Causes for Query Answers from Databases, Datalog Abduction and View-Updates: The Presence of Integrity Constraints
(Extended Version)

Babak Salimi and **Leopoldo Bertossi**

Carleton University, School of Computer Science
Ottawa, Canada.
bsalimi@cs.washington.edu bertossi@scs.carleton.ca

Abstract

Causality has been recently introduced in databases, to model, characterize and possibly compute causes for query results (answers). Connections between query-answer causality, consistency-based diagnosis, database repairs (wrt. integrity constraint violations), abductive diagnosis and the view-update problem have been established. In this work we further investigate connections between query-answer causality and abductive diagnosis and the view-update problem. In this context, we also define and investigate the notion of query-answer causality in the presence of integrity constraints.

Keywords: Causality in databases, view updates, delete propagation, abduction, integrity constraints

2010 MSC: 00-01, 99-00

1. Introduction

Causality is a deep subject that appears at the foundations of many scientific disciplines; and also something we want to represent and compute to deal with *uncertainty* of data, information and theories. In data management in particular, there is a need to represent, characterize and compute causes that explain why certain query results are obtained or not, or why natural semantic

conditions, such as integrity constraints, are not satisfied. Causality can also be used to explain the contents of a view, i.e. of a predicate with virtual contents that is defined in terms of other physical, materialized relations (tables).

Most of the work on causality by the computer science community has been done in the context of knowledge representation, but little has been said about causality in data management. This work is about causality as defined for queries posed to relational databases.

The notion of causality-based explanation for a query result was introduced in [18], on the basis of the deeper concepts of *counterfactual* and *actual causation*. This approach can be traced back to [13]. We will refer to this notion as *query-answer causality* (or simply, *QA-causality*). Under this approach, explanations for query answers are provided in terms causes for query answers; and these causes are ranked according to their *degree of responsibility*, which quantifies the extent by which a QA-cause contributes to an answer. In [19], *view-conditioned causality* (vc-causality) was proposed as a restricted form of QA-causality, to determine causes for unexpected query results, but conditioned to the correctness of prior knowledge that cannot be altered by counterfactual tuple-deletions.

In [25], connections were established between QA-causality and *database repairs* [3], which allowed to obtain several complexity results for QA-causality related problems. A connection between QA-causality and *consistency-based diagnosis* [21] was established in [25], characterizing causes and responsibilities in terms of diagnoses, and leading to new results for QA-causality. In [26] connections between QA-causality and *abductive diagnosis* [8, 10] were presented.

The definition of QA-causality applies to monotone queries [18], but all complexity and algorithmic results in [18, 25] have been for first-order monotone queries, mainly conjunctive queries. However, QA-causality can be applied to Datalog queries [1], which are also monotone, but may contain recursion. Oh the other hand, abductive diagnosis can be done on top of Datalog specifications, leading to Datalog-abduction, for which there are known complexity results [10]. Actually, in [26] computational and complexity results were obtained for

Datalog QA-causality from a connection with Datalog-abduction. In this work we further exploit this connection to obtain new complexity results for Datalog QA-causality.

In [26], connections are reported between QA-causality and the classical *view-update problem* in databases, which is about updating a database through views [1]. One wants the base relations (also called "the source database") to change in a minimal way while still producing the view updates. When only deletions are performed on monotone views, we have the *delete-propagation problem*, from views to base relations [4, 16, 17]. This is the one considered in this work.

In [26], several connections between QA-causality and the delete-propagation problem were established and used to obtain new results for the former. In this work we obtain new results for *view-conditioned causality* from this connection.

We define and investigate the notion of query-answer causality in the presence of integrity constraints. The latter are logical dependencies between database tuples, and under the assumption that the instance at hand satisfies them, they should have an effect on determining the causes for a query answer. We show that they do, and propose a notion of cause that takes them into account.

This paper is an extended version of [24].

2. Preliminaries

We consider relational database schemas of the form $\mathcal{S} = (U, \mathcal{P})$, where U is the possibly infinite data domain and $\mathcal{P}$ is a finite set of *database predicates* of fixed arities. We may also have built-in predicates, e.g. $\neq$, that we leave implicit. The schema can be used to define a language $\mathcal{L}(\mathcal{S})$ of the first-order predicate logic. A database instance D compatible with $\mathcal{S}$ is a finite set of ground atomic formulas (a.k.a. atoms or tuples) of the form $P(c_1, ..., c_n)$, and $P \in \mathcal{P}$ is not a built-in.

A *conjunctive query* (CQ) is a formula of $\mathcal{L}(\mathcal{S})$ of the form $\mathcal{Q}(\bar{x}) : \exists \bar{y}(P_1(\bar{s}_1) \wedge \cdots \wedge P_m(\bar{s}_m))$, where the $P_i(\bar{s}_i)$ are atomic formulas, i.e. each $P_i \in \mathcal{P}$ or is a

built-in, and the $\bar{s}_i$ are sequences of terms, i.e. variables or constants of U. The $\bar{x}$ in $\mathcal{Q}(\bar{x})$ shows all the free variables in the formula, i.e. those not appearing in $\bar{y}$. A sequence $\bar{c}$ of constants is an answer to query $\mathcal{Q}(\bar{x})$ if $D \models \mathcal{Q}[\bar{c}]$, i.e. the query becomes true in D when the variables are replaced by the corresponding constants in $\bar{c}$. We denote the set of all answers to a query $\mathcal{Q}(\bar{x})$ with $\mathcal{Q}(D)$. A conjunctive query is *Boolean* (a BCQ), if $\bar{x}$ is empty, i.e. the query is a sentence, in which case, it is true or false in D, denoted by $D \models \mathcal{Q}$ (or $\mathcal{Q}(D) = \{true\}$) and $D \not\models \mathcal{Q}$ (or $\mathcal{Q}(D) = \{false\}$), respectively.

A query $\mathcal{Q}$ is *monotone* if for every two instances $D_1 \subseteq D_2$, $\mathcal{Q}(D_1) \subseteq \mathcal{Q}(D_2)$. CQs and unions of CQs (UCQs) are monotone queries. Datalog queries [1], although not first-order, are also monotone. *In this work we consider only monotone queries.*

An *integrity constraint* (IC) is a sentence $\varphi \in \mathcal{L}(\mathcal{S})$. Then, given an instance D for schema $\mathcal{S}$, it may be true or false in D (denoted $D \models \varphi$, resp. $D \not\models \varphi$). Given a set Σ of ICs, a database instance D is *consistent* if $D \models \Sigma$; otherwise it is said to be *inconsistent*. In this work we assume that sets of ICs are always finite and logically consistent.

