The Causality/Repair Connection in Databases: Causality-Programs

Leopoldo Bertossi⋆

Carleton University, School of Computer Science, Ottawa, Canada.

Abstract. In this work, answer-set programs that specify repairs of databases are used as a basis for solving computational and reasoning problems about causes for query answers from databases.

1 Introduction

Causality appears at the foundations of many scientific disciplines. In data and knowledge management, the need to represent and compute *causes* may be related to some form of *uncertainty* about the information at hand. More specifically in data management, we need to understand why certain results, e.g. query answers, are obtained or not. Or why certain natural semantic conditions are not satisfied. These tasks become more prominent and difficult when dealing with large volumes of data. One would expect the database to provide *explanations*, to understand, explore and make sense of the data, or to reconsider queries and integrity constraints (ICs). Causes for data phenomena can be seen as a kind of explanations.

Seminal work on *causality in DBs* introduced in [17], and building on work on causality as found in artificial intelligence, appeals to the notions of counterfactuals, interventions and structural models [15]. Actually, [17] introduces the notions of: (a) a DB tuple as an *actual cause* for a query result, (b) a *contingency set* for a cause, as a set of tuples that must accompany the cause for it to be such, and (c) the *responsibility* of a cause as a numerical measure of its strength (building on [11]).

Most of our research on causality in DBs has been motivated by an attempt to understand causality from different angles of data and knowledge management. In [6], precise reductions between causality in DBs, DB repairs, and consistency-based diagnosis were established; and the relationships where investigated and exploited. In [7], causality in DBs was related to view-based DB updates and abductive diagnosis. These are all interesting and fruitful connections among several forms of non-monotonic reasoning; each of them reflecting some form of uncertainty about the information at hand. In the case of DB repairs [3], it is about the uncertainty due the non-satisfaction of given ICs, which is represented by presence of possibly multiple intended repairs of the inconsistent DB.

DB repairs can be specified by means of *answer-set programs* (or *disjunctive logic programs with stable model semantics*) [14], the so-called *repair-programs*. Cf. [10, 3] for repair-programs and additional references. In this work we exploit the reduction of DB causality to DB repairs established in [6], by taking advantage of repair programs for specifying and computing causes, their contingency sets, and their responsibility degrees. We show that that the resulting *causality-programs* have the necessary and sufficient expressive power to capture and compute not only causes, which can be done with

⋆ Email: bertossi@scs.carleton.ca. Research supported by NSERC Discovery Grant #06148.

less expressive programs [17], but specially minimal contingency sets and responsibilities (which can not). Causality programs can also be used for reasoning about causes. Finally, we briefly show how causality-programs can be adapted to give an account of other forms of causality in DBs.

2 Background

Relational DBs. A relational schema $\mathcal{R}$ contains a domain, $\mathcal{C}$, of constants and a set, $\mathcal{P}$, of predicates of finite arities. $\mathcal{R}$ gives rise to a language $\mathfrak{L}(\mathcal{R})$ of first-order (FO) predicate logic with built-in equality, $=$. Variables are usually denoted by $x, y, z, ...$, and sequences thereof by $\bar{x}, ...$; and constants with $a, b, c, ...$, etc. An *atom* is of the form $P(t_1, \ldots, t_n)$, with n-ary $P \in \mathcal{P}$ and $t_1, \ldots, t_n$ *terms*, i.e. constants, or variables. An atom is *ground* (aka. a tuple) if it contains no variables. A DB instance, D, for $\mathcal{R}$ is a finite set of ground atoms; and it serves as an interpretation structure for $\mathfrak{L}(\mathcal{R})$.

A *conjunctive query* (CQ) is a FO formula, $\mathcal{Q}(\bar{x})$, of the form $\exists \bar{y}\ (P_1(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m))$, with $P_i \in \mathcal{P}$, and (distinct) free variables $\bar{x} := (\bigcup \bar{x}_i) \smallsetminus \bar{y}$. If $\mathcal{Q}$ has n (free) variables, $\bar{c} \in \mathcal{C}^n$ is an *answer* to $\mathcal{Q}$ from D if $D \models \mathcal{Q}[\bar{c}]$, i.e. $Q[\bar{c}]$ is true in D when the variables in $\bar{x}$ are componentwise replaced by the values in $\bar{c}$. $\mathcal{Q}(D)$ denotes the set of answers to $\mathcal{Q}$ from D D. $\mathcal{Q}$ is a *boolean conjunctive query* (BCQ) when $\bar{x}$ is empty; and when *true* in D, $\mathcal{Q}(D) := \{true\}$. Otherwise, it is *false*, and $\mathcal{Q}(D) := \emptyset$.

