfrak{H}^n(D)$.*

[10] If $\mathcal{C}$ is a collection of non-empty subsets of a set S, a subset $S' \subseteq S$ is a *hitting-set* for $\mathcal{C}$ if, for every $C \in \mathcal{C}$, $C \cap S' \neq \emptyset$. S' is an S-minimal hitting-set if no proper subset of it is also a hitting-set. S is a minimum hitting-set if it has minimum cardinality.

The proof is similar to that of part (a) of Proposition 5.

Example 8 (ex. 5 and 7 cont.) D and $\mathcal{Q}$ are as before, but now all tuples are endogenous. Here, $\mathfrak{S}(D) = \mathfrak{S}^n(D) = \{\{P(a), Q(a,b)\}, \{P(a), R(a,c)\}\}$. $\mathfrak{H}^n(D)$ has two S-minimal hitting-sets: $H_1 = \{P(a)\}$ and $H_2 = \{Q(a,b), R(a,c)\}$. Each of them implicitly contains an actual cause (any of its elements) with an S-minimal contingency set (what's left after removing the actual cause). H_1 is also the C-minimal hitting-set, and contains the most responsible actual cause, $P(a)$.

Remark 4 For $\mathfrak{H}^n(D) = \langle D^n, \mathfrak{S}^n(D)\rangle$, $\mathfrak{S}^n(D)$ can be computed in PTIME in data complexity, and its elements are bounded in size by $|\mathcal{Q}|$, which is the maximum number of atoms in one of $\mathcal{Q}$'s disjuncts. This is a special kind of hitting-set problems. For example, deciding if there is a hitting-set of size at most k as been called the *d-hitting-set problem* [50], and d is the bound on the size of the sets in the set class. In our case, d would be $|\mathcal{Q}|$.

4.3 Causality, Repairs, and Consistent Answers

Corollary 2 and Proposition 6 can be used to compute repairs. If the classes of S- and C-minimal hitting-sets for $\mathfrak{H}^n(D)$ (with $D^n = D$) are available, computing S- and C-repairs will be in PTIME in the sizes of those classes. However, it is well known that computing minimal hitting-sets is a complex problem. Actually, as Example 6 implicitly shows, we can have exponentially many of them in $|D|$; so as exponentially many minimal repairs for D with respect to a denial constraint. We can see that the complexity of contingency sets computation is in line with the complexities of computing hitting-sets and repairs.

As Corollary 2 and Proposition 6 show, the computation of causes, contingency sets, and most responsible causes via minimal/minimum hitting-set computation can be used to compute repairs and decide about repair questions. Since the hitting-set problems in our case are of the d-hitting-set kind, good algorithms and approximations for the latter (cf. Section 6.1) could be used in the context of repairs.

In the rest of this section we consider an instance D whose tuples are all endogenous, and a set Σ of DCs. For the disjunctive violation view V^Σ, the following result is obtained from Propositions 3 and 4, and Corollary 2.

Corollary 3 *For an instance D, with $D^x = \emptyset$, and a set Σ of DCs, it holds:*

(a) *For every $t \in Causes(D, V^\Sigma)$, there is an S-repair that does not contain t.*
(b) *For every $t \in MRC(D, V^\Sigma)$, there is a C-repair that does not contain t.*
(c) *For every $D' \in Srep(D, \Sigma)$ and $D'' \in Crep(D, \Sigma)$, it holds $D \smallsetminus D' \subseteq Causes(D, V^\Sigma)$ and $D \smallsetminus D'' \subseteq MRC(D, V^\Sigma)$.*

For a projection-free, and a possibly non-Boolean CQ $\mathcal{Q}$, we are interested in its consistent answers from D with respect to Σ. For example, for $\mathcal{Q}(x, y, z)\colon\ R(x, y) \wedge$

$S(y, z)$, the S-consistent (C-consistent) answers would be of the form $\langle a, b, c\rangle$, where $R(a, b)$ and $S(b, c)$ belong to all S-repairs (C-repairs) of D.

From Corollary 3, $\langle a, b, c\rangle$ is an S-consistent (resp. C-consistent) answer iff $R(a, b)$ and $S(b, c)$ belong to D, but they are not actual causes (resp. most responsible actual causes) for V^{Σ}.

The following simple result and its corollary will be useful in Section 6.

Proposition 7 *For an instance D, with $D^x = \emptyset$, a set Σ of DCs, and a projection-free CQ $\mathcal{Q}(\bar{x})$: $P_1(\bar{x}_1) \wedge \cdots \wedge P_k(\bar{x}_k)$:*

(a) *$\bar{c}$ is an S-consistent answer iff, for each i, $P_i(\bar{c}_i) \in (D \smallsetminus Causes(D, V^{\Sigma}))$.*
(b) *$\bar{c}$ is a C-consistent answer iff, for each i, $P_i(\bar{c}_i) \in (D \smallsetminus MRC(D, V^{\Sigma}))$.*

Example 9 (ex. 5 cont.) Consider $\mathcal{Q}(x) : P(x)$. We had $Causes(D, V^{\Sigma}) = \{P(a), Q(a, b), R(a, c)\}$, $MRC(D, V^{\Sigma}) = \{P(a)\}$. Then, $\langle e\rangle$ is both an S- and a C-consistent answer.

Notice that Proposition 7 can easily be extended to conjunctions of ground atomic queries.

Corollary 4 *Given an instance D and a set Σ of DCs, the ground atomic query $\mathcal{Q}$: $P(c)$ is C-consistently true iff $P(c) \in D$ and it is not a most responsible cause for V^{Σ}.*

Example 10 For $D = \{P(a, b), R(b, c), R(a, d)\}$ and the DC $\kappa : \leftarrow P(x, y), R(y, z)$, we obtain: $Causes(D, V^{\kappa}) = MRC(D, V^{\kappa}) = \{P(a, b), R(b, c)\}$.

From Proposition 7, the ground atomic query $\mathcal{Q} : R(a, d)$ is both S- and C-consistently true in D with respect to κ, because, $D \smallsetminus Causes(D, V^{\kappa}) = D \smallsetminus MRC(D, V^{\kappa}) = \{R(a, d)\}$.

The CQs considered in Proposition 7 and its Corollary 4 are not particularly interesting *per se*, but we will use those results to obtain new complexity results for causality later on, e.g. Theorem 3.

5 Causes and Repairs from Consistency-Based Diagnosis

The main objective in this section is to characterize database causality computation as a diagnosis problem.[11] This is interesting *per se*, and will also allow us to apply ideas and techniques from model-based diagnosis to causality. As a side result we obtain a characterization of database repairs in terms of diagnosis.

[11] The other direction is beyond the scope of this work. More importantly, logic-based diagnosis in general is a much richer scenario than that of database causality. In the former, we can have arbitrary logical specification, whereas under data causality, we have only monotone queries at hand.

Let D be an instance for schema $\mathcal{S}$, and $\mathcal{Q} : \exists\bar{x}(P_1(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m))$, a BCQ. Assume $\mathcal{Q}$ is, possibly unexpectedly, true in D. So, for the associated DC $\kappa(\mathcal{Q}) : \forall\bar{x}\neg(P_1(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m))$, $D \not\models \kappa(\mathcal{Q})$. $\mathcal{Q}$ is our *observation*, for which we want to find explanations, using a consistency-based diagnosis approach.

For each predicate $P \in \mathcal{P}$, we introduce predicate Ab_P, with the same arity as P. Intuitively, a tuple in its extension is *abnormal* for P. The "system description", SD, includes, among other elements, the original database, expressed in logical terms, and the DC as true "under normal conditions".

More precisely, we consider the following *diagnosis problem*, $\mathcal{M} = (SD, D^n, \mathcal{Q})$, associated to $\mathcal{Q}$. The FO system description, SD, contains the following elements:

(a) $Th(D)$, which is Reiter's logical reconstruction of D as a FO theory [54] (cf. Example 11).

(b) Sentence $\kappa(\mathcal{Q})^{Ab}$, which is $\kappa(\mathcal{Q})$ rewritten as follows:

$$\kappa(\mathcal{Q})^{Ab}: \forall\bar{x}\neg(P_1(\bar{x}_1) \wedge \neg Ab_{P_1}(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m) \wedge \neg Ab_{P_m}(\bar{x}_m)). \quad (7)$$

(c) Formula (7) can be refined by applying the abnormality predicate, Ab, to endogenous tuples only. For this we need to use additional auxiliary predicates End_P, with the same arity of $P \in \mathcal{S}$, which contain the endogenous tuples in P's extension (see Example 11). Accordingly, we introduce the inclusion dependencies: For each $P \in \mathcal{P}$,

$$\forall\bar{x}(Ab_P(\bar{x}) \rightarrow End_P(\bar{x})), \quad \text{and } \forall\bar{x}(End_P(\bar{x}) \rightarrow P(\bar{x})).$$

The last entry, $\mathcal{Q}$, in $\mathcal{M}$ is the "observation", which together with SD will produce and inconsistent theory, because we make the initial and explicit assumption that all the abnormality predicates are empty (equivalently, that all tuples are normal), i.e. we consider, for each predicate P, the sentence[12]

$$\forall\bar{x}(Ab_P(\bar{x}) \rightarrow \textbf{false}), \quad (8)$$

where, **false** is a propositional atom that is always false.

The second entry in $\mathcal{M}$ is D^n. This is the set of "components" that we can use to try to restore consistency, in this case, by (minimally) changing the abnormality condition on tuples in D^n. In other words, the universal rules (8) are subject to exceptions or qualifications: some endogenous tuples may be abnormal. Each diagnosis shows an S-minimal set of endogenous tuples that are abnormal.

Example 11 (ex. 1 cont.) Consider the query $\mathcal{Q} : \exists x\exists y(S(x) \wedge R(x, y) \wedge S(y))$, and the instance $D = \{S(a_3), S(a_4), R(a_4, a_3)\}$, with $D^n = \{S(a_4), S(a_3)\}$, consider the diagnostic problem $\mathcal{M} = (SD, \{S(a_4), S(a_3)\}, \mathcal{Q})$, with SD containing the sentences in (a)-(c) below:

[12] Notice that these can also be seen as DCs, since they can be written as $\forall\bar{x}\neg Ab_P(\bar{x})$.

