From Database Repairs to Causality in Databases and Beyond

Leopoldo Bertossi(✉)

SKEMA Business School, Montreal, Canada
leopoldo.bertossi@skema.edu

Abstract. We describe some recent approaches to score-based explanations for query answers in databases. The focus is on work done by the author and collaborators. Special emphasis is placed on the use of counterfactual reasoning for score specification and computation. Several examples that illustrate the flexibility of these methods are shown.

1 Introduction

In data management one wants *explanations* for certain results. For example, for query results from databases. Explanations, that may come in different forms, have been the subject of philosophical enquires for a long time, but, closer to our discipline, they appear under different forms in model-based diagnosis and in causality as developed in artificial intelligence.

In the last few years, explanations that are based on *numerical scores* assigned to elements of a model that may contribute to an outcome have become popular. These scores attempt to capture the degree of contribution of those components to an outcome, e.g. answering questions like these: What is the contribution of this tuple to the answer to this query?

Different scores have been proposed in the literature, and some that have a relatively older history have been applied. Among the latter we find the general *responsibility score* as found in *actual causality* [11,14]. For a particular kind of application, one has to define the right causality setting, and then apply the responsibility measure to the participating variables (see [15] for an updated treatment of causal responsibility).

In data management, responsibility has been used to quantify the strength of a tuple as a cause for a query result [4,23] (see Sect. 3.1). The *responsibility score*, *Resp*, is based on the notions of *counterfactual intervention* as appearing in actual causality. More specifically, (potential) executions of *counterfactual interventions* on a *structural logico-probabilistic model* [14] are investigated, with the purpose of answering hypothetical questions of the form: *What would happen if we change ...?*.

Database repairs are commonly used to define and obtain semantically correct query answers from a database that may fail to satisfy a given set of integrity constraints (ICs) [3]. A connection between repairs and actual causality in DBs has been used to obtain complexity results and algorithms for the latter [4] (see Sect. 5).

L. Bertossi—Member of the Millennium Inst. for Foundational Research on Data (IMFD, Chile).

A. Hameurlain et al. (Eds.): *TLDKS LIV*, LNCS 14160, pp. 119–131, 2023.
https://doi.org/10.1007/978-3-662-68014-8_5

The *Causal Effect* score is also based on causality, mainly for *observational studies* [16,26,29]. It has been applied in data management in [30] (see Sect. 3.2).

The Shapley value, as found in *coalition game theory* [31], has been used for the same purpose [18,19]. Defining the right game function, the *Shapley value* assigned to a player reflects its contribution to the wealth function. The Shapley value, which is firmly established in game theory, and is also used in several other areas [28,31]. The main idea is that *several tuples together*, much like players in a coalition game, are necessary to produce a query result. Some may contribute more than others to the *wealth distribution function* (or simply, game function), which in this case becomes the query result, namely 1 or 0 if the query is Boolean, or a number if we have an aggregation query. This use of Shapley value was developed in [18,19] (see Sect. 6).

In this article we survey some of the recent advances on the use and computation of the above mentioned score-based explanations for query answering in databases. This is not intended to be an exhaustive survey of the area. Instead, it is heavily influenced by our latest research. To introduce the concepts and techniques we will use mostly examples, trying to convey the main intuitions and issues.

This paper is structured as follows. In Sect. 2, we provide some preliminaries on databases. In Sect. 3, we introduce causality in databases and the responsibility score, and also the causal effect score. In Sect. 4, we show the connection between causality in databases and database repairs. In Sect. 5, we show how integrate ICs in the causality setting. In Sect. 6, we show how to use the Shapley value to provide explanation scores to database tuples in relation to a query result. In Sect. 7, we make some general remarks on relevant open problems.

2 Background

A relational schema $\mathcal{R}$ contains a domain of constants, $\mathcal{C}$, and a set of predicates of finite arities, $\mathcal{P}$. $\mathcal{R}$ gives rise to a language $\mathfrak{L}(\mathcal{R})$ of first-order (FO) predicate logic with built-in equality, $=$. Variables are usually denoted with $x, y, z, ...$; and finite sequences thereof with $\bar{x}, ...$; 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* (a.k.a. a tuple) if it contains no variables. A database (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. $\mathcal{Q}$ is a *Boolean conjunctive query* (BCQ) when $\bar{x}$ is empty; and when *true* in D, $\mathcal{Q}(D) := \{\mathit{true}\}$. Otherwise, it is *false*, and $\mathcal{Q}(D) := \emptyset$. We will consider only conjunctive queries or disjunctions thereof.

