Query-Answer Causality in Databases: Abductive Diagnosis and View-Updates

Babak Salimi and **Leopoldo Bertossi**
School of Computer Science
Carleton University
Ottawa, Canada.
{bsalimi,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 causality and consistency-based diagnosis and database repairs (wrt. integrity constrain violations) have been established in the literature. In this work we establish connections between query causality and abductive diagnosis and the view-update problem. The unveiled relationships allow us to obtain new complexity results for query causality -the main focus of our work- and also for the two other areas.

Causality is an important notion that appears at the foundations of many scientific disciplines, in the practice of technology, and also in our everyday life. Causality is unavoidable to understand and manage *uncertainty* in data, information, knowledge, and theories. In data management in particular, there is a need to represent, characterize and compute the 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).

In this work we concentrate on causality as defined forand applied to relational databases. Most of the work on causality has been developed in the context of knowledge representation, and little has been said about causality in data management. Furthermore, in a world of big uncertain data, the necessity to understand the data beyond simple query answering, introducing explanations in different forms, has become particularly relevant.

The notion of causality-based explanation for a query result was introduced in (Meliou et al., 2010a), on the basis of the deeper concept of *actual causation*.[1] Intuitively, a tuple (of constants) 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" subset of tuples Γ, accompanying t, such that, after removing Γ from D, removing t from $D \smallsetminus \Gamma$ causes $\bar{a}$ to switch from being an answer to being a non-answer (i.e. not being an answer). Usually, 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*.

A cause t may have different associated contingency sets Γ. Intuitively, the smaller they are the strongest is t as a cause (it need less company to undermine the query answer). So, some causes may be stronger than others. This idea is formally captured through the notion of *causal responsibility*, and introduced in (Meliou et al., 2010a). It reflects the relative degree of actual causality. In applications involving large data sets, it is crucial to rank potential causes according to their responsibilities (Meliou et al., 2010b,a).

Furthermore, *view-conditioned causality* was proposed in (Meliou et al., 2010b, 2011) as a restricted form of query causality, to determine causes for a set of unexpected query results, but conditioned to the correctness of prior knowledge about some other set of results.

Actual causation, as used in (Meliou et al., 2010a,b, 2011), can be traced back to (Halpern & Pearl, 2001, 2005), which provides a model-based account of causation on the basis of *counterfactual dependence*.[2] *Causal responsibility* was introduced in Chockler & Halpern (2004), to provide a graded, quantitative notion of causality when multiple causes may over-determine an outcome.

---

[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".

[2]As discussed in (Salimi & Bertossi, 2015), some objections to the Halpern-Pearl model of causality and the corresponding changes (Halpern, 2014, 2015) do not affect results in the context of databases.

Model-based diagnosis (Struss, 2008, sec. 10.3), an area of knowledge representation, addresses the problem of, given the *specification* of a system in some logical formalism and a usually unexpected *observation* about the system, obtaining *explanations* for the observation, in the form of a diagnosis for the unintended behavior. Since this and causality are related to explanations, a first connection between causality and *consistency-based diagnosis* (Reiter, 1987), a form of model-based diagnosis, was established in (Salimi & Bertossi, 2014, 2015): Causality and the responsibility problem can be formulated as *consistency-based diagnosis* problems, which allowed to extend the results in (Meliou et al., 2010a). However, no precise connection has been established so far between causality and *abductive diagnosis* (Console et al., 1991; Eiter & Gottlob, 1995), another form of model-based diagnosis.

The definition of causality for query answers applies to monotone queries (Meliou et al., 2010a,b). However, all complexity and algorithmic results in (Meliou et al., 2010a; Salimi & Bertossi, 2015) have been restricted to first-order (FO) monotone queries. Other important classes of monotone queries, such as Datalog queries (Ceri et al., 1989; Abiteboul et al., 1995), possibly with recursion, require further investigation.

In (Salimi & Bertossi, 2015) connections were established between query causality, database repairs (Bertossi, 2011), and consistency-based diagnosis. In particular, complexity results for several causality problems were obtained from the repair connection. In the line of this kind of research, in this work we unveil natural connections between actual causation and *abductive diagnosis*, and also the view-update problem in databases (more on this latter connection later in the section).

As opposed to consistency-based diagnoses, which is usually practiced with FO specifications, abductive diagnosis is commonly performed under a logic programming (LP) approach (in the general sense of LP) to knowledge representation (Denecker & Kakas, 2002; Eiter et al., 1997; Gottlob et al., 2010b). Since Datalog can be seen as a form of LP, we manage to extend and formulate the notion of query-answer causality to Datalog queries via the abductive diagnosis connection, in this way extending causality to a new class of queries, e.g. recursive queries, and obtaining complexity results on causality for them.

Abductive reasoning/diagnosis has been applied to the view update problem in databases (Kakas & Mancarella, 1990; Console et al., 1995), which is about characterizing and computing updates of physical database relations that give an account of (or have as result) the intended updates on views. The idea is that abductive diagnosis provides (abduces) the reasons for the desired view updates, and they are given as changes on base tables.

In this work we also explore fruitful connections of causality with this *view-update problem* (Abiteboul et al., 1995), i.e. about updating a database through views. An important aspect of the problem is that one want the base, source database, i.e. the base relations, to change in a minimally way while still producing the view updates. Put in different terms, it is an update propagation problem, from views to base relations. This classical and important problem in databases.

The *delete-propagation* problem (Buneman et al., 2002; Kimelfeld, 2012; Kimelfeld et al., 2012) is a particular case of the view-update problem where only tuple deletions are allowed on/from the views. If the views are defined by monotone queries, only database deletions can give an account of view deletions. So, in this case, a minimal set (in some sense) of deletions from the base relations is expected to be performed. This is "minimal source-side-effect" case. It is also possible to consider minimizing the side-effect on the view, which also requires that other tuples in the (virtual) view contents are not affected (deleted) (Buneman et al., 2002).

In this work we provide a precise connection between different variants of the delete-propagation problem and query causality. In particular, we show that the minimal source-side-effect problem is related to the *most-responsible cause problem*, which was formulated and investigated in (Salimi & Bertossi, 2015); and also that the "minimal view side-effect problem" is related to view-conditioned causality we already mentioned above.

More precisely, our main results are as follows:[3] (the complexity results are all in data complexity)

1. We establish precise connections between causality for Datalog queries and abductive diagnosis. More precisely, we establish mutual characterizations of each in terms of the other, and computational reductions, between actual causes for Datalog queries and abductive diagnosis from Datalog specifications.