In this work we consider integrity constraints (ICs), i.e. sentences of $\mathfrak{L}(\mathcal{R})$, that are: (a) *denial constraints* (DCs), i.e. of the form $\kappa\colon \neg\exists \bar{x}(P_1(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m))$, where $P_i \in \mathcal{P}$, and $\bar{x} = \bigcup \bar{x}_i$; and (b) *functional dependencies* (FDs), i.e. of the form $\varphi\colon \neg\exists \bar{x}(P(\bar{v}, \bar{y}_1, z_1) \wedge P(\bar{v}, \bar{y}_2, z_2) \wedge z_1 \neq z_2)$. Here, $\bar{x} = \bar{y}_1 \cup \bar{y}_2 \cup \bar{v} \cup \{z_1, z_2\}$, and $z_1 \neq z_2$ is an abbreviation for $\neg z_1 = z_2$.[1] A *key constraint* (KC) is a conjunction of FDs: $\bigwedge_{j=1}^{k} \neg\exists \bar{x}(P(\bar{v}, \bar{y}_1) \wedge P(\bar{v}, \bar{y}_2) \wedge y_1^j \neq y_2^j)$, with $k = |\bar{y}_1| = |\bar{y}_2|$. A given schema may come with its set of ICs, and its instances are expected to satisfy them. If this is not the case, we say the instance is *inconsistent*.

Causality in DBs. A notion of *cause* as an explanation for a query result was introduced in [17], as follows. For a relational instance $D = D^n \cup D^x$, where D^n and D^x denote the mutually exclusive sets of endogenous and exogenous tuples, a tuple $\tau \in D^n$ is called a *counterfactual cause* for a BCQ $\mathcal{Q}$, if $D \models \mathcal{Q}$ and $D \smallsetminus \{\tau\} \not\models \mathcal{Q}$. Now, $\tau \in D^n$ is an *actual cause* for $\mathcal{Q}$ if there exists $\Gamma \subseteq D^n$, called a *contingency set* for τ, such that τ is a counterfactual cause for $\mathcal{Q}$ in $D \smallsetminus \Gamma$. This definition is based on [15].

The notion of *responsibility* reflects the relative degree of causality of a tuple for a query result [17] (based on [11]). The responsibility of an actual cause τ for $\mathcal{Q}$, is $\rho(\tau) := \frac{1}{|\Gamma|+1}$, where $|\Gamma|$ is the size of a smallest contingency set for τ. If τ is not an actual cause, $\rho(\tau) := 0$. Tuples with higher responsibility are stronger explanations.

In the following we will assume all the tuples in a DB instance are endogenous. (Cf. [6] for the general case.) The notion of cause as defined above can be applied to monotonic queries, i.e whose sets of answers may only grow when the DB grows [6].[2] *In this work we concentrate only on conjunctive queries, possibly with* $\neq$.

[1] The variables in the atoms do not have to occur in the indicated order, but their positions should be in correspondence in the two atoms.

[2] E.g. CQs, unions of CQs (UCQs), Datalog queries are monotonic.

Example 1. Consider the relational DB $D = \{R(a_4, a_3), R(a_2, a_1), R(a_3, a_3), S(a_4), S(a_2), S(a_3)\}$, and the query $\mathcal{Q}\colon\ \exists x \exists y(S(x) \land R(x, y) \land S(y))$. It holds, $D \models \mathcal{Q}$.

$S(a_3)$ is a counterfactual cause for $\mathcal{Q}$: if $S(a_3)$ is removed from D, $\mathcal{Q}$ is no longer true. Its responsibility is 1. So, it is an actual cause with empty contingency set. $R(a_4, a_3)$ is an actual cause for $\mathcal{Q}$ with contingency set $\{R(a_3, a_3)\}$: if $R(a_3, a_3)$ is removed from D, $\mathcal{Q}$ is still true, but further removing $R(a_4, a_3)$ makes $\mathcal{Q}$ false. The responsibility of $R(a_4, a_3)$ is $\frac{1}{2}$. $R(a_3, a_3)$ and $S(a_4)$ are actual causes, with responsibility $\frac{1}{2}$. □

Database repairs. Cf. [3] for a survey on DB repairs and consistent query answering in DBs. We introduce the main ideas by means of an example. The ICs we consider in this work can be enforced only by deleting tuples from the DB (as opposed to inserting tuples). Repairing the DB by changing attribute values is also possible [3, 4, 5], [6, sec. 7.4], but until further notice we will not consider this kind of repairs.