We consider as integrity constraints (ICs), i.e. sentences of $\mathfrak{L}(\mathcal{R})$: (a) *denial constraints* (DCs), i.e. of the form $\kappa : \neg\exists\bar{x}(P_1(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m))$, where $P_i \in \mathcal{P}$, and $\bar{x} = \bigcup \bar{x}_i$; and (b) *inclusion dependencies* (INDs), which are of the form $\forall\bar{x}\exists\bar{y}(P_1(\bar{x}) \rightarrow P_2(\bar{x}', \bar{y}))$, where $P_1, P_2 \in \mathcal{P}$, and $\bar{x}' \subseteq \bar{x}$. If an instance D does not satisfy the set Σ of ICs associated to the schema, we say that D is *inconsistent*, denoted with $D \not\models \Sigma$.

3 Causal Explanations in Databases

In data management we need to understand and compute *why* certain results are obtained or not, e.g. query answers, violations of semantic conditions, etc.; and we expect a database system to provide *explanations*.

3.1 Causal Responsibility

Here, we will consider *causality-based explanations* [23,24], which we will illustrate by means of an example.

Example 1. Consider the database D, and the Boolean conjunctive query (BCQ)

R	A	B
	a	b
	c	d
	b	b

S	C
	a
	c
	b

$\mathcal{Q}: \ \exists x \exists y (S(x) \wedge R(x,y) \wedge S(y))$, for which $D \models \mathcal{Q}$ holds, i.e. the query is true in D. We ask about the causes for $\mathcal{Q}$ to be true.

A tuple $\tau \in D$ is *counterfactual cause* for $\mathcal{Q}$ (being true in D) if $D \models \mathcal{Q}$ and $D \smallsetminus \{\tau\} \not\models \mathcal{Q}$. In this example, $S(b)$ is a counterfactual cause for $\mathcal{Q}$: If $S(b)$ is removed from D, $\mathcal{Q}$ is no longer true.

Removing a single tuple may not be enough to invalidate the query. Accordingly, a tuple $\tau \in D$ is an *actual cause* for $\mathcal{Q}$ if there is a *contingency set* $\Gamma \subseteq D$, such that τ is a counterfactual cause for $\mathcal{Q}$ in $D \smallsetminus \Gamma$. In this example, $R(a,b)$ is not a counterfactual cause for $\mathcal{Q}$, but it is an actual cause with contingency set $\{R(b,b)\}$: If $R(b,b)$ is removed from D, $\mathcal{Q}$ is still true, but further removing $R(a,b)$ makes $\mathcal{Q}$ false. $\square$

Notice that every counterfactual cause is also an actual cause, with empty contingent set. Actual causes that are not counterfactual causes need company to invalidate a query result. Now we ask how strong are tuples as actual causes. To answer this question, we appeal to the *responsibility* of an actual cause τ for $\mathcal{Q}$ [23], defined by:

$$Resp_D^{\mathcal{Q}}(\tau) \ := \ \frac{1}{|\Gamma| \, + \, 1},$$

where $|\Gamma|$ is the size of a smallest contingency set, Γ, for τ, and 0, otherwise.

Example 2. (ex. 1 cont.) The responsibility of $R(a,b)$ is $\frac{1}{2} = \frac{1}{1+1}$ (its several smallest contingency sets have all size 1). $R(b,b)$ and $S(a)$ are also actual causes with responsibility $\frac{1}{2}$; and $S(b)$ is actual (counterfactual) cause with responsibility $1 = \frac{1}{1+0}$. $\square$

High responsibility tuples provide more interesting explanations. Causes in this case are tuples that come with their responsibilities as "scores". All tuples can be seen as actual causes, but only those with non-zero responsibility score matter. Causality and responsibility in databases can be extended to the attribute-value level [4,6].