   We profit from these connections to obtain new algorithmic and complexity results for each of the two problems separately.

   (a) We characterize and obtain causes in terms of- and from abductive diagnoses.
   (b) We show that deciding causality for Datalog queries, possibly recursive, is *NP*-complete.
   (c) We introduce a class of Datalog queries for which deciding causality is tractable.

2. We establish and profit from precise connections between delete-propagation and causality. More precisely, we show that:

[3]The possible connections between the areas and problems in this paper were suggested in (Bertossi & Salimi, 2014), but no precise results were formulated there.

(a) Most-responsible causes and view-conditioned causes can obtained from solutions to different variants of the delete-propagation problem and vice-versa.

(b) Computing the size of the solution to a minimum source-side-effect problem is hard for $FP^{NP(log(n))}$.

(c) Deciding weather an answer has a view-conditioned cause is *NP*-complete.

(d) We can identify some new classes of queries for which computing minimum source-side-effect delete-propagation is tractable.

1 PRELIMINARIES AND DECISION PROBLEMS

We consider relational database schemas of the form $\mathcal{S} = (U, \mathcal{P})$, where U is the possibly infinite database domain and $\mathcal{P}$ is a finite set of *database predicates*[4] 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 the constants $c_1, \ldots, c_n \in U$.

A *conjunctive query* (CQ) is a formula $\mathcal{Q}(\bar{x})$ of the first-order (FO) 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 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 an open conjunctive 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 $D \models \mathcal{Q}$ and $D \not\models \mathcal{Q}$, respectively. When $\mathcal{Q}$ is a BCQ, or contains no free variables, $\mathcal{Q}(D) = \{yes\}$ if $\mathcal{Q}$ is true, and $\mathcal{Q}(D) = \emptyset$, otherwise.

A query $\mathcal{Q}$ is *monotone* if for every two instances $D_1 \subseteq D_2$, $\mathcal{Q}(D_1) \subseteq \mathcal{Q}(D_2)$, i.e. the set of answers grows monotonically with the instance. For example, CQs and unions of CQ (UCQs) are monotone queries. Datalog queries (Ceri et al., 1989; Abiteboul et al., 1995), although not FO, are also monotone.

CQs can be expressed as Datalog queries, e.g. the CQ $\exists \bar{y}(P_1(\bar{s}_1) \wedge \cdots \wedge P_m(\bar{s}_m))$ as the rule: $Ans(\bar{x}) \leftarrow P_1(\bar{s}_1), \ldots, P_m(\bar{s}_m)$, where $\bar{x}$ are the free variables in $\mathcal{Q}$. If the Datalog query is boolean (e.g. a BCQ), then the Datalog query program Π contains a top answer-collecting rule of the form $ans \leftarrow P_1(\bar{s}_1), \ldots, P_m(\bar{s}_m)$, with ans a propositional atom. When the query is true, $\mathcal{Q}(D) = \{ans\}$, and empty otherwise. $\mathcal{Q}(D) = \{ans\}$, which can also be denoted as $\Pi \cup D \models ans$, means that ans belongs to the minimal model of $\Pi \cup D$ (Ceri et al., 1989; Abiteboul et al., 1995).

[4] As opposed to built-in predicates (e.g. $\neq$) that we assume do not appear, unless explicitly stated otherwise.

1.1 CAUSALITY AND RESPONSIBILITY

In the rest of this work, unless otherwise stated, we will assume that a database instance D is split in two disjoint sets, $D = D^n \cup D^x$, where D^n and D^x denote the sets of *endogenous* and *exogenous* tuples, respectively; and $\mathcal{Q}$ is a monotone query.

Definition 1.1. A tuple $\tau \in D^n$ is a *counterfactual cause* for an answer $\bar{a}$ to $\mathcal{Q}$ in D if $D \models \mathcal{Q}(\bar{a})$ and $D \smallsetminus \{\tau\} \not\models \mathcal{Q}(\bar{a})$. 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 $D \smallsetminus \Gamma$. □

When the query is boolean and is true in D, we look for causes for the answer yes. Notice that this definition can be applied without any conceptual changes to Datalog queries. In this case, $D \models \mathcal{Q}(\bar{a})$ means $\Pi \cup D \models Ans(\bar{a})$, where predicate Ans is defined by an answer-collecting top rule in program Π; and entailment ($\models$) means belonging to the minimal model.

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

Given a $\tau \in Causes(D, \mathcal{Q}(\bar{a}))$, we collect all subset-minimal contingency sets associated with τ:

$$\begin{aligned} Cont(D, \mathcal{Q}(\bar{a}), \tau) \; := \; \{\Lambda \subseteq D^n \mid\; & D \smallsetminus \Lambda \models Q(\bar{a}), \\ & D \smallsetminus (\Lambda \cup \{\tau\}) \not\models \mathcal{Q}(\bar{a}), \text{ and} \\ & \forall \Lambda' \subsetneqq \Lambda,\; D \smallsetminus (\Lambda' \cup \{\tau\}) \models \mathcal{Q}(\bar{a})\}. \end{aligned}$$

The *responsibility* of actual cause τ for answer $\bar{a}$, denoted $\rho_{\mathcal{Q}(\bar{a})}(\tau)$, is $\frac{1}{(|\Gamma|+1)}$, where $|\Gamma|$ is the size of the smallest contingency set for τ. Responsibility can be extend to all tuples in D^n by setting their value to 0, and they are not actual causes for $\mathcal{Q}$.

In (Meliou et al., 2010a), causality for non-query answers is defined on basis of sets of *potentially missing tuples* that account for the missing answer. Computing actual causes and their responsibilities for non-answers becomes a rather simple variation of causes for answers. In this work we focus on causality for query answers.

Example 1.1. Consider a database D with relations *Author*(*Name*,*Journal*) and *Journal*(*JName*,*Topic*,*#Paper*), and contents as below:

Author	*Name*	*Journal*
	Joe	TKDE
	John	TKDE
	Tom	TKDE
	John	TODS

Journal	*JName*	*Topic*	*#Paper*
	TKDE	XML	30
	TKDE	CUBE	31
	TODS	XML	32

Consider the query:

$$Ans_{\mathcal{Q}}(\textit{Name}, \textit{Topic}) \quad \longleftarrow \quad \textit{Author}(\textit{Name},\textit{Journal}),$$
$$\textit{Journal}(\textit{JName},\textit{Topic},\textit{\#Paper}),$$

with the following answers:

$\mathcal{Q}(D)$	*Name*	*Topic*
	Joe	XML
	Joe	CUBE
	Tom	XML
	Tom	CUBE
	John	XML
	John	CUBE

Assume $\langle$*John, XML*$\rangle$ is an unexpected answer to $\mathcal{Q}$, and we want to compute its causes assuming that all tuples are endogenous.