(a) Predicate completion axioms plus *unique names assumption*:

$$\forall xy(R(x,y) \leftrightarrow x = a_4 \wedge y = a_3), \quad \forall x(S(x) \leftrightarrow x = a_3 \vee x = a_4), \tag{9}$$

$$\forall xy(End_R(x,y) \leftrightarrow \textbf{false}), \quad \forall x(End_S(x) \leftrightarrow x = a_3 \vee x = a_4), \tag{10}$$

$$a_4 \neq a_3. \tag{11}$$

(b) The denial constraint qualified by non-abnormality, $\kappa(\mathcal{Q})^{Ab}$:

$$\forall xy\neg(S(x) \wedge \neg Ab_S(x) \wedge R(x,y) \wedge \neg Ab_R(x,y) \wedge\ S(y) \wedge \neg Ab_S(y)).$$

In diagnosis formalizations this formula would be usually presented as:

$$\forall xy(\ (\neg Ab_S(x) \wedge \neg Ab_R(x,y) \wedge \neg Ab_S(y)) \longrightarrow \neg(S(x) \wedge R(x,y) \wedge\ S(y))).$$

That is, under the normality assumption, the "system" behaves as intended; in this case, there are no violations of the denial constraint. This main formula in the diagnosis specification can also be written as a disjunctive positive rule:

$$\forall xy(S(x) \wedge R(x,y) \wedge S(y) \longrightarrow\ Ab_S(x) \vee Ab_R(x,y) \vee Ab_S(y)). \tag{12}$$

(c) Abnormality/endogenousity predicates are in correspondence to the database schema, and only endogenous tuples can be abnormal:

$$\forall xy(Ab_R(x,y) \rightarrow End_R(x,y)), \quad \forall xy(End_R(x,y) \rightarrow R(x,y)), \tag{13}$$

$$\forall x(Ab_S(x) \rightarrow End_S(x)), \qquad \forall x(End_S(x) \rightarrow S(x)). \tag{14}$$

In addition to this specification, we have the observation $\mathcal{Q}$:

$$\exists x \exists y(S(x) \wedge R(x,y) \wedge S(y)). \tag{15}$$

Finally, we make the assumption that there are not abnormal tuples:

$$\forall xy(Ab_R(x,y) \rightarrow \textbf{false}), \quad \forall x(Ab_S(x) \rightarrow \textbf{false}). \tag{16}$$

The FO theory formed by (9) - (16) (more precisely, (9), (11), (12), (15) and (16)) is inconsistent.

Now, in more general terms, the observation is $\mathcal{Q}$ (being true), obtained by evaluating query $\mathcal{Q}$ on (theory of) D. In this case, $D \not\models \kappa(\mathcal{Q})$. Since all the abnormality predicates are assumed to be empty, $\kappa(\mathcal{Q})$ is equivalent to $\kappa(\mathcal{Q})^{Ab}$, which also becomes false with respect to D. As a consequence, $SD \cup \{(8)\} \cup \{\mathcal{Q}\}$ is an inconsistent FO theory. A diagnosis is a set of endogenous tuples that, by becoming abnormal, restore consistency.

Definition 4

(a) A *diagnosis* for $\mathcal{M}$ is a $\Delta \subseteq D^n$, such that

$$SD \ \cup\ \{Ab_P(\bar{c}) \mid P(\bar{c}) \in \Delta\} \ \cup\ \{\neg Ab_P(\bar{c}) \mid P(\bar{c}) \in D \smallsetminus \Delta\} \ \cup\ \{\mathcal{Q}\}$$

is consistent.

(b) $Diag^s(\mathcal{M}, t)$ denotes the set of S-minimal diagnoses for $\mathcal{M}$ that contain tuple $t \in D^n$.

(c) $Diag^c(\mathcal{M}, t)$ denotes the set of C-minimal diagnoses in $Diag^s(\mathcal{M}, t)$.

Example 12 (ex. 11 cont.) The theory can be made consistent by giving up (16), and making S-minimal sets of tuples abnormal. According to (13)-(14), those tuples have to be endogenous.

$\mathcal{M}$ has two S-minimal diagnosis: $\Delta_1 = \{S(a_3)\}$ and $\Delta_4 = \{S(a_4)\}$. The first one corresponds to replacing the second formula in (16) by $\forall x(Ab_S(x) \wedge x \neq a_3 \rightarrow$ **false**), obtaining now a consistent theory.

Here, $Diag^s(\mathcal{M}, S(a_3)) = Diag^c(\mathcal{M}, S(a_3)) = \{\{S(a_3)\}\}$, and $Diag^s(\mathcal{M}, S(a_4)) = Diag^c(\mathcal{M}, S(a_4)) = \{\{ S(a_4)\}\}$.

If $R(a_4, a_3)$ is also endogenous, then also $\{R(a_4, a_3)\}$ becomes a minimal diagnosis.

By definition, $Diag^c(\mathcal{M}, t) \subseteq Diag^s(\mathcal{M}, t)$. Diagnoses for $\mathcal{M}$ and actual causes for $\mathcal{Q}$ are related.

Proposition 8 *Consider an instance D, a BCQ $\mathcal{Q}$, and the diagnosis problem $\mathcal{M}$ associated to $\mathcal{Q}$. Tuple $t \in D^n$ is an actual cause for $\mathcal{Q}$ iff $Diag^s(\mathcal{M}, t) \neq \emptyset$.*

The responsibility of an actual cause t is determined by the cardinality of the diagnoses in $Diag^c(\mathcal{M}, t)$.

Proposition 9 *For an instance D, a BCQ $\mathcal{Q}$, the associated diagnosis problem $\mathcal{M}$, and a tuple $t \in D^n$, it holds:*

(a) $\rho_D(t) = 0$ *iff* $Diag^c(\mathcal{M}, t) = \emptyset$.
(b) *Otherwise,* $\rho_D(t) = \frac{1}{|\Delta|}$, *where* $\Delta \in Diag^c(\mathcal{M}, t)$.

For the proofs of Propositions 8 and 9, it is easy to verify that the conflict sets of $\mathcal{M}$ coincide with the sets in $\mathfrak{S}(D^n)$ (cf. Definition 3). The results are obtained from the characterization of minimal diagnosis as minimal hitting-sets of sets of conflict sets (cf. Section 2 and [53]) and Proposition 6.

Example 13 (ex. 12 cont.) From Propositions 8 and 9, $S(a_3)$ and $S(a_4)$ are actual cases, with responsibility 1. If $R(a_4, a_3)$ is also endogenous, it also becomes an actual cause with responsibility 1.

In consistency-based diagnosis, minimal diagnoses can be obtained as S-minimal hitting-sets of the collection of S-minimal *conflict sets* (cf. Section 2) [53]. In our case, conflict sets are S-minimal sets of endogenous tuples that, if not abnormal (only endogenous ones can be abnormal), and together, and possibly in combination with exogenous tuples, make (7) false.

It is easy to verify that the conflict sets of $\mathcal{M}$ coincide with the sets in $\mathfrak{S}(D^n)$ (cf. Definition 3 and Remark 4). As a consequence, conflict sets for $\mathcal{M}$ can be computed in PTIME, the hitting-sets for $\mathcal{M}$ contain actual causes for $\mathcal{Q}$, and the hitting-set problem for the diagnosis problems is of the d-hitting-set kind.

The reduction from causality to consistency-based diagnosis allows us to apply constructions and techniques for the latter (cf. [27, 49]), to the former.

Example 14 (ex. 11 cont.) The diagnosis problem $\mathcal{M} = (SD, \{S(a_4), S(a_3)\}, \mathcal{Q})$ gives rise to the hitting-set framework $\mathfrak{H}^n(D) = \langle\{S(a_4), S(a_3)\}, \{\{(S(a_3), S(a_4)\}\}\rangle$, with $\{S(a_3), S(a_4)\}$ corresponding to the conflict set $c = \{S(a_4), S(a_3)\}$.

$\mathfrak{H}^n(D)$ has two minimum hitting-sets: $\{S(a_3)\}$ and $\{S(a_4)\}$, which are the S-minimal diagnosis for $\mathcal{M}$. Then, the two tuples are actual causes for $\mathcal{Q}$ (cf. Proposition 8). From Proposition 9, $\rho_D(S(a_3)) = \rho_D(S(a_4)) = 1$.

The solutions to the diagnosis problem can be used for computing repairs.

Proposition 10 *Consider an instance D with $D^x = \emptyset$, a set of DCs of the form $\kappa: \forall\bar{x}\neg(P_1(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m)$, and their associated "abnormality-aware" integrity constraints*[13] *in (7) (in this case we do not need End_P atoms).*

Each S-minimal diagnosis Δ gives rise to an S-repair of D, namely $D_\Delta = D \smallsetminus \{P(\bar{c}) \in D \mid Ab_P(\bar{c}) \in \Delta\}$; and every S-repair can be obtained in this way. Similarly, for C-repairs using C-minimal diagnoses.

Example 15 (ex. 13 cont.) The instance $D = \{S(a_3), S(a_4), R(a_4, a_3)\}$, with all tuples endogenous, has three (both S- and C-) repairs with respect to the DC $\kappa: \forall xy\neg(S(x) \wedge R(x, y) \wedge S(y))$, namely $D_1 = \{S(a_3), R(a_4, a_3)\}$, $D_2 = \{S(a_4), R(a_4, a_3)\}$, and $D_3 = \{S(a_3), S(a_4)\}$. They can be obtained as $D_{\Delta_1}, D_{\Delta_2}, D_{\Delta_3}$ from the only (S- and C-) diagnoses, $\Delta_1 = \{S(a_3)\}$, $\Delta_2 = \{S(a_4)\}$, $\Delta_3 = \{R(a_4, a_3)\}$, resp.

We have characterized repairs in terms of diagnosis. Thinking of the other direction, and as a final remark, it is worth observing that the very particular kind of diagnosis problem we introduced above (with restricted logical formulas) can be formulated as a *preferred-repair problem* [9, Sec. 2.5]. Without going into the details, the idea is to materialize tables for the auxiliary predicates Ab_P and End_P, and consider the DCs of the form (7) (with the End_P atoms when not all tuples are endogenous), plus the DCs (8), saying that the initial extensions for the Ab_P predicates are empty. If D is inconsistent with respect to this set of DCs, the S-repairs that are obtained by only *inserting* endogenous tuples into the extensions of the Ab_P predicates correspond to S-minimal diagnosis, and each S-minimal diagnosis can be obtained in this way.

6 Complexity Results

There are *three main computational problems* in database causality. For a BCQ $\mathcal{Q}$ and database D:

(a) The *causality problem* (CP) is about computing the actual causes for $\mathcal{Q}$. Its decision version of this problem, CDP, is stated in (5). Both CP and CDP

[13] Notice that these are not denial constraints.

are solvable in polynomial time [47], which can be extended to UBCQs (cf. Proposition 5).

(b) The *responsibility problem* (RP) is about computing the responsibility $\rho_D(t)$ of a given actual cause t. (Since a tuple that is not an actual cause has responsibility 0, this problem subsumes (a).) This is a maximization problem due to the minimization of $|\Gamma|$ in the denominator.

We will consider the decision version of this problem that, as usual for maximization problems [29], asks whether the real-valued function being computed (responsibility in this case) takes a value greater than a given threshold v of the form $\frac{1}{k}$, for a positive integer k.

Definition 5 For a BCQ $\mathcal{Q}$, the *responsibility decision problem* (RDP) is (deciding about membership of):

$$\mathcal{RDP}(\mathcal{Q}) = \{(D, t, v) \mid t \in D^n, v \in \{0\} \cup \{\frac{1}{k} \mid k \in \mathbb{N}^+\}, \text{ and } D \models \mathcal{Q} \text{ and } \rho_D(t) > v\},$$

that is, deciding if a tuple has a responsibility greater than a bound v (as a cause for $\mathcal{Q}$).

The complexity analysis of RDP in [47] is restricted to conjunctive queries without self-joins. Here, we will generalize the complexity analysis for RDP to general CQs.

(c) Computing the *most responsible actual causes* (MRC). Its decision version, MRCDP, the *most responsible cause decision problem*, is a natural problem, because actual causes with the highest responsibility tend to provide most interesting explanations for query answers [47, 48].

Definition 6 For a BCQ $\mathcal{Q}$, the *most responsible cause decision problem* is (membership of):

$\mathcal{MRCDP}(\mathcal{Q}) = \{(D, t) | t \in D^n$ and $0 < \rho_D(t)$ is a maximum for $D\}$.