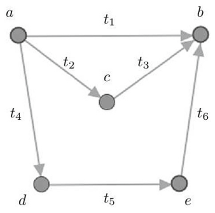
There are connections between database causality and *consistency-based diagnosis* and *abductive diagnosis*, that are two forms of *model-based diagnosis* [8,32]. There are also connections with *database repairs* [2,3]. These connections have led to complexity and algorithmic results for causality and responsibility [4,5] (see Sect. 4).

3.2 The Causal-Effect Score

Sometimes, as we will see right here below, responsibility does not provide intuitive or expected results, which led to the consideration of an alternative score, the *causal-effect score*. We show the issues and the score by means of an example.

Example 3. Consider the database E that represents the graph below, and the Boolean query $\mathcal{Q}$ that is true in E if there is a path from a to b. Here, $E \models \mathcal{Q}$. Tuples have global tuple identifiers (tids) in the left-most column, which is not essential, but convenient.

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



$$\mathcal{Q}:\ E(a,b)\ \vee$$
$$\exists x(E(a,x)\wedge E(x,b))\ \vee$$
$$\exists y\exists z(E(a,y)\wedge E(y,z)\wedge E(z,b))$$

All tuples are actual causes since every tuple appears in a path from a to b. Also, all the tuples have the same causal responsibility, $\frac{1}{3}$, which may be counterintuitive, considering that t_1 provides a direct path from a to b. $\square$

In [30], the notion *causal effect* was introduced. It is based on three main ideas, namely, the transformation, for auxiliary purposes, of the database into a probabilistic database, the expected value of a query, and interventions on the lineage of the query [10,33]. The lineage of a query represents, by means of a propositional formula, all the ways in which the query can be true in terms of the potential database tuples, and their combinations. Here, "potential" refers to tuples that can be built with the database predicates and the database (finite) domain. These tuples may belong to the database at hand or not. For a given database, D, some of those atoms become true, and others false, which leads to the instantiation of the lineage (formula) o D.

Example 4. Consider the database D below, and a BCQ.

R	A	B
	a	b
	a	c
	c	b

S	C
	b
	c

$\mathcal{Q}:\ \exists x\exists y(R(x,y)\wedge S(y))$, which is true in D.

For the database D in our example, the lineage of the query instantiated on D is given by the propositional formula:

$$\Phi_{\mathcal{Q}}(D) = (X_{R(a,b)} \wedge X_{S(b)}) \vee (X_{R(a,c)} \wedge X_{S(c)}) \vee (X_{R(c,b)} \wedge X_{S(b)}), \qquad (1)$$

where X_τ is a propositional variable that is true iff $\tau \in D$. Here, $\Phi_{\mathcal{Q}}(D)$ takes value 1 in D.

Now, for illustration, we want to quantify the contribution of tuple $S(b)$ to the query answer. For this purpose, we assign, uniformly and independently, probabilities to the tuples in D, obtaining a *probabilistic database* D^p [33]. Potential tuples outside D get probability 0.

R^p	A	B	prob
	a	b	$\frac{1}{2}$
	a	c	$\frac{1}{2}$
	c	b	$\frac{1}{2}$

S^p	C	prob
	b	$\frac{1}{2}$
	c	$\frac{1}{2}$

The X_τ's become independent, identically distributed Boolean random variables; and $\mathcal{Q}$ becomes a Boolean random variable. Accordingly, we can ask about the probability that $\mathcal{Q}$ takes the truth value 1 (or 0) when an *intervention* is performed on D.

Interventions are of the form $do(X = x)$, meaning making X take value x, with $x \in \{0, 1\}$, in the *structural model*, in this case, the lineage. That is, we ask, for $\{y, x\} \subseteq \{0, 1\}$, about the conditional probability $P(\mathcal{Q} = y \mid do(X_\tau = x))$, i.e. conditioned to making X_τ false or true.