It turns out that *Author(John, TODS)* is an actual cause, with contingency sets $\Gamma_1 = \{$*Author(John, TKDE)*$\}$ and Γ_2=$\{$*Journal(TKDE, XML, 32)*$\}$, because *Author(John, TODS)* is a counterfactual cause for answer $\langle$ *John, XML*$\rangle$ in both of $D \smallsetminus \Gamma_1$ and $D \smallsetminus \Gamma_2$. Therefore, the responsibility of *Author(John, TODS)* is $\frac{1}{2}$.

Likewise, *Journal(TKDE, XML, 32)*, *Author(John, TKDE)*, *Journal(TODS,XML, 32)* are actual causes for $\langle$*John, XML*$\rangle$ with responsibility $\frac{1}{2}$.

Now, under the assumption that the tuples in *Journal* are the endogenous tuples, the only actual causes for answer $\langle$*John, XML*$\rangle$ are *Author(John, TKDE)* and *Author(John, TODS)*. $\square$

The complexity of the computational and decision problems that arise in query causality have been investigated in (Meliou et al., 2010a; Salimi & Bertossi, 2015). Here we present some problems and results that we use throughout this paper. The first is the causality problem, about deciding whether a tuple is an actual cause for a query answer.

Definition 1.2. For a boolean monotone query $\mathcal{Q}$, the *causality decision problem* (CDP) is (deciding about membership of):
$CDP := \{(D, \tau) \mid \tau \in D^n, \text{ and } \tau \in \textit{Causes}(D, \mathcal{Q})\}$. $\square$

This problem is tractable for UCQs (Salimi & Bertossi, 2015). The next is the responsibility problem, about deciding responsibility (above a given bound) of a tuple for a query result.

Definition 1.3. For a boolean monotone query $\mathcal{Q}$, the *responsibility decision problem* (RDP) is (deciding about membership of):
$\mathcal{RDP}(\mathcal{Q}) = \{(D, \tau, v) \mid \tau \in D^n, v \in \{0\} \cup \{\frac{1}{k} \mid k \in \mathbb{N}^+\}, D \models \mathcal{Q} \text{ and } \rho_{\mathcal{Q}}(\tau) > v\}$. $\square$

This problem is *NP*-complete for UCQs (Salimi & Bertossi, 2015), but tractable for *linear* CQs (Meliou et al., 2010a). Roughly speaking, a CQ is linear if its atoms can be ordered in a way that every variable appears in a continuous sequence of atoms that does not contain a self-join (i.e. a join involving the same predicate), e.g. $\exists xvyu(A(x) \wedge S_1(x, v) \wedge S_2(v, y) \wedge R(y, u) \wedge S_3(y, z))$ is linear, but not $\exists xyz(A(x) \wedge B(y) \wedge C(z) \wedge W(x, y, z))$, for which RDP is *NP*-complete. The class of CQs for which RDP is tractable can be extended to *weakly linear*.[5]

The functional, non-decision version of RDP, about computing the responsibility, i.e. an optimization problem, is complete for $FP^{NP(log(n))}$ for UCQs (Salimi & Bertossi, 2015).

Finally, we have the problem of deciding weather a tuple is a most responsible cause:

Definition 1.4. For a boolean monotone query $\mathcal{Q}$, the *most responsible cause decision problem* (MRDP) is:
$\mathcal{MRCD}(\mathcal{Q}) = \{(D, \tau) \mid \tau \in D^n \text{ and } 0 < \rho_{\mathcal{Q}}(\tau) \text{ is a maximum for } D\}$. $\square$

For UCQs this problem is complete for $P^{NP(log(n))}$ (Salimi & Bertossi, 2015).

1.2 VIEW-CONDITIONED CAUSALITY

A form of *conditional causality* was informally introduced in (Meliou et al., 2010b), to characterize causes for a query answer that are conditioned by the other answers to the query. The notion was made precise in (Meliou et al., 2011), in a more general, non-relational setting that in particular includes the case of several queries. In them the notion of *view-conditioned causality was used*, and we adapt it in the following to the case of a single query, possibly with several answers.

Consider an instance $D = D^n \cup D^x$, and a monotone query $\mathcal{Q}$ with $\mathcal{Q}(D) = \{\bar{a}_1, \ldots \bar{a}_n\}$. Fix an answer, say $\bar{a}_k \in \mathcal{Q}(D)$, while the other answers will be used as a condition on $\bar{a}_k$'s causality. Intuitively, $\bar{a}_k$ is somehow unexpected, and we look for causes, by considering the other answers as "correct". The latter assumption has, in technical terms, the effect of reducing the spectrum of contingency sets, by keeping $\mathcal{Q}(D)$'s extension fixed, as a view, modulo the answer $\bar{a}_k$ at hand.

Definition 1.5. (a) A tuple $\tau \in D^n$ is called a *view-conditioned counterfactual cause* (VCC-cause) for answer $\bar{a}_k$ to $\mathcal{Q}$ if $D \smallsetminus \{\tau\} \not\models \mathcal{Q}(\bar{a}_k)$ and $D \smallsetminus \{\tau\} \models \mathcal{Q}(\bar{a}_i)$, for $i \in \{1, \ldots, n\} \smallsetminus \{k\}$.

(b) A tuple $\tau \in D^n$ is an *view-conditioned actual cause* (VC-cause) for $\bar{a}_k$ if there exists a contingency set, $\Gamma \subseteq D^n$, such τ is a VCC-cause for $\bar{a}_k$ in $D \smallsetminus \Gamma$.

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

[5] Computing sizes of minimum contingency sets is reduced to the max-flow/min-cut problem in a network.

Intuitively, a tuple τ is a VC-cause for $\bar{a}_k$ if there is a contingent state of the database that entails all the answers to $\mathcal{Q}$ and τ is a counterfactual cause for $\bar{a}_k$, but not for the rest of the answers. Obviously, VC-causes for $\bar{a}_k$ are also actual causes, but not necessarily the other way around: $\mathit{vc\text{-}Causes}(D, \mathcal{Q}(a_k)) \subseteq \mathit{Causes}(D, \mathcal{Q}(a_k))$.