We start by analyzing a more basic decision problem, that of deciding if a set of tuples Γ is an S-minimal contingency set associated to a cause t (cf. (3)). Due to the results in Sections 3 and 4, it is clear that there is a close connection between this problem and the *S-repair checking* problem [9, Chap. 5], about deciding if instance D' is an S-repair of instance D with respect to a set of integrity constraints. Actually, the following result is obtained from the PTIME solvability of the S-repair checking problem for DCs [18] (see also [1]).

Proposition 11 *For a BCQ $\mathcal{Q}$, the minimal contingency set decision problem (MCSDP), i.e. $\mathcal{MCSDP}(\mathcal{Q}) := \{(D, t, \Gamma) \mid \Gamma$ is minimal element in $Cont(D, \mathcal{Q}, t)\}$, belongs to PTIME.*

Proof To decide if $(D, t, \Gamma) \in \mathcal{MCSDP}(\mathcal{Q})$, it is good enough to observe, from Proposition 1, that $(D, t, \Gamma) \in \mathcal{MCSDP}(\mathcal{Q})$ iff $D \smallsetminus (\Gamma \cup \{t\})$ is an S-repair for D with respect to $\kappa(\mathcal{Q})$. S-repair checking can be done in PTIME in data [18]. □

We could also consider the decision problem defined in Proposition 11, but with C-minimal Γ. We will not use results about this problem in the following. Furthermore, its connection with the C-repair checking problem is less direct. As one can see from Section 3, C-minimal contingency sets correspond to a repair semantics somewhere between the S-minimal and C-minimal repair semantics (a subclass of Srep, but a superclass of Crep): It is about an S-minimal repair with minimum cardinality that does not contain a particular tuple.

Now we establish that RDP is *NP*-complete for CQs in general. The *NP*-hardness is shown in [47]. Membership of NP is obtained using Proposition 11.

Theorem 1 (a) *For every BCQ $\mathcal{Q}$, $\mathcal{RDP}(\mathcal{Q}) \in NP$.*
(b) [47] *There are CQs $\mathcal{Q}$ for which $\mathcal{RDP}(\mathcal{Q})$ is NP-hard.*

Proof (a) We give a non-deterministic PTIME algorithm to solve RDP. Non-deterministically guess a subset $\Gamma \subseteq D^n$, return *yes* if $|\Gamma| < \frac{1}{v}$ and $(D, t, \Gamma) \in \mathcal{MCSDP}$; otherwise return *no*. According to Proposition 11 this can be done in PTIME in data complexity. □

In order to better understand the complexity of RP, the responsibility computation problem, we will investigate the *functional*, non-decision version of RDP.

The main source of complexity when computing responsibilities is related to the hitting-set problem associated to $\mathfrak{H}^n(D) = \langle D^n, \mathfrak{S}^n(D)\rangle$ in Remark 4 (cf. (6)). In this case, it is about computing the cardinality of a minimum hitting-set that contains a given vertex (tuple) t. That this is a kind of *d-hitting-set problem* [50] will be useful in Section 6.1.

Remark 5 Our responsibility problem can also be seen as a *vertex cover problem* on the *hypergraph*[14]

$$\mathfrak{G}^n(D) = \langle D^n, \mathfrak{S}^n(D)\rangle \tag{17}$$

associated to $\mathfrak{H}^n(D) = \langle D^n, \mathfrak{S}^n(D)\rangle$ (that is, the hitting-set framework can be seen as a hypergraph). In it, the hyperedges are the members of $\mathfrak{S}^n(D)$. Determining the responsibility of a tuple t becomes the problem on hypergraphs of determining the size of a minimum vertex cover that contains vertex t (among all vertex covers that contain the vertex). Again, in this problem the hyperedges are bounded in size by $|\mathcal{Q}|$.[15]

[14] In an hypergraph $\mathcal{H}$, a set of vertices is a *vertex cover* if it intersects every hyperedge. A minimal vertex cover has no proper subset that is also a vertex cover. A *minimum* vertex cover has minimum cardinality among the vertex covers. Similarly, an *independent set* of $\mathcal{H}$ is a set of vertices such that no pair of them is contained in a hyperedge. Maximal and maximum independent sets are defined in an obvious manner.

[15] We recall that repairs of databases with respect to DCs can be characterized as maximal independent sets of *conflict hypergraphs* (*conflict graphs* in the case of FDs) whose vertices are the database tuples, and hyperedges connect tuples that together violate a DC [4, 18].

Example 16 For $\mathcal{Q}: \exists xy(P(x) \wedge R(x, y) \wedge P(y))$, and $D = D^n = \{P(a), P(c), R(a, c), R(a, a)\}$, $\mathfrak{S}(D) = \mathfrak{S}^n(D) = \{\{P(a), R(a, a)\}, \{P(a), P(c), R(a, c)\}\}$.

The hypergraph $\mathfrak{G}^n(D)$ has D as set of vertices, and its hyperedges are $\{P(a), R(a, a)\}$ and $\{P(a), P(c), R(a, c)\}$. Its minimal vertex covers are: $vc_1 = \{P(a)\}$, $vc_2 = \{P(c), R(a, a)\}$, $vc_3 = \{R(a, a), R(a, c)\}$. Only the first has minimum cardinality. Accordingly, its only element, $P(a)$, is an actual cause with responsibility 1. The other tuples are actual causes with responsibility $\frac{1}{2}$.

Remark 6 To simplify the presentation of the next computational problems (Lemmas 1 and 2 and Proposition 12), we will formulate and address them in terms of graphs. However, they still hold for hypergraphs [43, 44], which is what we need for the complexity results obtained in the rest of this section.

Lemma 1 (*representation lemma*) *There is a fixed database schema $\mathcal{S}$ and a BCQ $\mathcal{Q} \in \mathcal{L}(\mathcal{S})$, without built-ins, such that, for every graph $G = (V, E)$, with non-empty E, and $v \in V$, there is an instance D for $\mathcal{S}$ and a tuple $t \in D$, such that the size of a minimum vertex cover of G containing v is the inverse of the responsibility of t as an actual cause for $\mathcal{Q}$.*

Proof Consider a graph $G = (V, E)$, and assume the vertices of G are uniquely labeled.

Consider the database schema with relations $Ver(v_0)$ and $Edges(v_1, v_2, e)$, and the conjunctive query $\mathcal{Q}: \exists v_1 v_2 e(Ver(v_1) \wedge Ver(v_2) \wedge Edges(v_1, v_2, e))$. *Ver* stores the vertices of G, and *Edges*, the labeled edges. For each edge $(v_1, v_2) \in E$, $Edges$ contains n tuples of the form (v_1, v_2, i), where n is the number of vertices in G. All the values in the third attribute of $Edges$ are different, say from 1 to $n \times |E|$. This padding of relation $Edge$ will ensure in the rest of the proof that C-minimal contingency sets for the query answer consist only of vertices, i.e. elements of Ver (as opposed to Edge tuples). The size of the padded instance is still polynomial in the size of G. It is clear that $D \models \mathcal{Q}$.

Assume VC is the minimum vertex cover of G that contains vertex v, where tuple t is $Ver(v)$. Consider the set of tuples $\Lambda = \{Ver(x) \mid x \in VC\}$. Since $v \in VC$, $\Lambda = \Lambda' \cup \{Ver(v)\}$. Then, $D \smallsetminus (\Lambda' \cup Ver(v)) \not\models Q$. This is because for every tuple $Edge(v_i, v_j, k)$ in the instance, either v_i or v_j belongs to VC. Due to the minimality of VC, $D \smallsetminus \Lambda' \models \mathcal{Q}$. Therefore, tuple $Ver(v)$ is an actual cause for $\mathcal{Q}$.

Suppose Γ is a C-minimal contingency set associated to $Ver(v)$. Due to the C-minimality of Γ, it entirely consists of tuples in *Ver*. It holds that $D \smallsetminus (\Gamma \cup \{Ver(v)\}) \not\models \mathcal{Q}$ and $D \smallsetminus \Gamma \models \mathcal{Q}$. Consider the set $VC' = \{x \mid Ver(x) \in \Gamma\} \cup \{v\}$. Since $D \smallsetminus (\Gamma \cup \{Ver(v)\}) \not\models \mathcal{Q}$, for every tuple $Edge(v_i, v_j, k)$ in D, either $v_i \in VC'$ or $v_j \in VC'$. Therefore, VC' is a minimum vertex cover of G that contains v. It holds that $\rho_D(Ver(v)) = \frac{1}{1+|\Gamma|}$. So, the size of a minimum vertex cover of G that contains v can be obtained from $\rho_D(Ver(v))$. □

Having represented our responsibility problem as a graph-theoretic problem, we first consider functional computational problems in graphs.

Definition 7 The *minimal vertex cover membership problem* (MVCMP) consists in, given a graph $G = (V, E)$, and a vertex $v \in V$ as inputs, computing the size of a minimum vertex cover of G that contains v.

Lemma 2 *Given a graph G and a vertex v in it, there is a graph G' extending G that can be constructed in polynomial time in $|G|$, such that the size of a minimum vertex cover for G that contains v and the size of a minimum vertex cover for G' coincide.*

Proof The size of $VC_G(v)$, the minimum vertex cover of G that contains the vertex v, can be computed from the size of I_G, the maximum independent set of G, that *does not* contain v. In fact,

$$|VC_G(v)| = |G| - |I_G|. \tag{18}$$

Since I_G is a maximum independent set that does not contain v, it must contain one of the adjacent vertices of v (otherwise, I_G is not maximum, and v can be added to I_G). Therefore, $|VC_G(v)|$ can be computed from the size of a maximum independent set I that contains v', one of the adjacent vertices of v.

Given a graph G and a vertex v' in it, a graph G' that extends G can be constructed in polynomial time in the size of G, in such a way that: there is a maximum independent set I of G containing v' iff v' belongs to every maximum independent set of G' iff the sizes of maximum independent sets for G and G' differ by one.

Actually, graph G' can be obtained by adding a new vertex v'' that is connected only to the neighbors of v'. It holds:[16]

$$|I_G| = |I_{G'}| - 1, \tag{19}$$

$$|I_{G'}| = |G'| - |VC_{G'}|, \tag{20}$$

where $I_{G'}$ is a maximum indent set in G', and $VC_{G'}$ is a minimum vertex cover of G'. From (18), (19) and (20), we obtain: $|VC_G(v)| = |VC_{G'}|$. □

From Lemma 2 and the $FP^{NP(log(n))}$-completeness of determining the size of a maximum clique in a graph [39], we obtain:

Proposition 12 *The MVCMP problem for graphs is $FP^{NP(log(n))}$-complete.*

Proof We prove membership by describing an algorithm in $FP^{NP(log(n))}$ for computing the size of the minimum vertex cover of a graph $G = (V, E)$ that contains a vertex $v \in V$. We use Lemma 2, and build the extended graph G'.

The size of a minimum vertex cover for G' gives the size of the minimum vertex cover of G that contains v. Since computing the maximum cardinality of a clique can be done in time $FP^{NP(log(n))}$ [39], computing a minimum vertex cover can be done

[16] This construction is inspired by [43, Lemma 1]. More details can be found in [44].

in the same time (just consider the complement graph). Therefore, MVCMP belong to $FP^{NP(log(n))}$.