Example 2. The DB $D = \{P(a), P(e), Q(a, b), R(a, c)\}$ is inconsistent with respect to the (set of) *denial constraints* (DCs) $\kappa_1\colon\ \neg\exists x \exists y(P(x) \land Q(x, y))$, and $\kappa_2\colon\ \neg\exists x \exists y(P(x) \land R(x, y))$. It holds $D \not\models \{\kappa_1, \kappa_2\}$.

A *subset-repair*, in short an *S-repair*, of D wrt. the set of DCs is a $\subseteq$-maximal subset of D that is consistent, i.e. no proper superset is consistent. The following are S-repairs: $D_1 = \{P(e), Q(a, b), R(a, b)\}$ and $D_2 = \{P(e), P(a)\}$. A *cardinality-repair*, in short a *C-repair*, of D wrt. the set of DCs is a maximum-cardinality, consistent subset of D, i.e. no subset of D with larger cardinality is consistent. D_1 is the only C-repair. □

For an instance D and a set Σ of DCs, the sets of S-repairs and C-repairs are denoted with $\mathit{Srep}(D, \Sigma)$ and $\mathit{Crep}(D, \Sigma)$, resp.

3 Causality Answer Set Programs

Causes from repairs. In [6] it was shown that causes for queries can be obtained from DB repairs. Consider the BCQ $\mathcal{Q}\colon \exists \bar{x}(P_1(\bar{x}_1) \land \cdots \land P_m(\bar{x}_m))$ that is (possibly unexpectedly) true in D: $D \models \mathcal{Q}$. Actual causes for $\mathcal{Q}$, their contingency sets, and responsibilities can be obtained from DB repairs. First, $\neg\mathcal{Q}$ is logically equivalent to the DC:

$$\kappa(\mathcal{Q})\colon\ \neg\exists \bar{x}(P_1(\bar{x}_1) \land \cdots \land P_m(\bar{x}_m)). \tag{1}$$

So, if $\mathcal{Q}$ is true in D, D is inconsistent wrt. $\kappa(\mathcal{Q})$, giving rise to repairs of D wrt. $\kappa(\mathcal{Q})$.

Next, we build differences, containing a tuple τ, between D and S- or C-repairs:

(a) $\mathit{Dif}^s(D, \kappa(\mathcal{Q}), \tau) = \{D \smallsetminus D' \mid D' \in \mathit{Srep}(D, \kappa(\mathcal{Q})),\ \tau \in (D \smallsetminus D')\}$, (2)

(b) $\mathit{Dif}^c(D, \kappa(\mathcal{Q}), \tau) = \{D \smallsetminus D' \mid D' \in \mathit{Crep}(D, \kappa(\mathcal{Q})),\ \tau \in (D \smallsetminus D')\}$. (3)

It holds [6]: $\tau \in D$ is an actual cause for $\mathcal{Q}$ iff $\mathit{Dif}^s(D, \kappa(\mathcal{Q}), \tau) \neq \emptyset$. Furthermore, each S-repair D' for which $(D \smallsetminus D') \in \mathit{Dif}^s(D, \kappa(\mathcal{Q}), \tau)$ gives us $(D \smallsetminus (D' \cup \{\tau\}))$ as a subset-minimal contingency set for τ. Also, if $\mathit{Dif}^s(D\ \kappa(\mathcal{Q}), \tau) = \emptyset$, then $\rho(\tau) = 0$. Otherwise, $\rho(\tau) = \frac{1}{|s|}$, where $s \in \mathit{Dif}^s(D,\ \kappa(\mathcal{Q}), \tau)$ and there is no $s' \in \mathit{Dif}^s(D, \kappa(\mathcal{Q}), \tau)$ with $|s'| < |s|$. As a consequence we obtain that τ is a most responsible actual cause for $\mathcal{Q}$ iff $\mathit{Dif}^c(D, \kappa(\mathcal{Q}), \tau) \neq \emptyset$.