Example 1.2. (ex. 1.1 cont.) Consider the same instance D, query $\mathcal{Q}$, and the answer ⟨*John, XML*⟩, which does not have any VC-cause. To see this, take for example, the tuple *Author(John, TODS)* that is an actual cause for ⟨*John, XML*⟩, with two contingency sets, Γ_1 and Γ_2. It is easy to verify that none of these contingency sets satisfies the condition in Definition 1.5, e.g. the original answer ⟨*John, CUBE*⟩ is not such anymore from $D \smallsetminus \Gamma_1$. The same argument can be applied to all actual causes for ⟨*John, XML*⟩. □

This example shows that it makes sense to study the complexity of deciding whether a query answer has a VC-actual cause or not.

Definition 1.6. For a monotone query $\mathcal{Q}$, the *view-conditioned cause problem* is (deciding about membership of):

$$\mathcal{VCP}(\mathcal{Q}) = \{(D, \bar{a}) \mid \bar{a} \in \mathcal{Q}(D) \text{ and } \mathit{vc\text{-}Causes}(D, \mathcal{Q}(\bar{a})) \neq \emptyset\}. \quad \square$$

1.3 DATALOG ABDUCTIVE DIAGNOSIS

In this section we will establish connections between abductive diagnosis and database causality.[6] For that, we have to be more precise about the kind of abduction problems we will consider.

A *Datalog abduction problem* (Eiter et al., 1997) is of the form $\mathcal{AP} = \langle \Pi, E, \mathit{Hyp}, \mathit{Obs}\rangle$, where: (a) Π is a set of Datalog rules, (b) E is a set of ground atoms (the extensional database), whose predicates do not appear in heads of rules in Π, (c) Hyp, the hypothesis, is a finite set of ground atoms, the abducible atoms in this case,[7] and (d) Obs, the observation, is a finite conjunction of ground atoms. As it is common, we will start with the assumption that $\Pi \cup E \cup \mathit{Hyp} \models \mathit{Obs}$.

The *abduction problem* is about computing a minimal $\Delta \subseteq \mathit{Hyp}$ (under certain minimality criterion), such that $\Pi \cup E \cup \Delta \models \mathit{Obs}$. More specifically:

Definition 1.7. Consider a *Datalog abduction problem* $\mathcal{AP} = \langle \Pi, E, \mathit{Hyp}, \mathit{Obs}\rangle$

(a) An *abductive diagnosis* (or simply, *a solution*) for $\mathcal{AP}$ is a subset-minimal $\Delta \subseteq \mathit{Hyp}$, such that $\Pi \cup E \cup \Delta \models \mathit{Obs}$. This requires that no proper subset of Δ has this property. $\mathit{Sol}(\mathcal{AP})$ denotes the set of abductive diagnoses for problem $\mathcal{AP}$.

(b) A hypothesis $h \in \mathit{Hyp}$ is *relevant* for $\mathcal{AP}$ if h contained in at least one diagnosis of $\mathcal{AP}$. $\mathit{Rel}(\mathcal{AP})$ collects all relevant hypothesis for $\mathcal{AP}$. □

We are interested in deciding, for a fixed Datalog program, if an hypothesis is relevant or not, with all the data as input.

More precisely, we consider the following decision problem.

Definition 1.8. Given a Datalog program Π, the *relevance decision problem* (RLDP) is (deciding about the membership of):

$$\mathcal{RLDP}(\Pi) = \{(E, \mathit{Hyp}, \mathit{Obs}, h) \mid h \in \mathit{Rel}(\mathcal{AP}), \text{ with } \mathcal{AP} = \langle \Pi, E, \mathit{Hyp}, \mathit{Obs}\rangle \text{ and } h \in \mathit{Hyp}\}. \ \square$$

As it is common, we will assume that $|\mathit{Obs}|$, i.e. the number of atoms in the conjunction, is bounded above by a numerical parameter p. It is common that $p = 1$ (a single atomic observation).

The following result immediately follows from the *NP*-completeness of the relevance problem for abduction with propositional Horn theories (Friedrich et al., 1990). This is a particular case of a more general result in (Eiter et al., 1997).

Proposition 1.1. $\mathcal{RLDP}$ is *NP*-complete in $|E|+|\mathit{Hyp}|$. □

A tractable case of Datalog abduction is identified in (Gottlob et al., 2010b), on the basis of the notions of *tree-decomposition and bounded tree-width*, which we now briefly present.

Let $\mathcal{H} = \langle V, H\rangle$ be a hypergraph. V is the set of vertices, and H the set of hyperedges, i.e. of subsets of V. A tree-decomposition $\mathcal{T}$ of $\mathcal{H}$ is a pair $(\mathcal{T}, \lambda)$, where $\mathcal{T} = \langle N, E\rangle$ is a tree and λ is a labeling function that assigns to each node $n \in N$, a subset $\lambda(n)$ of V ($\lambda(n)$ is aka. bag), i.e. $\lambda(n) \subseteq V$, such that, for every node $n \in N$, the following hold: (a) For every $v \in V$, there exists $n \in N$ with $v \in \lambda(n)$. (b) For every $h \in H$, there exists a node $n \in N$ with $h \subseteq \lambda(n)$. (c) For every $v \in V$, the set of nodes $\{n \mid v \in \lambda(n)\}$ induces a connected subtree of $\mathcal{T}$.

The *width* of a tree decomposition $(\mathcal{T}, \lambda)$ of $\mathcal{H} = \langle V, H\rangle$, with $\mathcal{T} = \langle N, E\rangle$, is defined as $max\{|\lambda(n)|-1 : n \in N\}$. The tree-width $t_w(\mathcal{H})$ of $\mathcal{H}$ is the minimum width over all its tree decompositions.

Intuitively, the tree-width of a hypergraph $\mathcal{H}$ is a measure of the "tree-likeness" of $\mathcal{H}$. A set of vertices that form a cycle in $\mathcal{H}$ are put into a same bag, which becomes (the

[6] In (Salimi & Bertossi, 2015) we established such a connection between another form of model-based diagnosis (Struss, 2008), namely consistency-based diagnosis (Reiter, 1987). For relationships and comparisons between consistency-based and abductive diagnosis see (Console et al., 1991).

[7] It is common to accept as hypothesis all the possible ground instantiations of *abducible predicates*. We assume abducible predicates do not appear in rule heads.

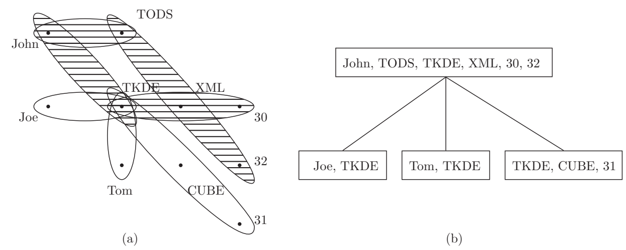

Figure 1: (a) $\mathcal{H}(D)$. (b) A tree decomposition of $\mathcal{H}(D)$.

bag of a) node in the corresponding tree-decomposition. If the tree-width of the hypergraph under consideration is bounded by a fixed constant, then many otherwise intractable problems become tractable (Gottlob et al., 2010a).