Hardness can be obtained by a reduction from computing minimum vertex covers in graphs to MVCMP. Given a graph G construct the graph G' as follows: Add a vertex v to G and connect it to all vertices of G. It is easy to see that v belongs to all minimum vertex covers of G'. Furthermore, the sizes of minimum vertex covers for G and G' differ by one. Consequently, the size of a minimum vertex cover of G can be obtained from the size of a minimum vertex cover of G' that contains v. Computing the minimum vertex cover is $FP^{NP(log(n))}$-complete. This follows from the $FP^{NP(log(n))}$-completeness of computing the maximum cardinality of a clique in a graph [39]. □

Theorem 2

(a) *For every BCQ, $\mathcal{Q}$, computing the responsibility of a tuple as a cause for $\mathcal{Q}$ is in $FP^{NP(log(n))}$.*

(b) *There is a database schema and a BCQ $\mathcal{Q}$, without built-ins, such that computing the responsibility of a tuple as a cause for $\mathcal{Q}$ is $FP^{NP(log(n))}$-complete.*

Proof For membership, we observe from Remark 5 that computing a tuple's responsibility amounts to computing the size of a minimum vertex cover containing the tuple in the graph associated to the query and instance at hand. By Proposition 12, this problem belongs to $FP^{NP(log(n))}$.

Hardness follows from Lemma 1 and the hardness result in Proposition 12. □

Now we address the most responsible causes problem, MRCDP (cf. Definition 6). We use the connection with consistent query answering of Section 4.3, namely Corollary 4, and the $P^{NP(log(n))}$-completeness of consistent query answering under the C-repair semantics for queries that are conjunctions of ground atoms and a particular DC [43, Theorem 4].

Theorem 3

(a) *For every BCQ, $\mathcal{MRCDP}(\mathcal{Q}) \in P^{NP(log(n))}$.*

(b) *There is a database schema and a BCQ $\mathcal{Q}$, without built-ins, for which $\mathcal{MRCDP}(\mathcal{Q})$ is $P^{NP(log(n))}$-complete.*

Proof

(a) To show that $\mathcal{MRCDP}(\mathcal{Q})$ belongs to $P^{NP(log(n))}$, consider first the hitting-set framework $\mathfrak{H}^n(D) = \langle D^n, \mathfrak{S}^n(D)\rangle$ (cf. Definition 3 and 6) and its associated hypergraph $\mathfrak{G}^n(D)$ (cf. (17)).

It holds that t is a most responsible cause for $\mathcal{Q}$ iff $\mathfrak{H}^n(D)$ has a C-minimal hitting-set that contains t (cf. Proposition 6). Therefore, t is a most responsible cause for $\mathcal{Q}$ iff t belongs to some minimum vertex cover of $\mathfrak{G}^n(D)$.

It is easy to see that $\mathfrak{G}^n(D)$ has a minimum vertex cover that contains t iff $\mathfrak{G}^n(D)$ has a maximum independent set that does not contains t. Checking if t belongs to all maximum independent set of $\mathfrak{G}^n(D)$ can be done in $P^{NP(log(n))}$ [43, Lemma 2].

If t belongs to all independent sets of $\mathfrak{G}^n(D)$, then $(D, t) \notin \mathcal{MRCDP}(\mathcal{Q})$; otherwise $(D, t) \in \mathcal{MRCDP}(\mathcal{Q})$. As a consequence, the decision can be made in time $P^{NP(log(n))}$.

(b) The proof is by a reduction, via Corollary 4, from consistent query answering under the C-repair semantics for queries that are conjunctions of ground atoms, which was proved to be $P^{NP(log(n))}$-complete in [43, Theorem 4]. Actually, that proof (of hardness) uses a particular database schema $\mathcal{S}$ and a DC κ. In our case, we can use the same schema $\mathcal{S}$ and the violation query V^κ associated to κ (cf. Section 4). □

From Proposition 6 and the $FP^{NP(log(n))}$-completeness of determining the size of C-repairs for DCs [43, Theorem 3], we obtain the following for the computation of the highest responsibility value.

Proposition 13

(a) *For every BCQ, computing the responsibility of the most responsible causes is in* $FP^{NP(log(n))}$.

(b) *There is a database schema and a BCQ* $\mathcal{Q}$*, without built-ins, for which computing the responsibility of the most responsible causes is* $FP^{NP(log(n))}$*-complete.*

Proof

(a) To show the membership of $FP^{NP(log(n))}$, consider the hypergraph $\mathfrak{G}^n(D)$ as obtained in Theorem 3. The responsibility of most responsible causes for $\mathcal{Q}$ can be obtained from the size of the minimum vertex cover of $\mathfrak{G}^n(D)$ (cf. Proposition 6). The size of the minimum vertex cover in a graph can be computed in $FP^{NP(log(n))}$, which is obtained from the membership of $FP^{NP(log(n))}$ of computing the maximum cardinality of a clique in graph [39].

It is easy to verify that minimum vertex covers in hypergraphs can be computed in the same time.

(b) This is by a reduction from the problem of determining the size of C-repairs for DCs shown to be $FP^{NP(log(n))}$-complete in [43, Theorem 3]. Actually, that proof (of hardness) uses a particular database schema $\mathcal{S}$ and a DC κ. In our case, we may consider the same schema $\mathcal{S}$ and the violation query V^κ associated to κ (cf. Section 4).

The size of C-repairs for an inconsistent instance D of the schema $\mathcal{S}$ with respect to κ can be obtained from the responsibility of most responsible causes for V^κ (cf. Corollary 2). □

6.1 FPT of Responsibility

We need to cope with the intractability of computing most responsible causes. The area of *fixed parameter tractability* (FPT) [28] provides tools to attack this problem. In this regard, we recall that a decision problem with inputs of the form (I, p), where p is a distinguished parameter of the input, is fixed parameter tractable (or belongs to the class FPT), if it can be solved in time $O(f(|p|) \cdot |I|^c)$, where c and the hidden constant do not depend on $|p|$ or $|I|$, and f does not depend on $|I|$.

In our case, the *parameterized version of the decision problem* $\mathcal{RDP}(\mathcal{Q})$ (cf. Definition 5) is denoted with $\mathcal{RDP}^p(\mathcal{Q})$, and the distinguished parameter is k, such that $v = \frac{1}{k}$.

That $\mathcal{RDP}^p(\mathcal{Q})$ belongs to FPT can be obtained from its formulation as a d-hitting-set problem (d being the fixed upper bound on the size of the sets in the set class). The latter problem consists in, given a hitting-set framework with d-bounded subsets and an element t (a tuple in our case), deciding if there is a hitting-set of cardinality smaller that k that contains t. This problem belongs to FPT.

Theorem 4 *For every BCQ $\mathcal{Q}$, $\mathcal{RDP}^p(\mathcal{Q})$ belongs to FPT, where the parameter is the inverse of the responsibility bound.*

Proof First, there is a PTIME parameterized algorithm for the d-hitting-set problem about deciding if there is a hitting-set of size at most k that runs in time $O(e^k + n)$, with n the size of the underlying set and $e = d - 1 + o(d^{-1})$ [50]. In our case, $n = |D|$, and $d = |\mathcal{Q}|$ (cf. also [26]).

Now, to decide if the responsibility of a given tuple t is greater than $v = \frac{1}{k}$, we consider the associated hypergraph $\mathfrak{G}^n(D)$, and we decide if it has a vertex cover that contains t and whose size is less than k. In order to answer this, we use Lemma 2, and build the extended hypergraph $\mathfrak{G}'$.

The size of a minimum vertex cover for $\mathfrak{G}'$ gives the size of the minimum vertex cover of $\mathfrak{G}^n(D)$ that contains t. If $\mathfrak{G}^n(D)$ has a vertex cover that contains t of size less than k, then $\mathfrak{G}'$ has a vertex cover of size less than k. If $\mathfrak{G}'$ has a vertex cover of size less than k, its minimum size for a vertex cover is less than k. Since this minimum is the same as the size of a minimum vertex cover for $\mathfrak{G}^n(D)$ that contains t, $\mathfrak{G}^n(D)$ has a vertex cover of size less than k that contains t. As a consequence, it is good enough to decide if $\mathfrak{G}'$ has a vertex cover of size less than k. For this, we use the hitting-set formulation of this hypergraph problem, and the already mentioned FPT algorithm. □

This result and the corresponding algorithm sketched in its proof show that the higher the required responsibility degree, the lower the computational effort needed to compute the actual causes with at least that level of responsibility. In other terms, parameterized algorithms are effective for computing actual causes with high responsibility or most responsible causes. In general, parameterized algorithms are very effective when the parameter is relatively small [28].

Now, in order to compute most responsible causes, we could apply, for each actual cause t, the just presented FPT algorithm on the hypergraph $\mathfrak{G}^n(D)$, starting with $k = 1$, i.e. asking if there is vertex cover of size less than 1 that contains t. If the algorithm returns a positive result, then t is a counterfactual cause, and has responsibility 1. Otherwise, the algorithm will be launched with $k = 2, 3, \ldots, |D^n|$, until a positive result is returned. (The procedure can be improved through binary search on $k = 1, 2, 3, \ldots, m$, with m possibly much smaller than $|D|$.)

The complexity results and algorithms provided in this section can be extend to UBCQs. This is due to Remark 2 and the construction of $\mathfrak{S}^n(D)$, which the results in this section build upon.

For the d-hitting-set problem there are also efficient parameterized approximation algorithms [11]. They could be used to approximate the responsibility problem. Furthermore, approximation algorithms developed for the minimum vertex cover problem on bounded hypergraphs [34, 51] should be applicable to approximate most responsible causes for query answers. Via the causality/repair connection (cf. Section 4.3), it should be possible to develop approximation algorithms to compute S-repairs of particular sizes, C-repairs, and consistent query answers with respect to DCs.

6.2 Complexity of Diagnosis with Positive Disjunctive Rules

It is known that consistency-based diagnosis decision problems can be unsolvable [53]. However, there are decidable classes of FO diagnosis specifications, and those classes are amenable to complexity analysis. However, there is little research on the complexity analysis of solvable classes of consistency-based diagnosis problems. The connection we established in the previous sections between causality, repairs and consistency-based diagnosis can be used to obtain new algorithmic and complexity results for the latter. Without trying to be exhaustive about this, which is beyond the scope of this paper, we give an example of the kind of results that can be obtained.

Considering the diagnosis problem we obtained in Section 5, we can define a class of diagnosis problems. Cf. Example 11, in particular (12), for motivation.

Definition 8 A *disjunctive positive* (DP) diagnosis specification Σ is a consistent FO logical theory, such that:

(a) Σ has a signature (schema) consisting of a finite set of constants, a set of predicates $\mathcal{S}$, a set $\mathcal{S}^{ab}$ of predicates of the form Ab_R,[17] with $R \in \mathcal{S}$, and Ab_R with the same arity of R. $\mathcal{S}$ and $\mathcal{S}^{ab}$ are mutually disjoint.

(b) Σ is inconsistent with $AB^{\mathcal{S}} := \{\forall \bar{x}(Ab_R(\bar{x}) \rightarrow \textbf{false}) \mid R \in \mathcal{S}\}$.