Example 3. (ex. 1 cont.) With the same instance D and query $\mathcal{Q}$, we consider the DC $\kappa(\mathcal{Q})$: $\neg\exists x \exists y(S(x) \land R(x, y) \land S(y))$, which is not satisfied by D. Here, $\mathit{Srep}(D, \kappa(\mathcal{Q})) = \{D_1, D_2, D_3\}$ and $\mathit{Crep}(D, \kappa(\mathcal{Q})) = \{D_1\}$, with $D_1 = \{R(a_4, a_3),$

$R(a_2,a_1), R(a_3,a_3), S(a_4), S(a_2)\}$, $D_2 = \{R(a_2,a_1), S(a_4), S(a_2), S(a_3)\}$, $D_3 = \{R(a_4,a_3), R(a_2,a_1), S(a_2), S(a_3)\}$.

For tuple $R(a_4,a_3)$, $\mathit{Dif}^s(D,\kappa(\mathcal{Q}), R(a_4,a_3)) = \{D \smallsetminus D_2\} = \{\{R(a_4,a_3), R(a_3,a_3)\}\}$. So, $R(a_4,a_3)$ is an actual cause, with responsibility $\frac{1}{2}$. Similarly, $R(a_3,a_3)$ is an actual cause, with responsibility $\frac{1}{2}$. For tuple $S(a_3)$, $\mathit{Dif}^c(D,\kappa(\mathcal{Q}), S(a_3)) = \{D \smallsetminus D_1\} = \{S(a_3)\}$. So, $S(a_3)$ is an actual cause, with responsibility 1, i.e. a most responsible cause. □

It is also possible, the other way around, to characterize repairs in terms of causes and their contingency sets. Actually this connection can be used to obtain complexity results for causality problems from repair-related computational problems [6]. Most computational problems related to repairs, specially C-repairs, which are related to most responsible causes, are provably hard. This is reflected in a high complexity for responsibility [6] (see below for some more details).

Answer-set programs for repairs. Given a DB D and a set of ICs, Σ, it is possible to specify the repairs of D wrt. Σ by means of an answer-set program (ASP) $\Pi(D,\Sigma)$, in the sense that the set, $\mathit{Mod}(\Pi(D,\Sigma))$, of its stable models is in one-to-one correspondence with $\mathit{Srep}(D,\Sigma)$ [10, 2] (cf. [3] for more references). In the following we consider a single denial constraint $\kappa\colon \neg\exists\bar{x}(P_1(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m))$.[3]

Although not necessary for repair purposes, it may be useful on the causality side having global unique tuple identifiers (tids), i.e. every tuple $R(\bar{c})$ in D is represented as $R(t,\bar{c})$ for some integer t that is not used by any other tuple in D. For the repair program we introduce a nickname predicate R' for every predicate $R \in \mathcal{R}$ that has an extra, final attribute to hold an annotation from the set $\{\mathsf{d},\mathsf{s}\}$, for "delete" and "stays", resp. Nickname predicates are used to represent and compute repairs.

The *repair-ASP*, $\Pi(D,\kappa)$, for D and κ contains all the tuples in D as facts (with tids), plus the following rules:

$$P_1'(t_1,\bar{x}_1,\mathsf{d}) \vee \cdots \vee P_m'(t_n,\bar{x}_m,\mathsf{d}) \leftarrow P_1(t_1,\bar{x}_1),\ldots,P_m(t_m,\bar{x}_m),$$
$$P_i'(t_i,\bar{x}_i,\mathsf{s}) \leftarrow P_i(t_i,\bar{x}_i),\ \mathit{not}\ P_i'(t_i,\bar{x}_i,\mathsf{d}),\ i = 1,\cdots,m.$$

A stable model M of the program determines a repair D' of D: $D' := \{P(\bar{c}) \mid P'(t,\bar{c},\mathsf{s}) \in M\}$, and every repair can be obtained in this way [10]. For an FD, say $\varphi\colon\ \neg\exists xyz_1z_2vw(R(x,y,z_1,v) \wedge R(x,y,z_2,w) \wedge z_1 \neq z_2)$, which makes the third attribute functionally depend upon the first two, the repair program contains the rules:

$$R'(t_1,x,y,z_1,v,\mathsf{d}) \vee R'(t_2,x,y,z_2,w,\mathsf{d}) \leftarrow R(t_1,x,y,z_1,v), R(t_2,x,y,z_2,w), z_1 \neq z_2.$$
$$R'(t,x,y,z,v,\mathsf{s}) \leftarrow R(t,x,y,z,v),\ \mathit{not}\ R'(t,x,y,z,v,\mathsf{d}).$$