A particular class of ICs is formed by *inclusion dependencies* (INDs), which are sentences of the form $\forall \bar{x}(P_i(\bar{x}) \rightarrow \exists \bar{y} P_j(\bar{x}', \bar{y}))$, with $\bar{x}' \cap \bar{y} = \emptyset, \bar{x}' \subseteq \bar{x}$. Another special class of ICs is formed by *functional dependencies* (FDs). For example, $\psi : \forall x \forall y \forall z(P(x, y) \wedge P(x, z) \rightarrow y = z)$ specifies that the second attribute of P functionally depends upon the first. Notice that it can be written as the negation of a BCQ: $\neg \exists x \exists y \exists z(P(x, y) \wedge P(x, z) \wedge y \neq z)$.

A Datalog query (DQ) $\mathcal{Q}(\bar{x})$ is a program Π, consisting of positive definite rules of the form $P(\bar{t}) \leftarrow P_1(\bar{t}_1), \ldots, P_n(\bar{t}_n)$, with the $P_i(\bar{t}_i)$ atomic formulas, that accesses an underlying extensional database D (the facts). In particular, Π defines an answer-collecting predicate $Ans(\bar{x})$ by means of a top rule of the form $Ans(\bar{x}) \leftarrow P_1(\bar{s}_1), \ldots, P_m(\bar{s}_m)$, where the P_i on the RHS are defined by other rules in Π or are database predicates for D. Here, the $\bar{s}_i$ are lists of variables or constants, and $\bar{x} \subseteq \bigcup_i \bar{s}_1$.

When $\Pi \cup D \models Ans(\bar{a})$, $\bar{a}$ is an answer to query Π on D. Here, $\models$ means

that the RHS belongs to the minimal model of $\Pi \cup D$. The Datalog query is Boolean (a BDQ) if the top answer-predicate is propositional, with a definition of the form $ans \leftarrow P_1(\bar{s}_1), \ldots, P_m(\bar{s}_m)$ [1]. CQs can be expressed as DQs.

3. QA-Causality and its Decision Problems

Following [18], in the rest of this work, unless otherwise stated, we assume that a relational database instance D is split in two disjoint sets, $D = D^n \cup D^x$, where D^n and D^x are the sets of *endogenous* and *exogenous* tuples, respectively. The former are admissible, interesting potential causes for query answers; but not the latter. *In the rest of this work, whenever a database instance is not explicitly partitioned, we assume all tuples are endogenous.*

A tuple $\tau \in D^n$ is a *counterfactual cause* for an answer $\bar{a}$ to $\mathcal{Q}(\bar{x})$ in D if $D \models \mathcal{Q}(\bar{a})$, but $D \smallsetminus \{\tau\} \not\models \mathcal{Q}(\bar{a})$. A tuple $\tau \in D^n$ is an *actual cause* for $\bar{a}$ if there exists $\Gamma \subseteq D^n$, called a *contingency set*, such that τ is a counterfactual cause for $\bar{a}$ in $D \smallsetminus \Gamma$. (In particular, this requires $D \smallsetminus \Gamma \models \mathcal{Q}(\bar{a})$.) $Causes(D, \mathcal{Q}(\bar{a}))$ denotes the set of actual causes for $\bar{a}$. If $\mathcal{Q}$ is Boolean, $Causes(D, \mathcal{Q})$ contains the causes for answer *true*. We collect all minimal contingency sets associated with $\tau \in D^n$: $Cont(D, \mathcal{Q}(\bar{a}), \tau) := \{\Gamma \subseteq D^n \mid D \smallsetminus \Gamma \models Q(\bar{a}), D \smallsetminus (\Gamma \cup \{\tau\}) \not\models \mathcal{Q}(\bar{a}),$ and for all $\Gamma' \subsetneq \Gamma,\ D \smallsetminus (\Gamma' \cup \{\tau\}) \models \mathcal{Q}(\bar{a})\}$.

The *causal responsibility* of a tuple τ for answer $\bar{a}$ is $\rho_{\mathcal{Q}(\bar{a})}(\tau) := \frac{1}{(|\Gamma|+1)}$, where $|\Gamma|$ is the size of the smallest contingency set for τ. When τ is not an actual cause for $\bar{a}$, $\rho_{\mathcal{Q}(\bar{a})}(\tau) := 0$.

The definition of QA-causality can be applied to DQs, in which case we denote with $Causes(D, \Pi(\bar{a}))$ the set of causes for answer $\bar{a}$.

Example 1. Consider the instance D with a single binary relation E as below (t_1-t_7 are tuple identifiers). Assume all tuples are endogenous.

E	A	B
t_1	a	b
t_2	b	e
t_3	e	d
t_4	d	b
t_5	c	a
t_6	c	b
t_7	c	d

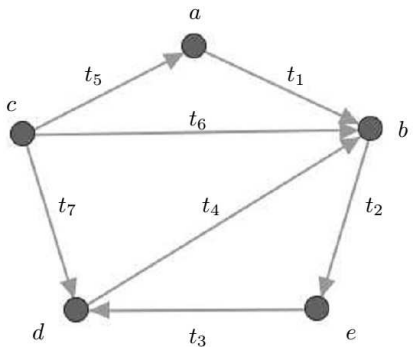

Figure 1: Graph G for database D in Example 1.

Instance D can be represented as the directed graph $G(\mathcal{V}, \mathcal{E})$ in Figure 1, where $\mathcal{V}$ coincides with the active domain of D (i.e. the set of constants in E), and $\mathcal{E}$ contains an edge (v_1, v_2) iff $E(v_1, v_2) \in D$. The tuple identifiers are used as labels for the corresponding edges in the graph. For simplicity, we will refer to the database tuples through their identifiers. Consider the DQ Π that collects pairs of vertices of G that are connected through a path, and is formed by the rules: $Ans(x, y) \leftarrow P(x, y)$. $P(x, y) \leftarrow E(x, y)$. and $P(x, y) \leftarrow P(x, z), E(z, y)$.