(c) Consists of:

(c1) Sentences of the form $\forall \bar{x}(C(\bar{x}) \longrightarrow \bigvee_i Ab_{R_i}(\bar{x}_i))$, with $\bar{x}_i \subseteq \bar{x}$, and $C(\bar{x})$ a conjunction of atoms that does not include Ab-atoms of any kind.

[17] Or any other "abducible" predicates that are different from those in $\mathcal{S}$.

(c2) Sentences of the forms $\forall \bar{x}(Ab_R(\bar{x}) \longrightarrow (R(\bar{x}) \wedge S(\bar{x})))$, with $S \in \mathcal{S}$.
(c3) A finite background universal theory $\mathcal{T}$ expressed in terms of predicates in $\mathcal{S}$ (and constants) that has a unique Herbrand model.[18]

As above, a diagnosis is a set of Ab_R-atoms that, when assumed to be true, restores the consistency of the correspondingly modified $\Sigma \cup AB^{\mathcal{S}}$.

There are at least two important computational tasks that emerge, namely, given a *disjunctive positive* (DP) diagnosis specification Σ together with $AB^{\mathcal{S}}$:

1. The *minimum-cardinality diagnosis* (MCD) problem, about computing minimum-cardinality diagnoses.
2. The *minimal membership diagnosis*, (MMD) about computing minimum-cardinality diagnoses that contain a given Ab-atom.

It is not difficult to see that these problems are computable (or solvable in their decision versions). Now we can obtain complexity lower bounds for them. Actually, in Section 5, the *responsibility* and *most responsible causes problem* were reduced to diagnosis problems for specifications that turned out to be disjunctive positive (see (12)).

More specifically, Proposition 9 reduces computing responsibility of a tuple to computing the size of a minimum-cardinality diagnosis that contains the tuple. Furthermore, as a simple corollary of Proposition 9, we obtain the computation of minimum-cardinality diagnoses allows us to compute most responsible causes. Now, combining all this with Proposition 13 and Theorem 2, we obtain the following lower bounds for our diagnosis problems.

Theorem 5 *For disjunctive positive diagnosis specifications, the MCD and MMD problems are $FP^{NP(log(n))}$-hard in the size of their underlying Herbrand structure.*

7 Preferred Causes for Query Answers

In Section 3 we characterized causes and most responsible causes in terms of S-repairs and C-repairs, resp. We could generalize the notion of a cause and/or its responsibility by using, in principle, any *repair semantics* S. The latter is represented by a class of repairs $Rep^{\mathsf{S}}(D, \Sigma)$, of D with respect to a set of denial constraints (cf. Section 2.2). When dealing with (sets of) DCs, the repair actions can only be of certain kinds. Usually tuple deletions have been considered. This is the case of the S- and C-repairs we have considered in this work so far.

We could go beyond and consider the notion of *prioritized repair* [59]. Also changes of attribute values can be the chosen repair actions, including the use of *null values*, to "destroy" joins (again, with different semantics, e.g. with nulls *à la* SQL [8, 12]).

[18] This condition is clearly satisfied by the logical reconstruction of a relational database, but can be relaxed in several ways.

In this section we explore the possibility of introducing a notion of *preferred cause* that is based on a given repair semantics. This idea is inspired by (and generalizes) the characterization of causes in terms of repairs that we obtained before, namely (1), (2), Proposition 1, and Corollary 1.

If we define causes and their (minimal) contingency sets on the basis of a given repair semantics, the minimality condition involved in the latter will have an impact on the notion of minimal (or preferred) contingency set, and indirectly, on the notions of responsibility and most responsible cause.[19]

In Section 7.1 we summarize prioritized repairs. In Section 7.2 we impose preferences on causes on the basis of the prioritized repairs introduced in [59] (and further investigated in [25]). In Section 7.3, we briefly investigate the possibility of capturing endogenous repairs, i.e. that do not change exogenous tuples, by means of a priority relation. Finally, in Section 7.4, we briefly consider the possibility of defining (preferred) causes via attribute-based repairs that use null values.

7.1 Prioritized Repairs

The prioritized repairs in [59] are based on a *priority relation*, $\succ$, on the set of database tuples. In the case of a pair of (mutually) *conflicting tuples*, i.e. that simultaneously violate a constraint in a given set set of DCs (possibly in company of other tuples), the repair process reflects the user preference -as captured by the priority relation- on the tuples that are privileged to be kept in the database, i.e. in the intended repairs.

Given such a priority relation, in [59] different classes of prioritized repairs are introduced, namely the class of *globally optimal repairs*, that of *Pareto-optimal repairs*, and that of *completion-optimal repairs*. Intuitively, each class relies on a different *optimality criterion* that is used to extend the priority relation $\succ$ on pairs of conflicting facts to a priority relation on the set of S-repairs. As a consequence, each of these three classes is contained in that of the S-repairs. In particular, all these repairs are based on tuple deletions.

Let us denote with $Rep^{\succ,\mathcal{X}}(D, \Sigma)$ the class of all prioritized repairs based on $\succ$ and the optimality criterion $\mathcal{X}$. Its elements are called $(\succ, \mathcal{X})$-*prioritized repairs* of D with respect to a the set Σ of DCs. It holds $Rep^{\succ,\mathcal{X}}(D, \Sigma) \subseteq Srep(D, \Sigma)$, and then, all the elements of $Rep^{\succ,\mathcal{X}}(D, \Sigma)$ are subsets of D.

In order to show a concrete class $Rep^{\succ,\mathcal{X}}(D, \Sigma)$, we first recall the definitions of priority relation and *global-optimal repair* from [59].

[19] We could say that the efforts in [35, 36] to modify the Halpern-Pearl (HP) original definition of causality are about considering more appropriate restrictions on contingencies. Since in some cases the original HP definition does not provide intuitive results regarding causality, the modifications avoid this by recognizing some contingencies as "unreasonable" or "farfetched".

Definition 9 Given an instance D and a set of denial constraints Σ , a binary relation $\succ$ on D is a *priority relation* with respect to Σ if: (a) $\succ$ is acyclic, and (b) for every $t, t' \in D$, if $t \succ t'$, then t and t' are mutually conflicting.[20]

Definition 10 Let D be an instance, Σ a set of DCs, and $\succ$ a corresponding priority relation. Let D' and D'' be two consistent sub-instances of D. D' is a *global improvement* of D'' if $D' \neq D''$, and for every tuple $t' \in D'' \setminus D'$, there exists a tuple $t \in D' \setminus D''$ such that $t \succ t'$. D' is a *global-optimal repair* of D, if D' is an S-repair and does not have a global improvement.

In this definition, the optimality criterion, a possible $\mathcal{X}$ above, is that of global-optimal repair, or $(\succ, go)$-repair, which leads to a class $Rep^{\succ,go}(D, \Sigma)$. We consider this repair semantics just for illustration purposes.

Example 17 Consider the database schema *Author*(*Name, JournalN*), $Journal$ ($JournalN$, $Topic$, $Paper\#$), and the following instance D:

D:

Author	*Name*	*JournalN*	Journal	*JournalN*	*Paper#*	*Topic*
	John	TKDE		TKDE	30	XML
	Tom	TKDE		TKDE	31	CUBE
	John	TODS		TODS	32	XML

Consider the following denial constraint:

$$\kappa: \ \forall xyzz' \neg(Author(x,y) \wedge Journal(y, z, z') \wedge x = \mathsf{John} \wedge z' = \mathsf{XML}), \qquad (21)$$

capturing the condition that "John has not published a paper in a journal that has published papers on XML".

D is inconsistent with respect to κ, and contains the following sets of conflicting tuples:

$$C_1 = \{Author(John,TKDE), Journal(TKDE,30,XML)\},$$
$$C_2 = \{Author(John,TODS), Journal(TODS,32,XML)\}.$$

[20] We can say $\{t, t'\}$ is a *conflict*, i.e. the two tuples jointly participate in the violation of one of the DCs in Σ.

D has the following S-repairs, each obtained by deleting one tuple from each of C_1 and C_2, to resolve the conflicts:

$$D_1 = \{Author(Tom, TKDE), Journal(TKDE, 31, CUBE), Author(John, TODS), Journal(TKDE, 30, XML)\}$$

$$D_2 = \{Author(Tom, TKDE), Journal(TKDE, 31, CUBE), Journal(TKDE, 30, XML), Journal(TODS, 32, XML)\}$$

$$D_3 = \{Author(Tom, TKDE), Journal(TKDE, 31, CUBE), Author(John, TKDE), Journal(TODS, 32, XML)\}$$

$$D_4 = \{Author(Tom, TKDE), Journal(TKDE, 31, CUBE), Author(John, TKDE), Author(John, TODS)\}$$

(a) Now, assume a user prefers to resolve a conflict by removing tuples from the *Author* table rather than the *Journal* table, maybe because he considers the latter more reliable than the former. This is expressed the following priority relationships on conflicting tuples: *Journal*(*TKDE*, 30, *XML*) $\succ$ *Author*(*John*, *TKDE*) and *Journal*(*TODS*, 32, *XML*) $\succ$ *Author*(*John*, *TODS*).

In this case only D_2 is a global-optimal repair. Actually, D_2 is a global improvement over each of D_1, D_3 and D_4. For D_1, for example: $D_2 \setminus D_1 = \{Journal(TODS, 32, XML)\}$ and $D_1 \setminus D_2 = \{Author(John, TODS)\}$. We can see that, for each tuple in $D_2 \setminus D_1$, there is a tuple in $D_1 \setminus D_2$ that has a higher priority. Therefore, D_2 is a global improvement on D_1. So, in this case $Rep^{\succ,go}(D, \kappa) = \{D_2\}$

In this case, the uniqueness of the global-optimal repair is quite natural as the preference relation among conflicting tuples is a total relation. So, we know how to resolve every conflict according to the user preferences.

(b) For a more subtle situation, assume the user has the priorities as before, but in addition he tends to believe that John has a paper in TODS. In this case we have only the relationship $Journal(TKDE, 30, XML) \succ' Author(John, TKDE)$, and no preference for resolving the second conflict. Now both D_1 and D_2 are global-optimal repairs. That is, now $Rep^{\succ',go}(D, \kappa) = \{D_1, D_2\}$.

7.2 Preferred causes from prioritized repairs

According to the motivation provided at the beginning of this section, we now define *preferred causes* on the basis of a class of prioritized repairs. (Compare (22) below with (1) and (2).) To keep things simple, we concentrate on single BCQs, $\mathcal{Q}$, whose associated denial constraints are denoted by $\kappa(\mathcal{Q})$.

Before providing technical details, we motivate the notion of preference in the context of causality. In this direction, first notice that under actual causality, we already make a difference -and only this difference- between endogenous and exogenous tuples. We can think of extending this priority relation among tuples in such a way

that, for example, we prioritize -as causes- tuples in a given relation R, and we are not interested in tuples in another relation S. So, the user can specify a priority relation between the two relations, or different *scores* for these relations [46].

In Section 4.2 actual causes and their minimal contingency sets for a UBCQ were characterized as the minimal hitting-sets of the collection $\mathcal{C}$ of minimal subsets of a database that entail the query. Those minimal hitting-sets are obtained by removing at least one tuple from each of the elements of $\mathcal{C}$ (cf. Proposition 6). At this point, user preferences, or priorities, could be applied to tuples that belong to a same set $\mathcal{C}$.