It is possible to associate an hypergraph to any finite structure D (think of a relational database): If its universe (the active domain in the case of a relational database) is V, define the hypergraph $\mathcal{H}(D) = (V, H)$, with $H = \{\ \{a_1, \ldots, a_n\} \mid D$ contains a ground atom $P(a_1 \ldots a_n)$ for some predicate symbol $P\}$.

Example 1.3. Consider instance D in Example 1.1. The hypergraph $\mathcal{H}(D)$ associated to D is shown in Figure 1(a). Its vertices are the elements of $adom(D) = \{John, Jone, Tom, TODS, TKDE, XML, Cube, 30, 31, 32\}$, the active domain of D. For example, since $Journal(TKDE, XML, 30) \in D$, $\{TKDE, XML, 30\}$ is one of the hyperedges.

The dashed ovals show four sets of vertices, i.e. hyperedges, that together form a cycle. Their elements are put into the same bag of the tree-decomposition. Figure 1(b) shows a possible tree-decomposition of $\mathcal{H}(D)$. In it, the maximum $|\lambda(n)| - 1$ is $6-1$, corresponding to the top box bag of the tree. So, $t_w(\mathcal{H}(D)) \leq 5$. $\square$

The following is a *fixed-parameter tractability* result for the relevance decision problem for Datalog abduction problems with a program Π that is *guarded*, which means that in every rule body there is an atom that contains (guards) all the variables appearing in that body.

Theorem 1.2. (Gottlob et al., 2010b) Let k be an integer, and $\mathcal{AP} = \langle \Pi, E, Hyp, Obs\rangle$ a Datalog abduction problem with guarded Π, and $t_w(\mathcal{H}(E)) \leq k$. It can be decided in polynomial time in $|E| + |Hyp|$, for $h \in Hyp$, whether $h \in Rel(\mathcal{AP})$.[8] $\square$

This is particular case of the problem in Definition 1.8 where the program is guarded and the fixed parameter is the tree-width of the extensional database.

1.4 THE DELETE-PROPAGATION PROBLEM

Given a monotone query $\mathcal{Q}$, we can think of it as defining a view with virtual contents $\mathcal{Q}(D)$. If $\bar{a} \in \mathcal{Q}(D)$, which may not be intended, we may try to delete some tuples from D, so that $\bar{a}$ disappears from $\mathcal{Q}(D)$. This is a common case of the problem of database updates through views (Abiteboul et al., 1995). In this work we consider some variations of this problem. We consider both the functional and the decision versions of them.

Definition 1.9. For an instance D and a monotone $\mathcal{Q}$:

(a) The *minimal source-side-effect problem* is about computing a subset-minimal $\Lambda \subseteq D$ with $\bar{a} \in (\mathcal{Q}(D) \smallsetminus \mathcal{Q}(D \smallsetminus \Lambda))$.

(b) The *minimal source-side-effect decision problem* is (deciding about the membership of): (superscript s stands for subset-minimal)

$$\mathcal{MSSEP}^s(\mathcal{Q}) = \{(D, \Lambda, \bar{a}) \mid \bar{a} \in \mathcal{Q}(D),\ \Lambda \subseteq D,\ \bar{a} \notin \mathcal{Q}(\Lambda),\ \text{and } \Lambda \text{ is subset-maximal}\}.\ \square$$

Definition 1.10. For an instance D and a monotone $\mathcal{Q}$:

(a) The *minimum source side-effect problem* is about computing a minimum-cardinality $\Lambda \subseteq D$, such that $\bar{a} \in (\mathcal{Q}(D) \smallsetminus \mathcal{Q}(D \smallsetminus \Lambda))$.

(b) The *minimum source side-effect decision problem* is (deciding about the membership of): (c stands for minimum cardinality)

$$\mathcal{MSSEP}^c(\mathcal{Q}) = \{(D, \Lambda, \bar{a}) \mid \bar{a} \in \mathcal{Q}(D), \Lambda \subseteq D, \bar{a} \notin \mathcal{Q}(\Lambda),\ \text{and } \Lambda \text{ has maximum cardinality}\}.\ \square$$

Definition 1.11. For an instance D and a monotone $\mathcal{Q}$:

(a) The *view side-effect-free problem* is about computing a $\Lambda \subseteq D$, such that $\mathcal{Q}(D) \smallsetminus \{\bar{a}\} = \mathcal{Q}(D \smallsetminus \Lambda)$ (Buneman et al., 2002).

(b) The *view side-effect-free decision problem* is (deciding about the membership of):

$$\mathcal{VSEFP}(\mathcal{Q}) = \{(D, \bar{a}) \mid \bar{a} \in \mathcal{Q}(D),\ \exists \Lambda \subseteq D\ \text{ with } \mathcal{Q}(\Lambda) = \mathcal{Q}(D) \smallsetminus \{\bar{a}\}\}.\ \square$$

2 CAUSALITY AND ABDUCTION

In general logical terms, an abductive explanation of an observation is a formula that, together with the background logical theory, entails the observation. So, one could see an abductive explanation as a cause for the observation. However, it has been argued that causes and abductive explanations are not necessarily the same (Psillos, 1996; Denecker & Kakas, 2002).

Under the abductive approach to diagnosis (Console et al., 1991; Eiter & Gottlob, 1995; Poole, 1992, 1994), it is common that the system specification rather explicitly describes causality information, specially in action theories where the effects of actions are directly represented by Horn formulas. By restricting the explanation formulas to the predicates describing primitive causes (action executions), an explanation formula which entails an observation gives also

[8] This is Theorem 7.9 in (Gottlob et al., 2010b), which can be traced back eventually to Theorem 4.3 therein.

a cause for the observation (Denecker & Kakas, 2002). In this case, and is some sense, causality information is imposed by the system specifier (Poole, 1992).

In database causality we do not have, at least not initially, a system description,[9] but just a set of tuples. It is when we pose a query that we create something like a description, and the causal relationships between tuples are captured by the combination of atoms in the query. If the query is a Datalog query (in particular, a CQ), then we have a Horn specification too.

In this section we show that, for the class of Datalog theories (system specifications), abductive inference corresponds to actual causation for monotone queries. That is, abductive diagnoses for an observation essentially contain actual causes for the observation.