For DCs and FDs, the repair program can be made non-disjunctive by moving all the disjuncts but one, in turns, in negated form to the body of the rule [10, 2]. For example, the rule $P(a) \vee R(b) \leftarrow \mathit{Body}$, can be written as the two rules $P(a) \leftarrow \mathit{Body}, \mathit{not}R(b)$ and $R(b) \leftarrow \mathit{Body}, \mathit{not}P(a)$. Still the resulting program can be *non-stratified* if there is recursion via negation [14], as in the case of FDs and DCs with self-joins.

Example 4. (ex. 3 cont.) For the DC $\kappa(\mathcal{Q})$: $\neg\exists x\exists y(S(x) \wedge R(x,y) \wedge S(y))$, the repair-ASP contains the facts (with tids) $R(1,a_4,a_3), R(2,a_2,a_1), R(3,a_3,a_3)$, $S(4,a_4)$, $S(5,a_2), S(6,a_3)$, and the rules:

[3] It is possible to consider a combination of several DCs and FDs, corresponding to UCQs (possibly with $\neq$), on the causality side [6].

$$S'(t_1, x, \mathsf{d}) \vee R'(t_2, x, y, \mathsf{d}) \vee S'(t_3, y, \mathsf{d}) \leftarrow S(t_1, x), R(t_2, x, y), S(t_3, y),$$
$$S'(t, x, \mathsf{s}) \leftarrow S(t, x),\ \mathit{not}\ S'(t, x, \mathsf{d}). \quad \text{etc.}$$

Repair D_1 is represented by the stable model M_1 containing $R'(1, a_4, a_3, \mathsf{s})$, $R'(2, a_2, a_1, \mathsf{s})$, $R'(3, a_3, a_3, \mathsf{s})$, $S'(4, a_4, \mathsf{s})$, $S'(5, a_2, \mathsf{s})$, and $S'(6, a_3, \mathsf{d})$. □

Specifying causes with repair-ASPs. According to (2), we concentrate on the differences between the D and its repairs, now represented by $\{P(\bar{c}) \mid P(t, \bar{c}, \mathsf{d}) \in M\}$, for M a stable model of the repair-program. They are used to compute actual causes and their $\subseteq$-minimal contingency sets, both identified by tids. So, given the repair-ASP for a DC $\kappa(\mathcal{Q})$, a binary predicate $\mathit{Cause}(\cdot, \cdot)$ will contain a tid for cause in its first argument, and a tid for a tuple belonging to its contingency set. Intuitively, $\mathit{Cause}(t, t')$ says that t is an actual cause, and t' accompanies t as a member of the former's contingency set (as captured by the repair at hand or, equivalently, by the corresponding stable model). More precisely, for each pair of predicates P_i, P_j in the DC $\kappa(\mathcal{Q})$ as in (1) (they could be the same if it has self-joins), introduce the rule $\mathit{Cause}(t, t') \leftarrow P_i'(t, \bar{x}_i, \mathsf{d}), P_j'(t', \bar{x}_j, \mathsf{d}), t \neq t'$, with the inequality condition only when P_i and P_j are the same.

Example 5. (ex. 3 and 4 cont.) The causes for the query, represented by their tids, can be obtained by posing simple queries to the program under the *uncertain or brave* semantics that makes true what is true in *some* model of the repair-ASP.[4] In this case, $\Pi(D, \kappa(\mathcal{Q})) \models_{brave} \mathit{Ans}(t)$, where the auxiliary predicate is defined on top of $\Pi(D, \kappa(\mathcal{Q}))$ by the rules: $\mathit{Ans}(t) \leftarrow R'(t, x, y, \mathsf{d})$ and $\mathit{Ans}(t) \leftarrow S'(t, x, \mathsf{d})$.