It is easy to see that, $\langle c, e \rangle$ is an answer to query Π on D. That is, $\Pi \cup D \models Ans(c, e)$. This is because there are three distinct paths between c and e in G. All tuples except for t_3 are actual causes for this answer, i.e., $Causes(E, \Pi(c, e)) = \{t_1, t_2, t_4, t_5, t_6, t_7\}$. Intuitively, this is because all of these tuples contribute to at least one path between c and e. Among them, t_2 has the highest responsibility, because, t_2 is a counterfactual cause for the answer, i.e. it has an empty contingency set. ■

The complexity of the computational and decision problems that arise in QA-causality have been investigated in [18, 25]. For a Boolean monotone query $\mathcal{Q}$, the *causality decision problem* (*CDP*) is (deciding about membership of):

$$\mathcal{CDP}(\mathcal{Q}) := \{(D, \tau) \mid \tau \in D^n, \text{and } \tau \in Causes(D, \mathcal{Q})\}.$$

It is tractable for *UCQ*s [25].

For a Boolean monotone query $\mathcal{Q}$, the *responsibility decision problem* (*RDP*)

is (deciding about membership of):

$$\mathcal{RDP}(\mathcal{Q}) \;=\; \{(D,\tau,v) \mid \tau \in D^n, v \in \{0\} \cup \{\tfrac{1}{k} \mid k \in \mathbb{N}^+\},\, D \models \mathcal{Q} \text{ and } \rho_{\mathcal{Q}}(\tau) > v\}.$$

It is *NP*-complete for UCQs [25].[1]

3.1. View-conditioned causality

Consider a query $\mathcal{Q}$ with $\mathcal{Q}(D) = \{\bar{a}_1, \ldots, \bar{a}_n\}$. Fix an answer, say $\bar{a}_1 \in \mathcal{Q}(D)$, while the other answers will be used as a condition on $\bar{a}_1$'s causality. Intuitively, $\bar{a}_1$ is somehow unexpected, we look for causes, but considering the other answers as "correct", which has the effect of reducing the spectrum of contingency sets, by keeping $\mathcal{Q}(D)$'s extension fixed, as a *view extension*, modulo $\bar{a}_1$ [19]. More precisely, if $\mathcal{Q}(D) = \{\bar{a}_1, \ldots, \bar{a}_n\}$, and $V := \mathcal{Q}(D) \setminus \{\bar{a}_1\}$:

(a) Tuple $\tau \in D^n$ is a *view-conditioned counterfactual cause* (vcc-cause) *for* $\bar{a}_1$ *in* D *relative to* V if $\bar{a}_1 \notin \mathcal{Q}(D \setminus \{\tau\})$ but $\mathcal{Q}(D \setminus \{\tau\}) = V$.

(b) Tuple $\tau \in D^n$ is a *view-conditioned actual cause* (vc-cause) *for* $\bar{a}_1$ *in* D *relative to* V if there exists a contingency set, $\Gamma \subseteq D^n$, such that τ is a vcc-cause for $\bar{a}_1$ in $D \setminus \Gamma$ relative to V.

Notice that the conditions on vc-causality with Γ are: $\bar{a}_1 \in \mathcal{Q}(D \setminus \Gamma)$, $\bar{a}_1 \notin (\mathcal{D} \setminus (\Gamma \cup \{\tau\}))$, and $\mathcal{Q}(D \setminus (\Gamma \cup \{\tau\})) = V$. In the following, we will omit saying "relative to V" since the fixed contents can be understood from the context.

(c) *vc-Causes*$(D, \mathcal{Q}(\bar{a}_1))$ denotes the set of all vc-causes for $\bar{a}_1$.

(d) The *vc-causal responsibility* of a tuple τ for answer $\bar{a}_1$ is *vc-*$\rho_{\mathcal{Q}(\bar{a}_1)}(\tau) := \frac{1}{1+|\Gamma|}$, where $|\Gamma|$ is the size of the smallest contingency set that makes τ a vc-cause for $\bar{a}_1$.

Clearly, *vc-Causes*$(D, Q(\bar{a})) \subseteq$ *Causes*$(D, Q(\bar{a}))$, but not necessarily the other way around.

[1] All the results are in data complexity.

Definition 1. (a) The *vc-causality decision problem* (VCDP) is about membership of $\mathcal{VCDP}(\mathcal{Q}) = \{(D, \bar{a}, \tau) \mid \bar{a} \in \mathcal{Q}(D)$ and $\tau \in$ *vc-Causes*$(D, \mathcal{Q}(\bar{a}))\ \}$.
(b) The *vc-causal responsibility decision problem* is about membership of:

$$\mathcal{VRDP}(\mathcal{Q}) \ = \ \{(D, \bar{a}, \tau, v) \mid \tau \in D^n, v \in \{0\} \ \cup \{\tfrac{1}{k} \mid k \in \mathbb{N}^+\}, D \models \mathcal{Q}(\bar{a}),$$
$$\text{and} \quad \textit{vc-}\rho_{\mathcal{Q}}(\tau) > v\}. \qquad \blacksquare$$

Since leaving the other answers fixed is a strong condition, it makes sense to study the complexity of deciding whether a query answer has a vc-cause or not.

Definition 2. For a monotone query $\mathcal{Q}$, the *vc-cause existence problem* is (deciding about membership of):

$$\mathcal{VCEP}(\mathcal{Q}) \ = \ \{(D, \bar{a}) \mid \bar{a} \in \mathcal{Q}(D) \text{ and } \ \textit{vc-Causes}(D, \mathcal{Q}(\bar{a})) \neq \emptyset\ \}. \qquad \blacksquare$$

Example 2. Consider relational predicates *GroupUser*(*User*, *Group*) and *GroupFile*(*File*, *Group*), with extensions as in instance D below. They represent users' memberships of groups, and access permissions for groups to files, respectively.

GroupUser	*User*	*Group*
	Joe	g_1
	Joe	g_2
	John	g_1
	Tom	g_2
	Tom	g_3
	John	g_3

GroupFiles	*File*	*Group*
	f_1	g_1
	f_1	g_3
	f_2	g_2
	f_3	g_3

It is expected that a user u can access the file f if u belongs to a group that can access f, i.e. there is some *Group* such that *GroupUser*(*User*, *Group*) and *GroupFile*(*File*, *Group*).