For example, with $do(X_{S(b)} = 0)$ and $do(X_{S(b)} = 1)$, the lineage in (1) becomes, resp., and abusing the notation a bit:

$$\Phi_{\mathcal{Q}}(D|do(X_{S(b)} = 0) := (X_{R(a,c)} \wedge X_{S(c)}).$$
$$\Phi_{\mathcal{Q}}(D|do(X_{S(b)} = 1) := X_{R(a,b)} \vee (X_{R(a,c)} \wedge X_{S(c)}) \vee X_{R(c,b)}.$$

On the basis of these lineages and D^p, when $X_{S(b)}$ is made false, the probability that the instantiated lineage becomes true in D^p is:

$$P(\mathcal{Q} = 1 \mid do(X_{S(b)} = 0)) = P(X_{R(a,c)} = 1) \times P(X_{S(c)} = 1) = \frac{1}{4}.$$

When $X_{S(b)}$ is made true, the probability of the lineage being true in D^p is:

$$P(\mathcal{Q} = 1 \mid do(X_{S(b)} = 1)) = P(X_{R(a,b)} \vee (X_{R(a,c)} \wedge X_{S(c)}) \vee X_{R(c,b)} = 1) = \frac{13}{16}.$$

The *causal effect* of a tuple τ is defined by:

$$\mathcal{CE}^{D,\mathcal{Q}}(\tau) := \mathbb{E}(\mathcal{Q} \mid do(X_\tau = 1)) - \mathbb{E}(\mathcal{Q} \mid do(X_\tau = 0)).$$

In particular, using the probabilities computed so far:

$$\mathbb{E}(\mathcal{Q} \mid do(X_{S(b)} = 0)) = P(\mathcal{Q} = 1 \mid do(X_{S(b)} = 0)) = \frac{1}{4},$$
$$\mathbb{E}(\mathcal{Q} \mid do(X_{S(b)} = 1)) = P(\mathcal{Q} = 1 \mid do(X_{S(b)} = 1)) = \frac{13}{16}.$$

Then, the causal effect for the tuple $S(b)$ is: $\mathcal{CE}^{D,\mathcal{Q}}(S(b)) = \frac{13}{16} - \frac{1}{4} = \frac{9}{16} > 0$, showing that the tuple is relevant for the query result, with a relevance score provided by the causal effect, of $\frac{9}{16}$. $\square$

Let us now retake the initial example of this section.

Example 5. (ex. 3 cont.) The query has the lineage:

$$\Phi_{\mathcal{Q}}(D) = X_{t_1} \vee (X_{t_2} \wedge X_{t_3}) \vee (X_{t_4} \wedge X_{t_5} \wedge X_{t_6}).$$

It holds:

$$\begin{aligned}
&\mathcal{CE}^{D,\mathcal{Q}}(t_1) = 0.65625,\\
&\mathcal{CE}^{D,\mathcal{Q}}(t_2) = \mathcal{CE}^{D,\mathcal{Q}}(t_3) = 0.21875,\\
&\mathcal{CE}^{D,\mathcal{Q}}(t_4) = \mathcal{CE}^{D,\mathcal{Q}}(t_5) = \mathcal{CE}^{D,\mathcal{Q}}(t_6) = 0.09375.
\end{aligned}$$

The causal effects are different for different tuples, and the scores are much more intuitive than the responsibility scores. □

4 The Database Repair Connection

In this section we will first establish a useful connection between database repairs and causes as tuples in a database [2,3]. The notion of *repair* of a relational database was introduced in order to formalize the notion of *consistent query answering* (CQA), as shown in Fig. 1: If a database D is inconsistent in the sense that is does not satisfy a given set of integrity constraints, ICs, and a query $\mathcal{Q}$ is posed to D (left-hand side of Fig. 1), what are the meaningful, or consistent, answers to $\mathcal{Q}$ from D? They are sanctioned as those that hold (are returned as answers) from *all* the *repairs* of D. The repairs of D are consistent instances D' (over the same schema of D), i.e. $D' \models ICs$, and *minimally depart* from D (right-hand side of Fig. 1).

Notice that: (a) We have now a *possible-world* semantics for (consistent) query answering; and (b) we may use in principle any reasonable notion of distance between database instances, with each choice defining a particular *repair semantics*. In the rest of this section we will illustrate two classes of repairs, which have been used and investigated the most in the literature. Actually, repairs in general have got a life of their own, beyond consistent query answering.