Definition 11 Given an instance D and a BCQ $\mathcal{Q}$, tuples t and t' are *jointly-contributing* if $t \neq t'$, and there exists an S-minimal $\Lambda \subseteq D$ such that $\Lambda \models \mathcal{Q}$ and $t, t' \in \Lambda$.

Now we define priority relations on jointly-contributing tuples.

Definition 12 Given an instance D and a BCQ $\mathcal{Q}$, a binary relation $\succ_c$ on D is a *causal priority relation* with respect to $\mathcal{Q}$ if: (a) $\succ_c$ is acyclic, and (b) for every $t, t' \in D$, if $t \succ_c t'$, then t and t' are jointly-contributing tuples.

This definition introduces a natural notion of preference on causality. Actually, this way of approaching priorities on causes is in (inverse) correspondence with preference on repairs as based on priority relations on conflicting tuples. To see this, first observe that for a given instance D and BCQ $\mathcal{Q}$: t and t' are jointly-contributing tuples for $\mathcal{Q}$ iff t and t' are mutually conflicting tuples for $\kappa(\mathcal{Q})$.

Next, in the context of prioritized repairs, a priority relation reflects a user preference on tuples that are preferred to be kept in the database. This is the inverse of causality, where a causal priority relation, as we defined it, reflects the tuples that are preferred to be (hypothetically or counterfactually) removed from database, to make them preferred causes.

In the following assume $\succ_c^r$ is the inverse of a causal priority relation $\succ_c$. That is, $t \succ_c^r t'$ iff $t' \succ_c t$. Clearly, $\succ_c^r$ is acyclic, and can be imposed, with the expected result, on pairs of conflicting tuples. As a consequence, $\succ_c^r$ can be used to define prioritized repairs.

Definition 13 Let D be an instance, $\mathcal{Q}$ a BCQ, t a tuple in D, $\succ_c$ a causal priority relation on D's tuples.

(a)

$$Diff^{\succ_c^r, \mathcal{X}}(D, \kappa(\mathcal{Q}), t) := \{D \smallsetminus D' \mid D' \in Rep^{\succ_c^r, \mathcal{X}}(D, \kappa(\mathcal{Q})), \text{ and } t \in D \smallsetminus D'\}. \tag{22}$$

(b) $t \in D$ is a $(\succ_c, \mathcal{X})$-*preferred cause* for $\mathcal{Q}$ iff $Diff^{\succ_c^r, \mathcal{X}}(D, \kappa(\mathcal{Q}), t) \neq \emptyset$.

Notice that every $(\succ_c, \mathcal{X})$-preferred cause is also an actual cause. This follows from Proposition 1 and the fact that prioritized repairs are also S-repairs.

Similarly to Proposition 2, for each $\Lambda \in \mathit{Diff}^{\succ_c^r,\mathcal{X}}(D, \kappa(\mathcal{Q}), t)$, it holds that $t \in \Lambda$, t is a $(\succ_c, \mathcal{X})$-preferred cause, and also an actual cause for $\mathcal{Q}$ with S-minimal contingency set $\Lambda \setminus \{t\}$. In particular, t's responsibility can be defined and computed as before, but now restricting its contingency sets to those of the form $\Lambda \setminus \{t\}$, with $\Lambda \in \mathit{Diff}^{\succ_c^r,\mathcal{X}}(D, \kappa(\mathcal{Q}), t)$. In this way, a causal priority relation may affect the responsibility of a cause (with respect to the non-prioritized case).

Example 18 (example 17 cont.) The following BCQ query $\mathcal{Q}$ is true in D:

$$\exists \mathit{JournalN}\, \exists \mathit{Paper\#}(\mathit{Author}(\mathsf{John},\ \mathit{JournalN}) \wedge \mathit{Journal}(\mathit{JournalN}, \mathit{Paper\#}, \mathsf{XML}));$$

and its associated DC $\kappa(\mathcal{Q})$ is κ in (21).

We want to obtain the preferred causes for $\mathcal{Q}$ being, possibly unexpectedly, true in D, with the following preferences: (a) We prefer those among the *Author* tuples. (b) It is likely that John does have a paper in TODS. So, we prefer *Author(John, TODS)* not to be the cause.

These causal priorities are in inverse correspondence with those in the second case of Example 17(b) about priorities for repairs. That is, for our causal priority relation $\succ_c$ here, its inverse $\succ_c^r$ is $\succ'$ in Example 17(b). There we had $\mathit{Rep}^{\succ',go}(D, \kappa(Q)) = \{D_1, D_2\}$, which we can use to apply Definition 13.

We obtain as the globally-optimal causes, i.e. as $(\succ_c, go)$-causes: *Author*(*John*, *TKDE*), *Author*(*TODS*, 32, *XML*) and *Author*(*John*, *TODS*), all with the same responsibility, $\frac{1}{2}$. □

Notice that Definition 13 can be easily extended to UBCQs. This is done, as earlier in this work, by considering the set Σ of denial constraints associated to a UBCQ. In the other direction, we recall that if we start with a set of DCs Σ, the corresponding UBCQ is denoted with V^{Σ}.

As we did in the previous sections of this work, we could take advantage of algorithmic and complexity results about prioritized repairs [25, 59], to obtain complexity results for preferred causes problems. As an example, we establish the complexity of the minimal contingency set decision problem for $(\succ_c, go)$*-preferred causes*. More precisely, for an instance D and a UBCQ $\mathcal{Q}$, the *minimal preference-contingency set* (decision) problem is about deciding if a set of tuples Γ is an S-minimal contingency set associated to a $(\succ_c, go)$-preferred cause t.

Notation $\mathit{Cont}^{\succ_c,\mathcal{X}}(D, \mathcal{Q}, t) := \{\Lambda \setminus \{t\} \mid \Lambda \in \mathit{Diff}^{\succ_c^r,\mathcal{X}}(D, \kappa(\mathcal{Q}), t)\}$ is the class of all S-minimal contingency sets for a $(\succ_c, \mathcal{X})$-preferred cause t.

Definition 14 For a UBCQ $\mathcal{Q}$, the *minimal preference-contingency set* decision problem is about membership of:

$\mathcal{MPCDP}(\mathcal{Q}) := \{(D, \succ_c, t, \Gamma) \mid t \in D, \Gamma \subseteq D,$ and $\Gamma \in \mathit{Cont}^{\succ_c,go}(D, \mathcal{Q}, t)\}$.

From Definition 13, there is a close connection between $\mathcal{MPCDP}$ and the *global-optimal repair checking problem*, i.e. about deciding if an instance D' is a

$(\succ, go)$-repair of D with respect to a set of denial constraints. If we accept functional dependencies (FDs) among our denial constraints (and then, UBCQs that involve inequalities), the following result can be obtained from the *NP*-completeness of globally-optimal repair checking [59] for FDs.

Proposition 14 *For a UBCQ $\mathcal{Q}$ with inequalities, $\mathcal{MPCDP}(\mathcal{Q})$ is NP-hard.*

Proof It is good enough to reduce globally-optimal repair checking to our contingency checking problem. So, consider an inconsistent instance D with respect to a set of denial constraint Σ, a priority relation for repairs $\succ$, and $D' \subseteq D$. To check if $D' \in Rep^{\succ, go}(D, \Sigma)$ we can check, for an arbitrary element $t \in D \smallsetminus D'$, if $(D, \succ^r, t, D \smallsetminus (D' \cup \{t\})) \in \mathcal{MPCDP}(V^{\Sigma})$. □

It is worth contrasting this result with the tractability result in Proposition 11 for the *minimal contingency set decision problem* (MCSDP) for actual causes. Notice that Proposition 11 still holds for UBCQs with inequality.

Notice that we could generalize the notion of preferred cause by appealing to any notion of repair. More precisely, if we have a *repair semantics* rSem (based on tuple deletions for DCs), we could replace $Rep^{\succ, \mathcal{X}}(D, \kappa(\mathcal{Q}))$ in (22) by $Rep^{\mathsf{S}}(D, \kappa(\mathcal{Q}))$. However, to obtain the intended results for causes, we have to be careful, as above, about a possible inverse relationship between preference on repairs and preference on causes.

7.3 Endogenous Repairs

The partition of a database into endogenous and exogenous tuples that is used in the causality setting may also be of interest in the context of repairs. Considering that we should have more control on endogenous tuples than on exogenous ones, which may come from external sources, it makes sense to consider *endogenous repairs*, which would be obtained by updates (of any kind) on endogenous tuples only. (Of course, a symmetric treatment of "exogenous" repairs is also possible; what is relevant here is the partition.)

For example, in the case of DCs, endogenous repairs would be obtained by deleting endogenous tuples only. More formally, given $D = D^n \cup D^x$, possibly inconsistent with a set of DCs Σ, an *endogenous repair* D' of D is a maximally consistent sub-instance of D with $D \smallsetminus D' \subseteq D^n$, i.e. D' keeps all the exogenous tuples of D. If endogenous repairs form the class $Srep^n(D, \Sigma)$, it holds $Srep^n(D, \Sigma) \subseteq Srep(D, \Sigma)$.

Example 19 Consider $D = D^n \cup D^x$, with $D^n = \{R(a_2, a_1), R(a_4, a_3), S(a_3), S(a_4)\}$ and $D^x = \{R(a_3, a_3), S(a_2)\}$, and the DC $\kappa: \neg\exists xy(S(x) \wedge R(x, y) \wedge S(y))$.

Here, $Srep(D, \kappa) = \{D_1, D_2, D_3\}$, with $D_1 = \{R(a_2, a_1), R(a_4, a_3), R(a_3, a_3), S(a_4), S(a_2)\}$, $D_2 = \{R(a_2, a_1), S(a_3), S(a_4), S(a_2)\}$, and $D_3 = \{R(a_2, a_1), R(a_4, a_3), S(a_3), S(a_2)\}$. The only endogenous S-repair is D_1.

In this section, without trying to be exhaustive or detailed, we consider the possibility of defining endogenous repairs on the basis of a suitable priority relation $\succ$ on tuples,[21] while at the same time taking advantage of the *op* optimality condition considered in Section 7.1.[22]

First, if we assume that relation $\succ'$, the extension of $\succ$, is such, that $t \succ' t'$ when $t \in D^x$ and $t' \in D^n$ ($\succ'$ is $\succ$ if the latter already has this property), then it is easy to verify that every endogenous S-repair globally improves any non-endogenous S-repair. As a consequence, if there is an endogenous S-repair, then all the $(\succ', go)$-repairs are endogenous. Notice that the extension $\succ'$ may destroy the acyclicity assumption on the priority relation, because we are starting from a given (acyclic) relation $\succ$, which we are now extending.

It might be the case that there is no endogenous S-repair, in which case non-endogenous S-repairs would not the improved by an endogenous one. So, if we want to prevent the existence of non-endogenous repairs, we can add an extra, dummy predicate $D(\cdot)$ to the schema, and the endogenous tuple $D(d)$ to D. We modify every DC in Σ, say $\kappa : \leftarrow \ C(\bar{x})$, by adding an extra, dummy condition: $\kappa^d : \leftarrow \ D(d), C(\bar{x})$, obtaining a set Σ^d of DCs. In this case, the S-repairs will be: $D^d := D \setminus \{D(d)\}$, which is endogenous, and also all those S-repairs of D with respect to Σ (now each including $D(d)$). The latter are all non-endogenous. If we assume that $t \succ' D(d)$, for every $t \in D^x$, then every non-endogenous S-repair will be improved by D^d, and will not be considered.