The following Datalog query collects users with the files they can access:

$$\textit{Access}(\textit{User}, \textit{File}) \ \leftarrow \ \textit{GroupUser}(\textit{User}, \textit{Group}), \textit{GroupFile}(\textit{File}, \textit{Group}). \quad (1)$$

Query *Access* in (1) has the following answers:

$Access(D)$	$User$	$File$
	Joe	f_1
	Joe	f_2
	Tom	f_1
	Tom	f_2
	Tom	f_3
	$John$	f_1
	$John$	f_3

Suppose the access of Joe to the file, f_1 -corresponding to the query answer $\langle Joe, f_1\rangle$- is deemed to be unauthorized, while all other users' accesses are authorized, i.e. the rest of the answers to the query are considered to be correct.

First, $GroupUser(Joe, g_1)$ is a counterfactual cause for answer $\langle Joe, f_1\rangle$, and also an actual cause, with empty contingency set.

But now we are interested in causes for the answer $\langle Joe, f_1\rangle$, while keeping all the other answers untouched.

$GroupUser(Joe, g_1)$ is also a vcc-cause. In fact:

$$Access(D \smallsetminus \{GroupUser(Joe, g_1)\}) = Access(D) \smallsetminus \{\langle Joe, f_1\rangle\},$$

showing that after the removal of $GroupUser(Joe, g_1)$, all the other previous answers remain. So, $GroupUser(Joe, g_1)$ is also a vc-cause with empty contingency set.

$GroupFile(f_1, g_1)$ is also an actual cause for $\langle Joe, f_1\rangle$, actually a counterfactual cause. However, it is not a vcc-cause, because its removal leads to the elimination of the previous answer $\langle John, f_1\rangle$. Even less can it be a vc-cause, because deleting a non-empty contingency set together with $GroupFile(f_1, g_1)$ can only make things worse: answer $\langle John, f_1\rangle$ would still be lost.

Actually, $GroupUser(Joe, g_1)$ is the only vc-cause and the only vcc-cause for $\langle Joe, f_1\rangle$.

Let us assume that, instead of D, we have instance D', with extensions:

GroupUser'	*User*	*Group*
	Joe	g_0
	Joe	g_1
	Joe	g_2
	John	g_1
	Tom	g_2
	Tom	g_3
	John	g_3

GroupFiles'	*File*	*Group*
	f_1	g_0
	f_1	g_1
	f_1	g_3
	f_2	g_2
	f_3	g_3

The answers to the query are the same as with D, in particular, we still have $\langle Joe, f_1\rangle$ as an answer to the query.

Now, $GroupUser(Joe, g_1)$ is not a counterfactual cause for $\langle Joe, f_1\rangle$ anymore, since this answer can still be obtained via the tuples involving g_0. However, $GroupUser(Joe, g_1)$ is an actual cause, with minimal contingency sets: $\Gamma_1 = \{GroupUsers(Joe, g_0)\}$ and $\Gamma_2 = \{GroupFiles(f_1, g_0)\}$.

Now, $GroupUser(Joe, g_1)$ is not a vcc-cause, but it is a vc-cause, with minimal contingence sets Γ_1 and Γ_2 as above: Removing Γ_1 or Γ_2 from D' keeps $\langle Joe, f_1\rangle$ as an answer. However, both under $D' \setminus (\Gamma_1 \cup \{GroupUser(Joe, g_1)\})$ and $D' \setminus (\Gamma_2 \cup \{GroupUser(Joe, g_1)\})$ the answer $\langle Joe, f_1\rangle$ is lost, but the other answers stay. ■

4. Causality and Abduction

An abductive explanation for an observation is a formula that, together with a background logical theory (a system description), entails the observation. In database causality we do not have an explicit system description, but just a set of tuples. Something like a system description emerges with a query, and causal relationships between tuples are captured by the combination of atoms in it. With a DQ , we have a specification in terms of positive definite rules.

A *Datalog abduction problem* [10] is of the form $\mathcal{AP} = \langle \Pi, E, Hyp, Obs\rangle$, where: (a) Π is a set of Datalog rules, (b) E is a set of ground atoms (the extensional database), (c) the hypothesis, Hyp, is a finite set of ground atoms,

the abducible atoms,[2] and (d) *Obs*, the observation, is a finite conjunction of ground atoms.

The *abduction problem* is about computing a subset-minimal $\Delta \subseteq Hyp$, such that $\Pi \cup E \cup \Delta \models Obs$. In this case, Δ is called an *abductive diagnosis*. So, no proper subset of Δ is an abductive diagnosis. $Sol(\mathcal{AP})$ denotes the set of abductive diagnoses for problem $\mathcal{AP}$. Now, a hypothesis $h \in Hyp$ is *relevant* for $\mathcal{AP}$ if h contained in at least one diagnosis of $\mathcal{AP}$, otherwise it is *irrelevant*. $Rel(\mathcal{AP})$ collects all relevant hypothesis for $\mathcal{AP}$. A hypothesis $h \in Hyp$ is *necessary* for $\mathcal{AP}$ if h contained in all diagnosis of $\mathcal{AP}$. $Ness(\mathcal{AP})$ collects all the necessary hypothesis for $\mathcal{AP}$.

The *relevance decision problem* (RLDP) is about deciding the membership of: $\mathcal{RLDP}(\Pi) = \{(E, Hyp, Obs, h) \mid h \in Rel(\mathcal{AP}), \text{with } \mathcal{AP} = \langle \Pi, E, Hyp, Obs \rangle\}$. The *necessity decision problem* (NDP) is about deciding the membership of: $\mathcal{NDP}(\Pi) = \{(E, Hyp, Obs, h) \mid h \in Ness(\mathcal{AP}), \text{with } \mathcal{AP} = \langle \Pi, E, Hyp, Obs \rangle\}$.

The following results can be obtained adapting results in [10, the. 26] and [11]: For every Datalog program Π, $\mathcal{NDP}(\Pi)$ is in *PTIME* (in data); and, for Datalog programs, Π, $\mathcal{RLDP}(\Pi)$ is *NP*-complete.