The repair-ASP can be extended with the following rules to compute causes with contingency sets:

$$\mathit{Cause}(t, t') \leftarrow S'(t, x, \mathsf{d}), R'(t', u, v, \mathsf{d}),$$
$$\mathit{Cause}(t, t') \leftarrow S'(t, x, \mathsf{d}), S'(t', u, \mathsf{d}), t \neq t',$$
$$\mathit{Cause}(t, t') \leftarrow R'(t, x, y, \mathsf{d}), S'(t', u, \mathsf{d}).$$

For the stable model M_2 corresponding to repair D_2, we obtain $\mathit{Cause}(1, 3)$ and $\mathit{Cause}(3, 1)$, from the repair difference $D \smallsetminus D_2 = \{R(a_4, a_3), R(a_3, a_3)\}$. □

We can use the DLV system [16] to build the contingency set associated to a cause, by means of its extension, DLV-Complex [9], that supports set building, membership and union, as built-ins. For every atom $\mathit{Cause}(t, t')$, we introduce the atom $\mathit{Con}(t, \{t'\})$, and the rule that computes the union of (partial) contingency sets as long as they differ by some element:

$$\mathit{Con}(T, \#\mathit{union}(C_1, C_2)) \leftarrow \mathit{Con}(T, C_1), \mathit{Con}(T, C_2), \#\mathit{member}(M, C_1),$$
$$\mathit{not}\ \#\mathit{member}(M, C_2).$$

The responsibility for an actual cause τ, with tid t, as associated to a given repair D' (with $\tau \notin D'$), and then to a given model M' of the extended repair-ASP, can be computed by counting the number of t's for which $\mathit{Cause}(t, t') \in M'$. This responsibility will be maximum within a repair (or model): $\rho(t, M') := 1/(1 + |d(t, M')|)$, where $d(t, M') := \{\mathit{Cause}(t, t') \in M'\}$. This value can be computed by means of the *count* function, supported by DLV [13], as follows: $\mathit{pre\text{-}rho}(T, N) \leftarrow \#\mathit{count}\{T' : \mathit{Con}(T, T')\} = N$, followed by the rule computing the responsibility:

[4] As opposed to the *skeptical or cautious* semantics that sanctions as true what is true in *all* models. Both semantics as supported by the DLV system [16], to which we refer below.

$rho(T, M) \leftarrow M * (pre\text{-}rho(T, M) + 1) = 1$. Or equivalently, via $1/|d(M)|$, with $d(M') := \{P(t', \bar{c}, \mathsf{d}) \mid P(t', \bar{c}, \mathsf{d}) \in M'\}$.

Each model M of the program so far will return, for a given tuple (id) that is an actual cause, a *maximal-responsibility contingency set within that model*: no proper subset is a contingency set for the given cause. However, its cardinality may not correspond to the (global) *maximum* responsibility for that tuple. For that we need to compute only maximum-cardinality repairs, i.e. C-repairs.

C-repairs can be specified by means of repair-ASPs [1] that contain *weak-program constraints* [8, 13]. In this case, we want repairs that minimize the number of deleted tuples. For each DB predicate P, we introduce the weak-constraint[5] $\Leftarrow P(t, \bar{x}), P'(t, \bar{x}, \mathsf{d})$. In a model M the body can be satisfied, and then the program constraint violated, but the number of violations is kept to a minimum (among the models of the program without the weak-constraints). A repair-ASP with these weak constraints specifies repairs that minimize the number of deleted tuples; and *minimum-cardinality* contingency sets and maximum responsibilities can be computed, as above.

Complexity. Computing causes for CQs can be done in polynomial time in data [17], which was extended to UCQs in [6]. As has been established in [17, 6], the computational problems associated to contingency sets and responsibility are in the second level of the polynomial hierarchy (PH), in data complexity [12]. On the other side, our causality-ASPs can be transformed into non-disjunctive, unstratified programs, whose reasoning tasks are also in the second level of the PH (in data). It is worth mentioning that the ASP approach to causality via repairs programs could be extended to deal with queries that are more complex than CQs or UCQs. (In [18] causality for queries that are conjunctions of literals was investigated; and in [7] it was established that cause computation for Datalog queries can be in the second level of the PH.)

Causality programs and ICs The original causality setting in [17] does not consider ICs. An extension of causality under ICs was proposed in [7]. Under it, the ICs have to be satisfied by the DBs involved, i.e. the initial one and those obtained by cause- and contingency-set deletions. When the query at hand is monotonic[6], monotonic ICs (e.g. denial constraints and FDs) are not much of an issue since they stay satisfied under deletions associated to causes. So, the most relevant ICs are non-monotonic, such as referential ICs, e.g. $\forall xy(R(x, y) \rightarrow S(x))$ in our running example. These ICs can be represented in a causality-program by means of (strong) program constraints. In the running example, we would have, for example, the constraint: $\leftarrow R'(t, x, y, \mathsf{s}), \mathit{not}\ S'(t', x, \mathsf{s})$.[7]