2.1 QUERY CAUSALITY FROM ABDUCTIVE DIAGNOSIS

Assume the relational instance is $D = D^x \cup D^n$. Part of it will serve as the extensional database for a Datalog program, Π, that represents a boolean, possibly recursive query. Then, as described above, we use a propositional top-level atom ans. Let us assume that $\Pi \cup D \models ans$.

We now show that 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 is the extensional database for Π (and then $\Pi \cup D^x$ becomes the *background theory*), D^n becomes the set of *hypothesis*, and atom ans is the observation.

Proposition 2.1. $t \in D^n$ is an actual cause for ans iff $t \in \mathit{Rel}(\mathcal{AP}^c)$. $\square$

Example 2.1. Consider the instance D with relations R and S as below, and the query $\Pi:\ ans \leftarrow R(x,y), S(y)$, which is true in D. Assume all tuples are endogenous.

R	X	Y
	a_1	a_4
	a_2	a_1
	a_3	a_3

S	X
	a_1
	a_2
	a_3

$\mathcal{AP}^c = \langle \Pi, \emptyset, D, ans\rangle$ has two (subset-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, $\mathit{Rel}(\mathcal{AP}^c) = \{S(a_3), R(a_3,a_3), S(a_1), R(a_2,a_1)\}$. It is easy to see that the relevant hypothesis are actual causes for ans. $\square$

We are interested in obtaining *responsibilities* of actual causes for ans.

[9]Having integrity constraints would go in that direction, but we are not considering their presence in this work. However, see (Salimi & Bertossi, 2015, sec. 5) for a consistency-based diagnosis connection.

Definition 2.1. Given a CDAP, $\mathcal{AP}^c = \langle \Pi, D^x, D^n, ans\rangle$, with $\mathit{Sol}(\mathcal{AP}^c) \neq \emptyset$, $N \subseteq D^n$ is a *necessary-hypothesis set* if N is subset-minimal such that $\mathit{Sol}(\mathcal{AP}^c_N) = \emptyset$, with $\mathcal{AP}^c_N := \langle \Pi, D^x, D^n \smallsetminus N, ans\rangle$. $\square$

Proposition 2.2. The responsibility of a tuple t for ans is $\frac{1}{|N|}$, where N is a necessary-hypothesis set with minimum cardinality for $\mathcal{AP}^c$ and $t \in N$. $\square$

In order to represent Datalog abduction in terms of actual causation for query answer, we show that abductive diagnoses from Datalog queries essentially are formed by actual causes for the observation.

More precisely, consider a Datalog abduction problem $\mathcal{AP} = \langle \Pi, E, \mathit{Hyp}, \mathit{Obs}\rangle$, where E is the underlying extensional database, and Obs is a conjunction of ground atoms.

Now we construct a query-causality setting: $D := D^x \cup D^n$, $D^x := E$, and $D^n := \mathit{Hyp}$. Consider the program $\Pi' := \Pi \cup \{ans \leftarrow \mathit{Obs}\}$ (with ans a fresh propositional atom). So, Π' is seen as monotone query on D.

Proposition 2.3. A hypothesis h is relevant for $\mathcal{AP}$, i.e. $h \in \mathit{Rel}(\mathcal{AP})$, iff h is an actual cause for ans wrt. Π', D. $\square$

In Section 4 will use the results obtained in this section to obtain new complexity results for Datalog abduction and query causality.

3 VIEW-UPDATES AND QUERY CAUSALITY

There is a close relationship between query causality and the view-update problem in the form of delete-propagation, which was first suggested in (Kimelfeld, 2012; Kimelfeld et al., 2012) (see also (Buneman et al., 2002)). *In this section all tuples in the instances involved are assumed to be endogenous.*

Consider a relational database D, a view $\mathcal{V}$ defined by a monotone query $\mathcal{Q}$. So, the virtual view extension, $\mathcal{V}(D)$, is $\mathcal{Q}(D)$.

For a tuple $\bar{a} \in \mathcal{V}(D)$, the delete-propagation problem, in its most general form, is the task of deleting a set of tuples from D, and so obtaining a subinstance D' of D, such that $\bar{a} \notin \mathcal{V}(D')$. It is natural to expect that the deletion of $\bar{a}$ from the view can be achieved through deletions from D of its actual causes. However, to obtain solutions to the different variants of this problem introduced in Section 1, different sets of actual causes must be considered.

First, we show that an actual cause for $\bar{a}$ to be in $\mathcal{V}(D)$ with a contingency set forms a solution to the minimal source-side-effect problem (cf. Definition 1.9).

Proposition 3.1. Consider an instance D, a view $\mathcal{V}$ defined by a monotone query $\mathcal{Q}$, and $\bar{a} \in \mathcal{V}(D)$: $D' \subseteq D$ is a solution to the minimal source-side-effect problem, i.e. $(D, D', \bar{a}) \in \mathcal{MSSEP}^s(\mathcal{Q})$, iff there is $t \in D \smallsetminus D'$, such that $t \in \mathit{Causes}(D, \mathcal{Q}(\bar{a}))$ and $D \smallsetminus (D' \cup \{t\}) \in \mathit{Cont}(D, \mathcal{Q}(\bar{a}), t)$. $\square$

Now we show that, in order to minimize the side-effect on the source (cf. Definition 1.10), it is good enough to pick a most responsible cause for $\bar{a}$ with its minimum-cardinality contingency set.

Proposition 3.2. Consider an instance D, a view $\mathcal{V}$ defined by a monotone query $\mathcal{Q}$, and $\bar{a} \in \mathcal{V}(D)$: $D' \subseteq D$ is a solution to the minimum source-side-effect problem, i.e. $(D, D', \bar{a}) \in \mathcal{MSSEP}^c(\mathcal{Q})$, iff there is $t \in D \smallsetminus D'$, such that $t \in \mathcal{MRC}(D, \mathcal{Q}(\bar{a}))$, $\Lambda := D \smallsetminus (D' \cup \{t\}) \in \mathit{Cont}(D, \mathcal{Q}(\bar{a}), t)$, and there is no $\Lambda' \in \mathit{Cont}(D, \mathcal{Q}(\bar{a}), t)$ with $|\Lambda'| < |\Lambda|$. $\square$

Next, we show that in order to check if there exists a solution to the view side-effect-free problem for $\bar{a} \in \mathcal{V}(D)$ (cf. Definition 1.11), it is good enough to check if $\bar{a}$ has a view-conditioned cause.