For a BDQ Π with $\Pi \cup D \models ans$, the causality decision problem takes the form: $\mathcal{CDP}(\Pi) := \{(D, \tau) \mid \tau \in D^n, \text{and } \tau \in Causes(D, \Pi)\}$. It turns out that, for Datalog system specifications, actual causes for *ans* can be obtained from abductive diagnoses of the associated *causal Datalog abduction problem* (CDAP): $\mathcal{AP}^c := \langle \Pi, D^x, D^n, ans \rangle$, where D^x takes the role of the extensional database for Π. Accordingly, $\Pi \cup D^x$ becomes the *background theory*, D^n becomes the set of *hypothesis*, and atom *ans* is the observation.

Proposition 1. For an instance $D = D^x \cup D^n$ and a BDQ Π, with $\Pi \cup D \models ans$, and its associated CDAP $\mathcal{AP}^c$, the following hold: (a) $\tau \in D^n$ is an counterfactual cause for *ans* iff $\tau \in Ness(\mathcal{AP}^c)$. (b) $\tau \in D^n$ is an actual cause for *ans* iff $\tau \in Rel(\mathcal{AP}^c)$. ■

[2] The hypothesis can be all the possible ground instantiations of *abducible predicates*, which do not appear in rule's LHSs.

Example 3. Consider the instance D with relations R and S as below, and the query Π: $ans \leftarrow R(x, y), S(y)$, which is true in D. Assume all tuples are endogenous.

R	A	B
	a_1	a_4
	a_2	a_1
	a_3	a_3

S	A
	a_1
	a_2
	a_3

Here, $\mathcal{AP}^c = \langle \Pi, \emptyset, D, ans \rangle$, which has two (minimal) abductive diagnoses: $\Delta_1 = \{S(a_1), R(a_2, a_1)\}$ and $\Delta_2 = \{S(a_3), R(a_3, a_3)\}$. Then, $Rel(\mathcal{AP}^c) = \{S(a_3), R(a_3, a_3), S(a_1), R(a_2, a_1)\}$. It is clear that the relevant hypothesis are actual causes for ans. ■

We can use the results mentioned above to obtain new complexity results for Datalog QA-causality. First, for the problem of deciding if a tuple is a *counterfactual cause* for a query answer. This is a tuple that, when removed from the database, undermines the query-answer, without having to remove other tuples, as is the case for *actual causes*. Actually, for each of the latter there may be an exponential number of contingency sets [25]. A counterfactual cause is an actual cause with responsibility 1. The complexity of this problem can be obtained from the connection between counterfactual causation and the necessity of hypothesis in Datalog abduction.

Proposition 2. For BDQs Π, $\mathcal{CFDP}(\Pi) := \{(D, \tau) \mid \tau \in D^n \text{ and } \rho_{\mathcal{Q}}(\tau) = 1\}$. is in *PTIME* (in data). ■

For BDQs Π, deciding actual causality, i.e. the problem $\mathcal{CDP}(\Pi)$, is *NP*-complete (in data) [26]. The same problem is tractable for UCQs [25]. Finally, we establish the complexity of the responsibility problem for DQs.

Proposition 3. For BDQs Π, $\mathcal{RDP}(\Pi)$ is *NP*-complete. ■

5. Causality and View-Updates

There is a close relationship between QA-causality and the view-update problem in the form of delete-propagation [1].

Let D be a database instance, and $\mathcal{Q}$ a monotone query. For $\bar{a} \in \mathcal{Q}(D)$, the *minimal-source-side-effect deletion-problem* is about computing a subset-minimal $\Lambda \subseteq D$, such that $\bar{a} \notin \mathcal{Q}(D \setminus \Lambda)$.

Now, following [4], let D be a database instance D, and $\mathcal{Q}$ a monotone query: (a) For $\bar{a} \in \mathcal{Q}(D)$, the *view-side-effect-free deletion-problem* is about computing a $\Lambda \subseteq D$, such that $\mathcal{Q}(D) \smallsetminus \{\bar{a}\} = \mathcal{Q}(D \smallsetminus \Lambda)$. (b) The *view-side-effect-free decision problem* is (deciding about the membership of): $\mathcal{VSEFP}(\mathcal{Q}) = \{(D, \bar{a}) \mid \bar{a} \in \mathcal{Q}(D),$ and exists $D' \subseteq D$ with $\mathcal{Q}(D) \smallsetminus \{\bar{a}\} = \mathcal{Q}(D')\}$. The latter decision problem is *NP*-complete for conjunctive queries [4, theorem 2.1].

Consider a relational instance D, a view $\mathcal{V}$ defined by a monotone query $\mathcal{Q}$. Then, the virtual view extension, $\mathcal{V}(D)$, is $\mathcal{Q}(D)$. For a tuple $\bar{a} \in \mathcal{Q}(D)$, the delete-propagation problem, in its most general form, is about deleting a set of tuples from D, and so obtaining a subinstance D' of D, such that $\bar{a} \notin \mathcal{Q}(D')$. It is natural to expect that the deletion of $\bar{a}$ from $\mathcal{Q}(D)$ can be achieved through deletions from D of actual causes for $\bar{a}$ (to be in the view extension). However, to obtain solutions to the different variants of this problem, different combinations of actual causes must be considered [26].

In particular, in [26], it has been shown that actual causes of $\bar{a}$ with their minimal contingency sets are in correspondence with the solutions to the minimal-source-side-effect deletion-problem of $\bar{a}$.

Now, in order to check if there exists a solution to the view-side-effect-free deletion-problem for $\bar{a} \in \mathcal{V}(D)$, it is good enough to check if $\bar{a}$ has a view-conditioned cause. Actually, it holds [26]: For an instance D, a view $\mathcal{V}$ defined by a monotone query $\mathcal{Q}$ with $\mathcal{Q}(D) = \{\bar{a}_1, \ldots, \bar{a}_n\}$, and $\bar{a_k} \in \mathcal{Q}(D)$, $(D, \bar{a}_k) \in \mathcal{VSEFP}(\mathcal{Q})$ iff *vc-Causes*$(D, \mathcal{Q}(\bar{a}_k)) \neq \emptyset$.