7.4 Null-based Causes

Consider an instance $D = \{R(c_1, \ldots, c_n), \ldots\}$ that may be inconsistent with respect to a set of DCs. The allowed repair updates are changes of attribute values by the constant *null*. We assume that *null* does not join with any other value, including *null* itself.

In order to keep track of changes, we may introduce numbers as first arguments in tuples, as global tuple identifiers (ids). So, D becomes $D = \{R(1; c_1, \ldots, c_n), \ldots\}$. Assume that $id(t)$ returns the id of the tuple $t \in D$. For example, $id(R(1; c_1, \ldots, c_n)) = 1$.

If, by updating D into D' in this way, the value of the ith attribute in R is changed to *null*, then the change is captured as the string $R[1; i]$. These strings are collected forming the set $Diff^{null}(D, D')$. For example, if $D = \{R(1; a, b), S(2; c, d), S(3; e, f)\}$ is changed into $D' = \{R(1; a, null), S(2; null, d), S(3; null, null)\}$, we have $Diff^{null}(D, D') = \{R[1; 2], S[2; 1], S[3; 1], S[3; 2]\}$.

A *null*-repair of D with respect to a set of DCs Σ is a consistent instance D', such that $Diff^{null}(D, D')$ is minimal under set inclusion.[23] $Rep^{null}(D, \Sigma)$ denotes the class of null-based repairs of D with respect to Σ.

[21] Pairs of conflicting tuples would inherit the priority relationships from the general priority relation.

[22] Of course, we could use other optimality criteria at this points, but considering all possibilities is beyond the scope of this work.

[23] An alternative, but equivalent formulation can be found in [8].

Example 20 (example 19 cont.) Consider the following inconsistent instance with respect to DC $\kappa: \neg\exists xy(S(x) \wedge R(x, y) \wedge S(y))$:

$D = \{R(1; a_2, a_1), R(2; a_3, a_3), R(3; a_4, a_3), S(4; a_2), S(5; a_3), S(6; a_4)\}$.

For simplicity, we do not make any difference between endogenous and exogenous tuples. Here, the class of *null-based repairs*, $Rep^{null}(D, \kappa)$, is formed by:

$D_1 = \{R(1; a_2, a_1), R(2; a_3, a_3), R(3; a_4, a_3), S(4; a_2), S(5; null), S(6; a_4)\}$,
$D_2 = \{R(1; a_2, a_1), R(2; null, a_3), R(3; a_4, null), S(4; a_2), S(5; a_3), S(6; a_4)\}$,
$D_3 = \{R(1; a_2, a_1), R(2; null, a_3), R(3; a_4, a_3), S(4; a_2), S(5; a_3), S(6; null)\}$,
$D_4 = \{R(1; a_2, a_1), R(2; a_3, null), R(3; a_4, null), S(4; a_2), S(5; a_3), S(6; a_4)\}$,
$D_5 = \{R(1; a_2, a_1), R(2; a_3, null), R(3; null, a_3), S(4; a_2), S(5; a_3), S(6; a_4)\}$,
$D_6 = \{R(1; a_2, a_1), R(2; a_3, null), R(3; a_4, a_3), S(4; a_2), S(5; a_3), S(6; null)\}$.

Here, $Diff^{null}(D, D_2) = \{R[2; 1], R[3; 2]\}$, and $Diff^{null}(D, D_3) = \{R[2; 1], S[6; 1]\}$.

According to the motivation provided at the beginning of this section, we can now define causes appealing to the class of null-based repairs of D. Since repair actions in this case, are attribute-value changes, causes can be defined at both the tuple and attribute levels. The same applies to the definition of responsibility (in this case generalizing Proposition 2).

Definition 15 For D an instance and $\mathcal{Q}$ a BCQ, and $t \in D$ be a tuple of the form $R(i; c_1, \ldots, c_n)$.

(a) $R[i; c_j]$ is a *null-based attribute-value cause* for $\mathcal{Q}$ if there is $D' \in Rep^{null}(D, \kappa(\mathcal{Q}))$ with $R[i; j] \in Diff^{null}(D, D')$.
(That is, the value c_j for attribute A_j in the tuple is a cause if it is changed into a null in some repair.)

(b) t is a *null-based tuple cause* for $\mathcal{Q}$ if some $R[i; c_j]$ is a *null-based attribute-value cause* for $\mathcal{Q}$.
(That is, the whole tuple is a cause if at least one of its attribute values is changed into a null in some repair.)

(c) The responsibility, $\rho^{t-null}(t)$, of t, a *null-based tuple cause* for $\mathcal{Q}$, is the inverse of $min\{|Diff^{null}(D, D')| : R[i; j] \in Diff^{null}(D, D')$, for some j, and $D' \in Rep^{null}(D, \kappa(\mathcal{Q}))\}$.

(d) The responsibility, $\rho^{a-null}(R[i; c_j])$, of $R[i; c_j]$, a *null-based attribute-value cause* for $\mathcal{Q}$, is the inverse of $min\{|Diff^{null}(D, D')| : R[i; j] \in Diff^{null}(D, D')$, and $D' \in Rep^{null}(D, \kappa(\mathcal{Q}))\}$.

In cases (c) and (d) we minimize over the number of changes in a repair that are made together with that of the candidate tuple/attribute-value to be a cause. In the case of a tuple cause, any change made in one of its attributes is considered in the minimization. For this reason, the minimum may be smaller than the one for a fixed attribute value change; and so the responsibility at the tuple level may be greater than that at the attribute level. More precisely, if $t = R(i; c_1, \ldots, c_n) \in D$, and $R[i; c_j])$ is a *null-based attribute-value cause*, then it holds $\rho^{a-null}(R[i; c_j]) \leq \rho^{t-null}(t)$.

Example 21 (ex. 20 cont.) Consider $R(2; a_3, a_3) \in D$. Its projection on its first (non-id) attribute, $R[2; a_3]$, is an attribute-level cause since $R[2; 1] \in \mathit{Diff}^{null}(D, D_2)$. Also $R[2; 1] \in \mathit{Diff}^{null}(D, D_3)$.

Since $|\mathit{Diff}^{null}(D, D_2)| = |\mathit{Diff}^{null}(D, D_3)| = 2$, it holds $\rho^{a-null}(R[2; 1]) = \frac{1}{2}$.

Clearly $R(2; a_3, a_3)$ is a *null-based tuple cause* for $\mathcal{Q}$, with $\rho^{t-null}(t) = \frac{1}{2}$.

Notice that the definition of tuple-level responsibility, i.e. case (c) in Definition 15, does not take into account that a same id, i, may appear several times in a $Diff^{null}(D, D')$. In order to do so, we could redefine the size of the latter by taking into account those multiplicities. For example, if we decrease the size of the *Diff* by one with every repetition of the id, the responsibility for a cause may (only) increase, which makes sense.

8 Discussion and Conclusions

Our work opens interesting research directions, some of which are briefly discussed below. They are matter of ongoing and future research.

8.1 Endogenous Repairs

As discussed in Section 7, the partition of a database into endogenous and exogenous tuples may also be of interest in the context of repairs. We may prefer endogenous repairs that change (delete in this case) only endogenous tuples. However, if there are no endogenous tuples, a preference condition could be imposed on repairs, keeping those that change exogenous tuples the least. This is something to explore.

As a further extension, it could be possible to assume that combinations of (only) exogenous tuples never violate the integrity constraints, which could be checked at upload time. In this sense, there would be a part of the database that is considered to be consistent, while the other is subject to possible repairs. For somehow related research, see [31].

Going a bit further, we could even consider the relations in the database with an extra, binary attribute, N, that is used to annotate if a tuple is endogenous or exogenous (it could be both), e.g. a tuple like $R(a, b, yes)$. integrity constraints could be annotated too, e.g. the "exogenous" version of DC κ, could be $\kappa^E : \leftarrow P(x, y, yes), R(y, z, yes)$, and could be assumed to be satisfied.

8.2 Objections to Causality

Causality as introduced by Halpern and Pearl in [32, 33], aka. HP-causality, is the basis for the notion of causality in [47]. HP-causality has been the object of some criticism [35], which is justified in some (more complex, non-relational) settings, specially due to the presence of different kinds of *logical variables* (or lack thereof). In our context the objections do not apply: variables just say that a certain tuple

belongs to the instance (or not); and for relational databases the closed-world assumption applies. In [35, 36], the definition of HP-causality is slightly modified. In our setting, this modified definition does not change actual causes or their properties.

8.3 Open Queries

We have limited our discussion to Boolean queries. It is possible to extend our work to consider conjunctive queries with free variables, e.g. $\mathcal{Q}(x) : \exists yz(R(x, y) \wedge S(y, z))$. In this case, a query answer would be of the form $\langle a \rangle$, for a a constant, and causes would be found for such an answer. In this case, the associated DC would be of the form $\kappa^{\langle a \rangle} : \leftarrow R(a, y), S(y, z)$, and the rest would be basically as above.

8.4 ASP Specification of Causes

S-repairs can be specified by means of *answer set programs* (ASPs) [3, 6], and C-repairs too, with the use of weak program constraints [3]. This should allow for the introduction of ASPs in the context of causality, for specification and reasoning. There are also ASP-based specifications of diagnosis [24] that could be brought into a more complete picture.

8.5 Causes and Functional Dependencies, and Beyond

Functional dependencies are DCs with conjunctive violation views with inequality, and are still monotonic. There is much research on repairs and consistent query answering for functional dependencies, and more complex integrity constraints [9]. In causality, mostly CQs without built-ins have been considered. The repair connection could be exploited to obtain more refined results for causality and CQs with inequality, and also other classes of queries, even non-monotonic ones, that correspond violation views for other kinds of integrity constraints. In a different, but related direction, causality for monotonic queries in the presence of integrity constraints has been investigated in [56].

8.6 View Updates and Abduction

Abduction [20, 23] is another form of model-based diagnosis, and is related to the subjects investigated in this work. The *view update problem*, about updating a database through views, is a classical problem in databases that has been treated through abduction [21, 37]. User knowledge imposed through view updates creates or reflects *uncertainty* about the base data, because alternative base instances may give an account of the intended view updates. The view update problem, specially in its particular form of *deletion propagation*, has been recently related in [41, 42] to causality as introduced in [47]. (Notice only tuple deletions are used with violation views and repairs associated to DCs.)

Database repairs are also related to the view update problem. Actually, *answer set programs* (ASP) for database repairs [6] implicity repair the database by updating

intentional, annotated predicates (cf. Section 8.4). Even more, in [8], in order to protect sensitive information, databases are explicitly and virtually "repaired" through secrecy views that specify the information that has to be kept secret. These are prioritized repairs that have been specified via ASPs. Abduction has been explicitly applied to database repairs [5].

The deep interrelations between causality, abductive reasoning, view updates and repairs are the objects of our ongoing research efforts [10, 57].

To conclude, let us emphasize that in this research we have unveiled and formalized some first interesting relationships between causality in databases, database repairs, and consistency-based diagnosis. These connections allow us to apply results and techniques developed for each of them to the others. This is particularly beneficial for causality in databases, where still a limited number of results and techniques have been obtained or developed.