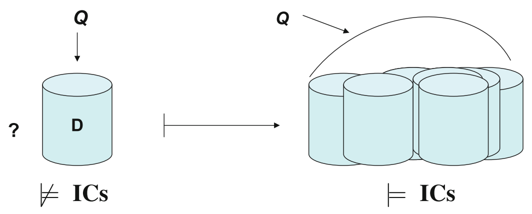


Fig. 1. Database repairs and consistent query answers

Example 6. Let us consider the following set of *denial constraints* (DCs) and a database D, whose relations (tables) are shown right here below. D is inconsistent, because it violates the DCs: it satisfies the joins that are prohibited by the DCs.

$$\neg\exists x\exists y(P(x)\wedge Q(x,y))$$
$$\neg\exists x\exists y(P(x)\wedge R(x,y))$$

P	A
	a
	e

Q	A	B
	a	b

R	A	C
	a	c

We want to repair the original instance by *deleting tuples* from relations. Notice that, for DCs, insertions of new tuple will not restore consistency. We could change (update) attribute values though, a possibility that has been investigated in [6].

Here we have two *subset repairs*, a.k.a. *S-repairs*. They are subset-maximal consistent subinstances of D: $D_1=\{P(e),Q(a,b),R(a,c)\}$ and $D_2=\{P(e),\ P(a)\}$. They are consistent, subinstances of D, and any proper superset of them (still contained in D) is inconsistent. (In general, we will represent database relations as set of tuples.)

We also have *cardinality repairs*, a.k.a. *C-repairs*. They are consistent subinstances of D that minimize the *number* of tuples by which they differ from D. That is, they are maximum-cardinality consistent subinstances. In this case, only D_1 is a C-repair. Every C-repair is an S-repair, but not necessarily the other way around. $\square$

Let us now consider a BCQ

$$\mathcal{Q}\colon \exists\bar{x}(P_1(\bar{x}_1)\wedge\cdots\wedge P_m(\bar{x}_m)), \quad (2)$$

which we assume is true in a database D. It turns out that we can obtain the causes for $\mathcal{Q}$ to be true D, and their contingency sets from database repairs. In order to do this, notice that $\neg\mathcal{Q}$ becomes a DC

$$\kappa(\mathcal{Q})\colon\ \neg\exists\bar{x}(P_1(\bar{x}_1)\wedge\cdots\wedge P_m(\bar{x}_m)); \quad (3)$$

and that $\mathcal{Q}$ holds in D iff D is inconsistent w.r.t. $\kappa(\mathcal{Q})$.

It holds that S-repairs are associated to causes with minimal contingency sets, while C-repairs are associated to causes for $\mathcal{Q}$ with minimum contingency sets, and maximum responsibilities [4]. In fact, for a database tuple $\tau\in D$:

(a) τ is actual cause for $\mathcal{Q}$ with subset-minimal contingency set Γ iff $D\smallsetminus(\Gamma\cup\{\tau\})$ is an S-repair (w.r.t. $\kappa(\mathcal{Q})$), in which case, its responsibility is $\frac{1}{1+|\Gamma|}$.
(b) τ is actual cause with minimum-cardinality contingency set Γ iff $D\smallsetminus(\Gamma\cup\{\tau\})$ is C-repair, in which case, τ is a maximum-responsibility actual cause.

Conversely, repairs can be obtained from causes and their contingency sets [4]. These results can be extended to unions of BCQs (UBCQs), or equivalently, to sets of denial constraints.

One can exploit the connection between causes and repairs to understand the computational complexity of the former by leveraging existing results for the latter. Beyond the fact that computing or deciding actual causes can be done in polynomial time in data for CQs and UCQs [4,23], one can show that most computational problems related to responsibility are hard, because they are also hard for repairs, in particular, for C-repairs (all this in data complexity) [20]. In particular, one can prove [4]: (a) The *responsibility problem*, about deciding if a tuple has responsibility above a certain threshold, is NP-complete for UCQs. (b) Computing $Resp_D^{\mathcal{Q}}(\tau)$ is $FP^{NP(log(n))}$-complete for BCQs.

This the *functional*, non-decision, version of the responsibility problem. The complexity class involved is that of computational problems that use polynomial time with a logarithmic number of calls to an oracle in *NP*. (c) Deciding if a tuple τ is a most responsible cause is $P^{NP(log(n))}$-complete for BCQs. The complexity class is as the previous one, but for decision problems [1].

5 Causes Under Integrity Constraints

In this section we consider tuples as causes for query answering in the more general setting where databases are subject to integrity constraints (ICs). In this scenario, and in comparison with Sect. 3.1, not every intervention on the database is admissible, because the ICs have to be satisfied. As a consequence, the definitions of cause and responsibility have to be modified accordingly. We illustrate the issues by means of an example. More details can be found in [5,6].