We now consider the complexity of the view-conditioned causality problem (cf. Definition 1). By appealing to the connection between vc-causality and delete-propagation, we obtain for the *vc-cause existence problem* (cf. Definition 2): For CQs $\mathcal{Q}$, $\mathcal{VCEP}(\mathcal{Q})$ is *NP*-complete (in data) [26]. A polynomial-time Turing (or Cook) reduction from this problem allows us to obtain the next result about deciding vc-causality (cf. Definition 1).

Proposition 4. For CQs $\mathcal{Q}$, $\mathcal{VCDP}(\mathcal{Q})$ is *NP*-complete. ■

By a (Karp) reduction from this problem, we settle the complexity of the

vc-causality responsibility problem for conjunctive queries.

Proposition 5. For CQs $\mathcal{Q}$, $\mathcal{VRDP}(\mathcal{Q})$ is *NP*-complete. ■

These results on vc-causality also hold for UCQs.

6. QA-Causality under Integrity Constraints

To motivate a definition of QA-causality in the presence of integrity constraints (ICs), we start with some remarks.

Interventions are at the base of Halpern & Pearl's approach to causality [13], i.e. actions on the model that define counterfactual scenarios. In databases, they take the form of tuple deletions. If a database D satisfies a prescribed set of integrity constraints (ICs), the instances obtained from D by tuple deletions, as used to determine causes, should be expected to satisfy the ICs.

On a different side, QA-causality in [18] is *insensitive* to equivalent query rewriting (as first pointed out in [12]): QA-causes coincide for logically equivalent queries. However, QA-causality might be sensitive to equivalent query rewritings in the presence of ICs, as the following example shows.

Example 4. Let $\mathcal{S} = \{Dep(DName, TStaff), Course(CName, LName, DName)\}$ be relational schema with inclusion dependency

$$I: \quad \forall x \forall y \ (Dep(x, y) \rightarrow \exists u \ Course(u, y, x));$$

and instance D for $\mathcal{S}$:

Dep	*DName*	*TStaff*
t_1	*Computing*	*John*
t_2	*Philosophy*	*Patrick*
t_3	*Math*	*Kevin*

Course	*CName*	*LName*	*DName*
t_4	*Com08*	*John*	*Computing*
t_5	*Math01*	*Kevin*	*Math*
t_6	*Hist02*	*Patrick*	*Philosophy*
t_7	*Math08*	*Eli*	*Math*
t_8	*Com01*	*John*	*Computing*

Clearly, $D \models I$. Now, consider the CQ that collects the teaching staff who are lecturing in the department they are associated with:

$$\mathcal{Q}(TStaff) \leftarrow Dep(DName, TStaff), Course(CName, TStaff, DName). \qquad (2)$$

The answers are: $\mathcal{Q}(D) = \{John, Patrick, Kevin\}$; and answer $\langle John \rangle$ has the following actual causes: t_1, t_4 and t_8. t_1 is a counterfactual cause, t_4 has a single minimal contingency set $\Gamma_1 = \{t_8\}$; and t_8 has a single minimal contingency set $\Gamma_2 = \{t_4\}$.

Now, in the presence of IC I, $\mathcal{Q}$ is equivalent with the following query $\mathcal{Q}'$: (denoted $\mathcal{Q} \equiv_{\{I\}} \mathcal{Q}'$, and meaning they give the same answers for every instance that satisfies I)

$$\mathcal{Q}'(TStaff) \quad \leftarrow \quad Dep(DName, TStaff)).$$

In particular, $\langle John \rangle$ is still an answer to $\mathcal{Q}'$ from D. However, on the basis of query $\mathcal{Q}'$ and instance D alone, there is single cause, t_1, which is also a counterfactual cause. ■

Definition 3. Given an instance $D = D^n \cup D^x$ that satisfies a set Σ of ICs, i.e. $D \models \Sigma$, and a monotone query $\mathcal{Q}$ with $D \models \mathcal{Q}(\bar{a})$, a tuple $\tau \in D^n$ is an *actual cause for* $\bar{a}$ *under* Σ if there is $\Gamma \subseteq D^n$, such that:

(a) $D \smallsetminus \Gamma \models \mathcal{Q}(\bar{a})$, and (b) $D \smallsetminus \Gamma \models \Sigma$.

(c) $D \smallsetminus (\Gamma \cup \{t\}) \not\models \mathcal{Q}(\bar{a})$, and (d) $D \smallsetminus (\Gamma \cup \{t\}) \models \Sigma$.

$Causes(D, \mathcal{Q}(\bar{a}), \Sigma)$ denotes the set of actual causes for $\bar{a}$ under Σ. ■

Example 5. (ex. 4 cont.) Consider answer $\langle John \rangle$ to $\mathcal{Q}$, for which t_4 was a cause with minimal contingency set $\Gamma_1 = \{t_8\}$. It holds $D \smallsetminus \Gamma_1 \models I$, but $D \smallsetminus (\Gamma_1 \cup \{t_2\}) \not\models I$. So, the new definition does not allow t_4 to be an actual cause for answer $\langle John \rangle$ to $\mathcal{Q}$. Actually, $\mathcal{Q}$ and $\mathcal{Q}'$ have the same actual causes for answer $\langle John \rangle$ under I, namely t_1. ■

Since functional dependencies (FDs) are never violated by tuple deletions, they have no effect on the set of causes for a query answer. Actually, this applies to all *denial constraints* (DCs), i.e. of the form $\neg\forall\bar{x}(A_1(\bar{x}_1) \wedge \cdots \wedge A_n(\bar{x}_n))$, with A_i a database predicate or a built-in.

Proposition 6. Given an instance D, a monotone query $\mathcal{Q}$, and a set of ICs Σ, the following hold:

(a) $Causes(D, \mathcal{Q}(\bar{a}), \Sigma) \subseteq Causes(D, \mathcal{Q}(\bar{a}))$.

(b) $Causes(D, \mathcal{Q}(\bar{a}), \emptyset) = Causes(D, \mathcal{Q}(\bar{a}))$.

(c) When Σ consists of DCs, $Causes(D, \mathcal{Q}(\bar{a}), \Sigma) = Causes(D, \mathcal{Q}(\bar{a}))$.