The connections we established here inspired complexity results for causality, e.g. Theorems 2 and 3, and were used to prove them. We appealed to several non-trivial results found in [43] (and the proofs thereof found in [44]) about repairs and CQA. It is also the case that the well-established hitting-set approach to diagnosis inspired a similar approach to causal responsibility, which in its turn allowed us to obtain results about its fixed-parameter tractability. It is also the case that diagnostic reasoning, as a form of non-monotonic reasoning, can provide a solid foundation for causality in databases and query answer explanation, in general [16, 17].

In ongoing research we have established connections between query answer causality, abductive diagnosis and database updates through views [57]. It is interesting that several of these areas of data management and knowledge representation, including those considered in this work, fall under what has been called "reverse data management" tasks [45]. Our work establishes formal connections between them and sets the ground for further investigation into their interrelationships.

Acknowledgments Research funded by NSERC Discovery, and the NSERC Strategic Network on Business Intelligence (BIN). Conversations with Alexandra Meliou during Leo Bertossi's visit to U. of Washington in 2011 are much appreciated. He is also grateful to Dan Suciu and Wolfgang Gatterbauer for their hospitality. L. Bertossi is grateful to Benny Kimelfeld for stimulating conversations. Part of the research was developed by L. Bertossi during partial sabbatical stays at *LogicBlox* and *The Center for Semantic Web Research* (Chile). Their support is much appreciated. We appreciate the comments from the anonymous reviewers.

References

1. Afrati, F., Kolaitis, P.: Repair checking in inconsistent databases: Algorithms and complexity. Proc. ICDT, 31–41 (2009)
2. Arenas, M., Bertossi, L., Chomicki, J.: Consistent query answers in inconsistent databases. Proc. ACM PODS, 68–79 (1999)
3. Arenas, M., Bertossi, L., Chomicki, J.: Answer sets for consistent query answers. Theory Pract. Logic Programm. **3**(4–5), 393–424 (2003)
4. Arenas, M., Bertossi, L., Chomicki, J., He, X., Raghavan, V., Spinrad, J.: Scalar aggregation in inconsistent databases. Theor. Comput. Sci. **296**, 405–434 (2003)

5. Arieli, O., Denecker, M., Van Nuffelen, B., Bruynooghe, M.: Coherent integration of databases by abductive logic programming. J. Artif Intell. Res. **21**, 245–286 (2004)
6. Barcelo, P., Bertossi, L., Bravo, L.: Characterizing and computing semantically correct answers from databases with annotated logic and answer sets. in semantics of databases. In: Semantics of Databases, Springer LNCS 2582, pp. 1–27 (2003)
7. Bertossi, L.: Consistent query answering in databases. ACM SIGMOD Rec. **35**(2), 68–76 (2006)
8. Bertossi, L., Li, L.: Achieving data privacy through secrecy views and null-based virtual updates. IEEE Trans Knowl. Data Eng. **25**(5), 987–1000 (2013)
9. Bertossi, L.: Database repairing and consistent query answering, Morgan & Claypool, Synthesis Lectures on Data Management (2011)
10. Bertossi, L., Salimi, B.: Unifying causality, diagnosis, repairs and view-updates in databases. Presented at the First International Workshop on Big Uncertain Data (BUDA 2014). Posted at: arXiv:1405.4228 [cs.DB]
11. Brankovic, L., Fernau, H.H.: Parameterized approximation algorithms for hitting set. In: Approximation and Online Algorithms, pp. 63–76. Springer LNCS 7164 (2012)
12. Bravo, L., Bertossi, L.: Semantically correct query answers in the presence of null values. In: Chomicki, J., Wijsen, J. (eds.) Proceedings EDBT WS on Inconsistency and Incompleteness in Databases (IIDB 06), pp. 336–357. Springer LNCS 4254 (2006)
13. Buneman, P., Khanna, S., Tan, W.C.: Why and where: A characterization of data provenance. Proc. ICDT, 316–330 (2001)
14. Buneman, P., Tan, W.C.: Provenance in databases. Proc. ACM SIGMOD, 1171–1173 (2007)
15. Cheney, J., Chiticariu, L., Tan, W.C.: Provenance in databases why, how, and where. Found. Trends Databases **1**(4), 379–474 (2009)
16. Cheney, J., Chong, S., Foster, N., Seltzer, M.I., Vansummeren, S.: Provenance a future history. OOPSLA Companion (Onward!), 957–964 (2009)
17. Cheney, J.: Is Provenance Logical? Proc. LID, 2–6 (2011)
18. Chomicki, J., Marcinkowski, J.: Minimal-change integrity maintenance using tuple deletions. Inf. Comput. **197**(1-2), 90–121 (2005)
19. Chockler, H., Halpern, J.Y.: Responsibility and blame: a structural-model approach. J. Artif. Intell. Res. **22**, 93–115 (2004)
20. Console, L., Torasso, P.: A spectrum of logical definitions of model-based diagnosis. Comput. Intell. **7**, 133–141 (1991)
21. Console, L., Sapino, M.L., Theseider-Dupre, D.: The role of abduction in database view updating. J. Intell. Inf. Syst. **4**(3), 261–280 (1995)
22. Cui, Y., Widom, J., Wiener, J.L.: Tracing the lineage of view data in a warehousing environment. ACM Trans. Database Syst. **25**(2), 179–227 (2000)
23. Eiter, T., Gottlob, G., Leone, N.: Abduction from logic programs semantics and complexity. Theor. Comput. Sci. **189**(1-2), 129–177 (1997)
24. Eiter, T.h., Faber, W., Leone, N., Pfeifer, G.: The diagnosis frontend of the DLV system. AI Commun. **12**(1-2), 99–111 (1999)
25. Fagin, R., Kimelfeld, B., Kolaitis, Ph.: Dichotomies in the complexity of preferred repairs. Proc. ACM PODS, 3–15 (2015)
26. Fernau, H.: Parameterized algorithmics for *d*-hitting set. Int. J. Comput. Math. **87**(14), 3157–3174 (2010)
27. Feldman, A., Provan, G., Gemund, A.V.: Approximate model-based diagnosis using greedy stochastic search. J. Artif. Intell. Res. (JAIR) **87**(14), 3157–3174 (2010)
28. Flum, J., Grohe, M.: Parameterized complexity theory. Texts in Theoretical Computer Science, Springer Verlag (2006)
29. Garey, M., Johnson, D.S.: Computers and intractability: a guide to the theory of NP-completenes. W. H. Freeman (1979)
30. Gertz, M.: Diagnosis and repair of constraint violations in database systems. PhD Thesis, Universität Hannover (1996)
31. Greco, S., Pijcke, F., Wijsen, J.: Certain query answering in partially consistent databases. PVLDB **7**(5), 353–364 (2014)
32. Halpern, J., Pearl, J.: Causes and explanations: a structural-model approach: part 1. Proc. UAI, 194–202 (2001)

33. Halpern, J., Pearl, J.: Causes and explanations: a structural-model approach: part 1. British J. Philos. Sci. **56**, 843–887 (2005)
34. Halperin, E.: Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. Proceedings ACM-SIAM Symposium on Discrete Algorithms, 329–337 (2000)
35. Halpern, J.: Appropriate causal models and stability of causation. Proc. KR'14 (2014)
36. Halpern, J.: A modification of Halpern-Pearl definition of causality. Proc. IJCAI (2015)
37. Kakas A. C., Mancarella, P.: Database updates through abduction. Proc. VLDB, 650–661 (1990)
38. Karvounarakis, G., Green, T.J.: Semiring-annotated data queries and provenance? SIGMOD Rec. **41**(3), 5–14 (2012)
39. Krentel, M.: The complexity of optimization problems. J. Comput. Syst. **36**, 490–509 (1988)
40. Karvounarakis, G., Ives Z. G., Tannen, V.: Querying provenance. Proc. ACM SIGMOD, 951–962 (2010)
41. Kimelfeld, B.: A dichotomy in the complexity of deletion propagation with functional dependencies. Proc. ACM PODS (2012)
42. Kimelfeld, B., Vondrak, J., Williams, R.: Maximizing conjunctive views in deletion propagation. ACM Trans. Database Syst. **37**(4), 24 (2012)
43. Lopatenko, A., Bertossi, L.: Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. Proc. ICDT, 2007, Springer LNCS 4353, pp. 179–193. Proofs of results are found in [44]
44. Lopatenko, A., Bertossi, L.: Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. Extended version of [43], including proofs. Posted at: arXiv:cs/1605.07159 [cs.DB]
45. Meliou, A., Gatterbauer, W., Suciu, D.: Reverse data management. PVLDB **4**(12), 1490–1493 (2011)
46. Meliou, A., Gatterbauer, W., Suciu, D.: Bringing provenance to its full potential using causal reasoning. Proc. TaPP (2011)
47. Meliou, A., Gatterbauer, W., Moore K. F., Suciu, D.: The complexity of causality and responsibility for query answers and non-answers. Proc. VLDB, 34–41 (2010)
48. Meliou, A., Gatterbauer, W., Halpern, J.Y., Koch, C., Moore K. F., Suciu, D.: Causality in databases. IEEE Data Eng. Bull. **33**(3), 59–67 (2010)
49. Mozetic, I., Holzbaur, C.: Controlling the complexity in model-based diagnosis. Ann. Math. Artif. Intell. **11**(1-4), 297–314 (1994)
50. Niedermeier, R., Rossmanith, P.: An efficient fixed-parameter algorithm for 3-hitting set. J Discret. Algorithm. **1**(1), 89–102 (2003)
51. Okun, M.: On approximation of the vertex cover problem in hypergraphs. Discret. Optim. **2**(1), 101–111 (2005)
52. Papadimitriou, C.H.: Computational complexity. Addison-Wesley (1994)
53. Reiter, R.: A theory of diagnosis from first principles. Artif. Intell. **32**(1), 57–95 (1987)
54. Reiter, R.: Towards a logical reconstruction of relational database theory. In: On Conceptual Modelling, pp. 191–233. Springer (1984)
55. Salimi, B., Bertossi, L.: Causality in databases: the diagnosis and repair connections. Presented at The 15th International Workshop on Non-Monotonic Reasoning (NMR 2014). Posted at: arXiv:1404.6857[cs.DB]
56. Salimi, B., Bertossi, L.: Causes for query answers from databases, datalog abduction and view-updates: the presence of integrity constraints. Proc. FLAIRS, 2016. Posted as Corr arXiv:1602.06458
57. Salimi, B., Bertossi, L.: Query-answer causality in databases: abductive diagnosis and view-updates. In: Proceedings UAI Causal Inference Workshop, 2015. CEUR-WS Proceedings Vol-1504 (2015)
58. Salimi, B., Bertossi, L.: From causes for database queries to repairs and model-based diagnosis and back. In: Proceedings 18th International Conference on Database Theory (ICDT 2015)
59. Staworko, S., Chomicki, J., Marcinkowski, J.: Prioritized repairing and consistent query answering in relational databases. Ann. Math. Artif. Intell. **64**(2-3), 209–246 (2012)
60. Struss, P.: Model-based problem solving. In: Handbook of Knowledge Representation, chap. 10. Elsevier (2008)
61. Tannen, V.: Provenance propagation in complex queries. In: Buneman Festschrift, 2013, Springer LNCS 8000, pp. 483–493