Proposition 3.3. Consider an instance D, a view $\mathcal{V}$ defined by a monotone query $\mathcal{Q}$, and $\bar{a} \in \mathcal{V}(D)$: There is a solution to the view side-effect-free problem for $\bar{a}$, i.e. $(D, \bar{a}) \in \mathcal{VSEFP}(\mathcal{Q})$, iff $\mathit{vc\text{-}Causes}(D, \mathcal{Q}(\bar{a})) \neq \emptyset$. $\square$

Example 3.1. (ex. 1.1 cont.) Consider the same instance D, query $\mathcal{Q}$, and answer $\langle$ *John, XML*$\rangle$.

Consider the following sets of tuples:

S_1={ *Author(John, TKDE), Journal(TODS, XML, 32)*},

S_2={ *Author(John, TODS), Journal(TKDE, XML, 30)*},

S_3={ *Journal(TODS, XML, 30), Journal(TKDE, XML, 30)*},

S_4={ *Author(John, TODS), Author(John, TKDE)*}.

Each of the subinstances $D \smallsetminus S_i$, $i = 1, \ldots, 4$, is a solution to both the minimum and minimal source-side-effect problems. These solutions essentially contain the actual causes for answer $\langle$ *John, XML*$\rangle$, as computed in Example 1.1. Moreover, there is no solution to the view side-effect-free problem associated to this answer, which coincides with the result obtained in Example 1.2, and confirms Proposition 3.3. $\square$

Now we show, the other way around, that actual causes, most responsible causes, and VC causes can be obtained from solutions to different variants of the delete-propagation problem.

First, we show that actual causes for a query answer can be obtained from the solutions to the minimal source-side-effect problem.

Proposition 3.4. Consider an instance D, a view $\mathcal{V}$ defined by a monotone query $\mathcal{Q}$, and $\bar{a} \in \mathcal{V}(D)$: Tuple t is an actual cause for $\bar{a}$ iff there is a $D' \subseteq D$ with $t \in (D \smallsetminus D') \subseteq D^n$ and $(D, D', \bar{a}) \in \mathcal{MSSEP}^s(\mathcal{Q})$. $\square$

Similarly, most-responsible causes for a query answer can be obtained from solutions to the associated minimum source-side-effect problem.

Proposition 3.5. Consider an instance D, a view $\mathcal{V}$ defined by a monotone query $\mathcal{Q}$, and $\bar{a} \in \mathcal{V}(D)$: Tuple t is a most responsible actual cause for $\bar{a}$ iff there is a $D' \subseteq D$ with $t \in (D \smallsetminus D') \subseteq D^n$ and $(D, D', \bar{a}) \in \mathcal{MSSEP}^c(\mathcal{Q})$. $\square$

Finally, VC-causes for an answer can obtained from solutions to the view side-effect-free problem.

Proposition 3.6. Consider an instance D, a view $\mathcal{V}$ defined by a monotone query $\mathcal{Q}$, and $\bar{a} \in \mathcal{V}(D)$: Tuple t is a VC-cause for $\bar{a}$ iff there is a $D' \subseteq D$ with $t \in (D \smallsetminus D') \subseteq D^n$ and D' is a solution to the view side-effect-free problem associated to $\bar{a}$. $\square$

The partition of a database into endogenous and exogenous tuples used in causality may also be of interest in the context of delete propagation. It makes sense to consider *endogenous delete-propagation* that are obtained through deletions on endogenous tuples only. Actually, given an instance $D = D^n \cup D^x$, a view $\mathcal{V}$ defined by a monotone query $\mathcal{Q}$, and $\bar{a} \in \mathcal{V}(D)$, endogenous delete-propagations for $\bar{a}$ (in all of its flavors) can be obtained from actual causes for $\bar{a}$ from the partitioned instance.

Example 3.2. (ex. 3.1 cont.) Consider again that tuple $\langle$ *John, XML*$\rangle$ must be deleted from the query result; and assume now the data in *Journal* is reliable. Therefore, only deletions from *Author* make sense. This can be captured by considering *Journal*-tuples as exogenous and *Author*-tuples as endogenous. With this partitioning, only *Author(John, TODS)* and *Author(John, TKDE)* are actual causes for $\langle$ *John, XML*$\rangle$, and each of them forms a singleton and unique contingency set of the other as a cause (See Exampleex:cfex1). Therefore, $D \smallsetminus \{$*Author(John, TODS), Author(John, TKDE)*$\}$ is a solution to the associated minimal- and minimum endogenous delete-propagation of $\langle$ *John, XML*$\rangle$. $\square$

4 COMPLEXITY RESULTS

The established connections between abductive diagnoses, query causality and delete-propagation problems allow us to adopt (and possibly adapt) established results for some of them for application to the others.

First, we establish the complexity of causality decision problem (cf. Definition 1.2) for Datalog queries (possibly recursive). For this purpose, we take advantages of the connections in Section 2 between causality and Datalog abduction. In particular, the following is obtained from Propositions 1.1 and 2.1:

Proposition 4.1. For Datalog queries, $\mathcal{CDP}$ is *NP*-complete. $\square$

This result should be contrasted with the tractability of same problem for UCQs (Meliou et al., 2010a; Salimi & Bertossi, 2015).

We now introduce a fixed-parameter tractable case of this problem. For this we take advantage of the tractable case of Datalog abduction presented in Section 1. The following is a consequence of Theorem 1.2 and Proposition 2.1.

Proposition 4.2. For guarded Datalog queries and a set of exogenous tuples with bounded tree-width, $\mathcal{CDP}$ is fixed-parameter tractable in $|D|$, with the parameter being the three-width bound. $\square$

Next, we use the tractability results for causality for UBCQs (Salimi & Bertossi, 2015)) to obtain a tractability result for Datalog abduction, via Proposition 2.3.

Proposition 4.3. For recursion-free Datalog abduction problems deciding if an hypothesis is relevant is tractable. $\square$

We now investigate the complexity of the view-conditioned causality problem (cf. Definition 1.6). For this, we take advantage of the connection between VC-causality and the view side-effect-free problem. Actually, the following result is obtained from the *NP*-completeness of view side-effect-free problem (Buneman et al., 2002) and Proposition 3.3.

Proposition 4.4. For CQs, the view-conditioned causality decision problem, $\mathcal{VCP}$, is *NP*-complete. $\square$

Actually, this result also holds for UCQs. The next result is obtained from the $FP^{NP(log(n))}$-completeness of computing the responsibility of the most responsible causes (obtained in (Salimi & Bertossi, 2015)) and Proposition 3.2.