(d) For a monotone query $\mathcal{Q}'$ with $\mathcal{Q}' \equiv_\Sigma \mathcal{Q}$:

$$Causes(D, \mathcal{Q}(\bar{a}), \Sigma) = Causes(D, \mathcal{Q}'(\bar{a}), \Sigma).$$

(e) For a monotone query $\mathcal{Q}'$ which is minimally contained in $\mathcal{Q}$ with $\mathcal{Q}' \equiv_\Sigma \mathcal{Q}$:[3]
$Causes(D, \mathcal{Q}(\bar{a}), \Sigma) = Causes(D, \mathcal{Q}'(\bar{a}))$. ■

Notice that item (e) here relates to the rewriting of the query in Example 4. Notice that this rewriting resembles the resolution-based rewritings used in *semantic query optimization* [20].

Since FDs have no effect on causes, the causality decision problems in the presence of FDs have the same complexity upper bound as causality without FDs. For example, for Σ a set of FDs, $\mathcal{RDP}(\mathcal{Q}, \Sigma)$, the responsibility problem now under FDs, is *NP*-complete (as it was without ICs [25]). However, when an instance satisfies a set of FDs, the decision problems may become tractable depending on the query structure. For example, for the class of *key-preserving* CQs, deciding responsibility over instances that satisfy the key constraints (KCs) is in *PTIME* [5]. A KC is a particular kind of FD where some of the predicate attributes functionally determine *all* the others. Given a set κ of KCs, a CQ is key-preserving if, whenever an instance D satisfies κ, all key attributes of base relations involved in $\mathcal{Q}$ are included among the attributes of $\mathcal{Q}$.

By appealing to the connection between vc-causality and delete-propagation [26], vc-responsibility under KCs is tractable (being intractable in general, because the problem without KCs already is, as shown in Proposition 5):

Proposition 7. Given a set κ of KCs, and a key-preserving CQ query $\mathcal{Q}$, deciding $\mathcal{VRDP}(\mathcal{Q}, \kappa)$ is in *PTIME*. ■

[3] This means $\mathcal{Q}' \subseteq \mathcal{Q}$ and there is no $\mathcal{Q}''$ with $\mathcal{Q}'' \subsetneq \mathcal{Q}'$ and $\mathcal{Q}'' \equiv_\Sigma \mathcal{Q}$.

New subclasses of (view-defining) CQs for which different variants of delete-propagation are tractable are introduced in [16, 17] (generalizing those in [7]). The established connections between delete-propagation and causality should allow us to adopt them for the latter.

QA-causality under ICs can capture vc-causality:

Proposition 8. For a conjunctive query $\mathcal{Q}(\bar{x}) \in L(\mathcal{S})$, and an instance D for $\mathcal{S}$, with $\mathcal{Q}(D) = \{\bar{a}_1, \ldots, \bar{a}_n\}$ and a fixed $k \in \{1, \ldots, n\}$, there is a set of inclusion dependencies Σ over schema $\mathcal{S} \cup \{V\}$, with V a fresh $|\bar{x}|$-ary predicate, and an instance D' for $\mathcal{S} \cup \{V\}$, such that $\mathit{vc\text{-}Causes}(D, \mathcal{Q}(\bar{a}_k)) = \mathit{Causes}(D', \mathcal{Q}(\bar{a}), \Sigma)$.

Proof: Given D, consider the instance $D' := D \cup (\mathcal{Q}(D) \setminus \{\bar{a}_k\})$, where the second disjunct is the extension for predicate V. Now, consider the set of ICs for schema $\mathcal{S} \cup \{V\}$: $\Sigma := \{\forall \bar{x}(V(\bar{x}) \rightarrow \mathcal{Q}(\bar{x}))\}$. ■

Deciding causality in the absence of ICs is tractable, but their presence has an impact on this problem. The following is obtained from Propositions 4 and 8.

Proposition 9. For CQs $\mathcal{Q}$ and a set Σ of inclusion dependencies, $\mathcal{Q}, \mathcal{CDP}(\mathcal{Q}, \Sigma)$ is *NP*-complete.

Proof: Membership is clear. Now, hardness is established by reduction from the *NP*-complete vc-causality decision problem (cf. Proposition 4) for a CQ $\mathcal{Q}(\bar{x})$ over schema $\mathcal{S}$. Now, consider the schema $\mathcal{S}' := \mathcal{S} \cup \{V\}$ and the set of ICs Σ as in Proposition 8. In order to decide about $(D, \mathcal{Q}(\bar{a}), \tau)$'s membership of $\mathcal{VCDP}(\mathcal{Q})$, consider the instance D' for $\mathcal{S}'$ as in Proposition 8, with $\bar{a}$ as $\bar{a}_k$. It holds: $(D, \mathcal{Q}(\bar{a}), \tau) \in \mathcal{VCDP}(\mathcal{Q})$ iff $(D', \mathcal{Q}(\bar{a}), \tau) \in \mathcal{CDP}(\mathcal{Q}, \Sigma)$. ■

Some ICs may be implicative, which makes it tempting to give them a causal semantics. For example, in [23] and more in the context of interventions for explanations, a ground instantiation, $P_i(\bar{t}_i) \rightarrow P_j(\bar{t}_j)$, of an inclusion dependency is regarded a causal dependency of $P_j(\bar{t}_j)$ upon $P_i(\bar{t}_i)$. On this basis, a *valid*

intervention removes $P_j(\bar{t}_j)$ whenever $P_i(\bar{t}_i)$ is removed from the instance.

Giving to ICs a causal connotation is controversial. Actually, according to [14] logical dependencies are not causal dependencies *per se*. Our approach is consistent with this view. Even more, we should point out that there are different ways of seeing ICs, and they could have an impact on the notion of cause. For example, according to [22], ICs are "epistemic in nature", in the sense that rather than being statements about the domain represented by a database (or knowledge base), they are statement about the *contents* of the database, or about what it *knows* (cf. [22] for a discussion).

View-updates and abduction have been related [15], also including ICs on the base relations [9]. On the other side, we have connected QA-causality with both abduction and view-updates. We briefly illustrate using our ongoing example how the approach in [9] can be used to determine view-updates in the presence of ICs, which should have an impact on the characterization and computation of causes, now under ICs.

Example 6. (ex. 4 cont.) Formulated as a view-update problem on a Datalog setting, we have the query (2) defining an intensional predicate, $\mathcal{Q}(TStaff)$. The tuples in the underlying database are all considered to be abducible. The view-update request is the deletion of $\mathcal{Q}(John)$.