Preferred causes and repairs. In [6], generalized causes were introduced on the basis of arbitrary repair semantics (i.e. classes of preferred consistent subinstances, commonly under some maximality criterion), basically starting from the characterization in (2) and (3), but using repairs of D wrt. $\kappa(\mathcal{Q})$ in a class, $Rep(D, \kappa(\mathcal{Q}))$, possibly different from $Srep(D, \kappa(\mathcal{Q}))$ or $Crep(D, \kappa(\mathcal{Q}))$. As a particular case in [6], *causes based on changes of attribute values* (as opposed to tuple deletions) were defined. In that case, admissible

[5] Hard program-constraints, of the form $\leftarrow Body$, eliminate the models where they are violated.

[6] I.e. the set of answers may only grow when the instance grows.

[7] Or better, to make it *safe*, by a rule and a constraint: $aux(x) \leftarrow S'(t', x, \mathsf{s})$ and $\leftarrow R'(t, x, y, \mathsf{s}), \mathit{not}\ aux(x)$.

updates are replacements of data values by null values, to break joins, in a minimal or minimum way. Those underlying DB repairs were used in [4] to hide sensitive data that could be exposed through CQ answering; and corresponding repair programs were introduced. They could be used, as done earlier in this paper, as a basis to reason about- and compute the new resulting causes (at the tuple or attribute-value level) and their contingency sets.[8]

References

[1] Arenas, M., Bertossi, L., Chomicki, J. Answer Sets for Consistent Query Answers. *Theory and Practice of Logic Programming*, 2003, 3(4&5):393-424.

[2] Barcelo, P., Bertossi, L. and Bravo, L. Characterizing and Computing Semantically Correct Answers from Databases with Annotated Logic and Answer Sets. In *Semantics in Databases*, Springer LNCS 2582, 2003, pp. 7-33.

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

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

[5] Bertossi, L. and Bravo, L. Consistency and Trust in Peer Data Exchange Systems. *Theory and Practice of Logic Programming*, 2017, 17(2):148-204.

[6] Bertossi, L. and Salimi, B. From Causes for Database Queries to Repairs and Model-Based Diagnosis and Back. *Theory of Computing Systems*, 2017, 61(1):191232.

[7] Bertossi, L. and Salimi, B. Causes for Query Answers from Databases: Datalog Abduction, View-Updates, and Integrity Constraints. To appear in *Int. J. Approximate Reasoning*. Corr Arxiv Paper cs.DB/1611.01711.

[8] Buccafurri, F., Leone, N. and Rullo, P. Enhancing Disjunctive Datalog by Constraints. *IEEE Tran. Knowledge and Data Engineering*, 2000, 12, 5, 845860.

[9] Calimeri, F. Cozza, S. Ianni, G. and Leone, N. An ASP System with Functions, Lists, and Sets. Proc. LPNMR 2009, Springer LNCS 5753, 2009, pp. 483-489.

[10] Caniupan-Marileo, M. and Bertossi, L. The Consistency Extractor System: Answer Set Programs for Consistent Query Answering in Databases". *Data & Know. Eng.*, 2010, 69(6):545-572.

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

[12] Dantsin, E., Eiter, T., Gottlob, G. and Voronkov, A. Complexity and Expressive Power of Logic Programming, *ACM Computing Surveys*, 2001, 33(3):374425.

[13] Faber, W., Pfeifer, G., Leone, N., Dell'Armi, T. and Ielpa, G. Design and implementation of aggregate functions in the DLV system. *The. Pr. Logic Prog.*, 2008, 8(5-6):545-580.

[14] Gelfond, M. and Kahl, Y. *Knowledge Representation and Reasoning, and the Design of Intelligent Agents*. Cambridge Univ. Press, 2014.

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

[16] Leone, N., Pfeifer, G., Faber,W., Eiter, T., Gottlob, G., Perri, S. and Scarcello, F. The DLV System for Knowledge Representation and Reasoning. *ACM Trans. Comput. Logic.*, 2006, 7(3):499-562.

[17] 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.

[18] Salimi, B., Bertossi, L., Suciu, D. and Van den Broeck, G. Quantifying Causal Effects on Query Answering in Databases. Proc. TaPP, 2016.

[8] Cf. also [5] for an alternative null-based repair semantics and its repair programs.