Proposition 4.5. Computing the size of a solution to the minimum source-side-effect problem (cf. Definition 1.10(a)) is $FP^{NP(log(n))}$-hard. $\square$

As mentioned in Section 1.1, responsibility computation (more precisely the RDP problem in Definition 1.3) is tractable for weakly linear queries. We can take advantage of this result and obtain, via Proposition 3.2, a new tractability result for the minimum source-side-effect problem, which has been shown to be *NP*-hard for general CQs in (Buneman et al., 2002).

Proposition 4.6. For weakly linear queries, the minimum source-side-effect decision problem is tractable. $\square$

The class of weakly linear queries generalizes that of linear queries (cf. Section 1.1). So, Proposition 4.6 also holds for linear queries.

In (Buneman et al., 2002) it has been shown that the minimum source-side-effect decision problem is tractable for the class of project-join queries with *chain joins*. Now, a join on k atoms with different predicates, say $R_1, ..., R_k$, is a chain join if there are no attributes (variables) shared by any two atoms R_i and R_j with $j > i + 1$. That is, only consecutive relations may share attributes. For example, $\exists xvyu(A(x) \wedge S_1(x, v) \wedge S_2(v, y) \wedge R(y, u) \wedge S_3(y, z))$ is a project-join query with chain joins.

We observe that project-join queries with chain joins correspond linear queries. Actually, the tractability results for these classes of queries are both obtained via a reduction to maximum flow problem (Meliou et al., 2010a; Buneman et al., 2002). As a consequence, the result in Proposition 4.6 extends that in (Buneman et al., 2002), from linear queries to weakly-linear queries. For example, $\exists xyz(R(x, y) \wedge S(y, z) \wedge T(z, x) \wedge V(x))$ is not linear (then, nor with chain joins), but it is weakly linear (Meliou et al., 2010a).

5 CONCLUSIONS

We have related query causality to abductive diagnosis and the view-update problem. Some connections between the last two have been established before. More precisely, the view-update problem has been treated from the point of view of abductive reasoning (Kakas & Mancarella, 1990; Console et al., 1995). The idea is to "abduce" the presence of tuples in the base tables that explain the presence of those tuples in the view extension that one would like, e.g. to get rid of.

In combination with the results reported in (Salimi & Bertossi, 2015), we can see that there are deeper and multiple connections between the areas of query causality, abductive and consistency-based diagnosis, view updates, and database repairs. Results for any of these areas can be profitably applied to the others.[10]

We point out that database repairs are related to the view-update problem. Actually, *answer set programs* (ASPs) (Brewka et al., 2011) for database repairs (Bertossi, 2011) implicity repair the database by updating conjunctive combinations of intentional, annotated predicates. Those logical combinations -views after all- capture violations of integrity constraints in the original database or along the (implicitly iterative) repair process (a reason for the use of annotations).

Even more, in (Bertossi & Li, 2013), 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. In order to protect information, a user is allowed to interact only with the virtually repaired versions of the original database that result from making those views empty or contain only null values. Repairs are specified and computed using ASP, and an explicit connection to prioritized attribute-based repairs (Bertossi, 2011) is made (Bertossi & Li, 2013).

[10]Connections between consistency-based and abductive diagnosis have been established, e.g. in (Console & Torasso, 1991).

Finally, we should note that abduction has also been explicitly applied to database repairs (Arieli et al., 2004). The idea, again, is to "abduce" possible repair updates that bring the database to a consistent state.

Acknowledgments: Research funded by NSERC Discovery, and the NSERC Strategic Network on Business Intelligence (BIN).

References

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

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.

Bertossi, L. and Li, L. Achieving Data Privacy through Secrecy Views and Null-Based Virtual Updates. *IEEE Transaction on Knowledge and Data Engineering*, 2013, 25(5):987-1000.

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

Bertossi, L. and Salimi, B. Unifying Causality, Diagnosis, Repairs and View-Updates in Databases. First International PODS-Workshop on Big Uncertain Data (BUDA 2014). `http://www.sigmod2014.org/buda/papers/p5.pdf`

Brewka, G., Eiter, Th. and Truszczynski, M. Answer Set Programming at a Glance. *Communications of the ACM*, 2011, 54(12):92-103.

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

Ceri, S., Gottlob, G. and Tanca, L. *Logic Programming and Databases*. Springer, 1989.

Chockler, H. and Halpern, J. Y. Responsibility and Blame: A Structural-Model Approach. *J. Artif. Intell. Res.*, 2004, 22:93-115.

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

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

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.

Denecker, M., and Kakas A. C. Abduction in Logic Programming. In *Computational Logic: Logic Programming and Beyond*, 2002, LNCS 2407, pp. 402-436.

Eiter, T. and Gottlob, G. The Complexity of Logic-Based Abduction. *J. ACM* , 1995, 42(1): 3-42.

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

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.

Gottlob, G., Pichler, R. and Wei, F. Bounded Treewidth as a Key to Tractability of Knowledge Representation And Reasoning. *Artificial Intelligence*, 2010a, 174(1):105132.

Gottlob, G., Pichler, R. and Wei, F. Tractable Database Design and Datalog Abduction through Bounded Treewidth. *Inf. Syst.*, 2010b, 35(3):278-298.

Halpern, J., and Pearl, J. Causes and Explanations: A Structural-Model Approach: Part 1 Proc. UAI, 2001, pp. 194-202.

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

Halpern, J. A Modification of Halpern-Pearl Definition of Causality. To appear in Proc. IJCAI, 2015.

Halpern, J. Appropriate Causal Models and Stability of Causation. Proc. KR, 2014.

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

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

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

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

Meliou, A., Gatterbauer. W., Halpern, J. Y., Koch, C., Moore K. F. and Suciu, D. Causality in Databases. *IEEE Data Eng. Bull*, 2010b, 33(3):59-67.

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

Psillos., A. Ampliative Reasoning: Induction or Abduction. Proc. ECAI'96 Workshop on Abductive and Inductive Reasoning, 1996.

Poole, D. Logic Programming, Abduction and Probability. Proc. FGCS, 1992, pp. 530-538.

Poole, D. Representing Diagnosis Knowledge. *Annals of Mathematics and Artificial Intelligence*, 1994, 11(1-4):33-50.

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

Salimi, B. and Bertossi, L. Causality in Databases: The Diagnosis and Repair Connections. Proc. 15th International Workshop on Non-Monotonic Reasoning (NMR 2014), 2014. Corr Arkiv Paper cs.DB/1404.6857.

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

Struss, P. Model-based Problem Solving. In *Handbook of Knowledge Representation*, chap. 10. Elsevier, 2008.