According to [9], the potential abductive explanations are maximal subsets E of the original instance D, such that R plus rule (2) does not entail $\mathcal{Q}(John)$ anymore. They are: $E_1 = D \setminus \{t_1\}$, and $E_2 = D \setminus \{t_4, t_8\}$, and are determined by finding minimal abductive explanations for $\mathcal{Q}(John)$. However, without considering the IC I.

Now, these explanations have to be examined at the light of the ICs. In this case, E_1 does satisfy I, but this is not the case for E_2. So, the latter is rejected. As a consequence, the only admissible update is the deletion of t_1 from D.

The admissible (and minimal) view-updates could be used *to define* actual causes under ICs. In this case, and according to Section 5, the admissible view-update (under ICs) should be in correspondence, by definition, with an

admissible and minimal combination of an actual cause and one of its contingency sets. This would make t_1 the only actual cause (also counterfactual) for $\langle John \rangle$ under I, which corresponds with the result obtained following our direct definition. ■

7. Conclusions

In combination with the results reported in [25], we can see that there are deeper and multiple connections between the areas of QA-causality, abductive and consistency-based diagnosis, view updates, and database repairs. Abduction has also been explicitly applied to database repairs [2]. The idea, again, is to "abduce" possible repair updates that bring the database to a consistent state. Connections between consistency-based and abductive diagnosis have been established, e.g. in [6]. Exploring and exploiting all the possible connections is matter of ongoing and future research.

Acknowledgments: Research funded by NSERC DG (250279), and the NSERC Strategic Network on Business Intelligence (BIN). This work was done while B. Salimi was a PhD student at Carleton University.

References

[1] Abiteboul, S. Hull R., and Vianu V. *Foundations of Databases*. Addison-Wesley, 1995.

[2] Arieli, O., Denecker, M., Van Nuffelen, B. and Bruynooghe, M. Coherent Integration of Databases by Abductive Logic Programming. *J. Artif. Intell. Res.*, 2004, 21:245-286.

[3] Bertossi, L. *Database Repairing and Consistent Query Answering*. Morgan & Claypool, Synthesis Lectures on Data Management, 2011.

[4] Buneman, P., Khanna, S. and Tan, W. C. On Propagation of Deletions and Annotations Through Views. Proc. PODS, 2002, pp. 150-158.

[5] Cibele, F., Gatterbauer, W., Immerman, N. and Meliou A. A Characterization of the Complexity of Resilience and Responsibility for Conjunctive Queries. *PVLDB*, 2016, 9(3). To appear.

[6] Console, L., and Torasso, P.,. A Spectrum of Logical Definitions of Model-Based Diagnosis. *Comput. Intell.*, 1991, 7:133-141.

[7] Cong, G., Fan, W. and Geerts, F. Annotation Propagation Revisited for Key Preserving Views. Proc. CIKM, 2006, pp. 632-641.

[8] Console, L., Theseider-Dupre, D. and Torasso, P. On the Relationship between Abduction and Deduction. *J. Log. Comput.*, 1991, 1(5):661-690.

[9] Console, L., Sapino M. L., Theseider-Dupre, D. The Role of Abduction in Database View Updating. *J. Intell. Inf. Syst.*, 1995, 4(3): 261-280.

[10] Eiter, T., Gottlob, G. and Leone, N. Abduction from Logic Programs: Semantics and Complexity. *Theor. Comput. Sci.*, 1997, 189(1-2):129-177.

[11] Friedrich, G., Gottlob. G. and Nejdl, W. Hypothesis Classification, Abductive Diagnosis and Therapy. Proc. Internat. Workshop on Expert Systems in Engineering, 1990, LNCS 462, pp. 69-78.

[12] Glavic, B. and Miller., R. J. Reexamining Some Holy Grails of Data Provenance. *Proc. Theory and Practice of Provenance (TaPP)*, 2011.

[13] Halpern, Y. J., Pearl, J. Causes and Explanations: A Structural-Model Approach: Part 1. *British J. Philosophy of Science*, 2005, 56:843-887.

[14] Halpern, J. Y. and Hitchcock, C. R. Actual Causation and The Art of Modelling. In *Causality, Probability, and Heuristics: A Tribute to Judea Pearl*. College Publications, 2010, pp. 383-406.

[15] Kakas A. C. and Mancarella, P. Database Updates through Abduction. Proc. VLDB, 1990, pp. 650-661.

[16] Kimelfeld, B. A Dichotomy in the Complexity of Deletion Propagation with Functional Dependencies. Proc. PODS, 2012, pp. 191-202.

[17] Kimelfeld, B., Vondrak, J. and Williams, R. Maximizing Conjunctive Views in Deletion Propagation. *ACM TODS*, 2012, 7(4):24.

[18] Meliou, A., Gatterbauer, W. Moore, K. F. and Suciu, D. The Complexity of Causality and Responsibility for Query Answers and Non-Answers. Proc. VLDB, 2010, pp. 34-41.

[19] Meliou, A., Gatterbauer, S. Nath. and Suciu, D. Tracing Data Errors with View-Conditioned Causality. Proc. SIGMOD, 2011.

[20] Chakravarthy, U.S., John Grant, J. and Minker, J. Logic-Based Approach to Semantic Query Optimization. *ACM Trans. Database Syst.*, 1990, 15(2):162-207.

[21] Reiter, R. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 1987, 32(1):57-95.

[22] Reiter, R. What Should A Database Know? *J. Log. Program*, 1992, 14(1&2):127-153.

[23] Roy, S. and Suciu, D. A Formal Approach to Finding Explanations for Database Queries. Proc. SIGMOD, 2014, pp. 1579-1590,

[24] Salimi, B. and Bertossi, L. Causes for Query Answers from Databases, Datalog Abduction and View-Updates: The Presence of Integrity Constraints. Proc. FLAIRS, 2016. To appear.

[25] Salimi, B. and Bertossi, L. From Causes for Database Queries to Repairs and Model-Based Diagnosis and Back. Proc. ICDT, 2015.

[26] Salimi, B. and Bertossi, L. Query-Answer Causality in Databases: Abductive Diagnosis and View-Updates. Proc. UAI'15 Workshop on Causal Inference. CEUR-WS Proceedings, Vol-1504, 2015.