We start assuming that a database D satisfies a set of ICs, Σ, i.e. $D \models \Sigma$. If we concentrate on BCQs, or more, generally on monotone queries, and consider causes at the tuple level, only instances obtained from D by interventions that are tuple deletions have to be considered; and they should satisfy the ICs. More precisely, for τ to be actual cause for $\mathcal{Q}$, with a contingency set Γ, it must hold [5]:

(a) $D \smallsetminus \Gamma \models \Sigma$, and $D \smallsetminus \Gamma \models \mathcal{Q}$.
(b) $D \smallsetminus (\Gamma \cup \{\tau\}) \models \Sigma$, and $D \smallsetminus (\Gamma \cup \{\tau\}) \not\models \mathcal{Q}$.

The *responsibility* of τ, denoted $Resp^{\mathcal{Q}}_{D,\Sigma}(\tau)$, is defined as in Sect. 3.1, through minimum-size contingency sets.

Example 7. Consider the database instance D below, initially without additional ICs.

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

Course	*CName*	*TStaff*	*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

Let us first consider the following open query:[1]

$$\mathcal{Q}(x) \colon \exists y \exists z (Dep(y, x) \wedge Course(z, x, y)). \qquad (4)$$

In this case, we get answers other that *yes* or *no*. Actually, $\langle \text{John} \rangle \in \mathcal{Q}(D)$, the set of answers to $\mathcal{Q}$, and we look for causes for this particular answer. It holds: (a) t_1 is a counterfactual cause; (b) t_4 is actual cause with single minimal contingency set $\Gamma_1 = \{t_8\}$; (c) t_8 is actual cause with single minimal contingency set $\Gamma_2 = \{t_4\}$.

Let us now impose on D the *inclusion dependency* (IND):

[1] The fact that it is open is not particularly relevant, because we can instantiate the query with the answer, obtaining a Boolean query.

$$\psi: \quad \forall x \forall y\, (Dep(x,y) \rightarrow \exists u\; Course(u,y,x)), \tag{5}$$

which is satisfied by D. Now, t_4 t_8 are not actual causes anymore; and t_1 is still a counterfactual cause.

Let us now consider the query: $\mathcal{Q}_1(x)\colon\ \exists y\; Dep(y,x)$. Now, $\langle \mathsf{John}\rangle \in \mathcal{Q}_1(D)$, and under the IND (5), we obtain the same causes as for Q, which is not surprising considering that $\mathcal{Q} \equiv_{\psi} \mathcal{Q}_1$, i.e. the two queries are logically equivalent under (5).

And now, consider the query: $\mathcal{Q}_2(x)\colon\ \exists y \exists z\, Course(z,x,y)$, for which $\langle \mathsf{John}\rangle \in \mathcal{Q}_2(D)$. For this query we consider the two scenarios, with and without imposing the IND. Without imposing (5), t_4 and t_8 are the only actual causes, with contingency sets $\Gamma_1 = \{t_8\}$ and $\Gamma_2 = \{t_4\}$, resp.

However, imposing (5), t_4 and t_8 are still actual causes, but we lose their smallest contingency sets Γ_1 and Γ_2 we had before: $D \smallsetminus (\Gamma_1 \cup \{t_4\}) \not\models \psi$, $D \smallsetminus (\Gamma_2 \cup \{t_8\}) \not\models \psi$. Actually, the smallest contingency set for t_4 is $\Gamma_3 = \{t_8, t_1\}$; and for t_8, $\Gamma_4 = \{t_4, t_1\}$. We can see that under the IND, the responsibilities of t_4 and t_8 decrease:

$$Resp_{D}^{\mathcal{Q}_2(\mathsf{John})}(t_4) = \frac{1}{2}, \text{ and } Resp_{D,\psi}^{\mathcal{Q}_2(\mathsf{John})}(t_4) = \frac{1}{3}.$$

Tuple t_1 is not an actual cause, but it affects the responsibility of actual causes., □

Some results about causality under ICs can be obtained [5]: (a) Causes are preserved under logical equivalence of queries under ICs, (b) Without ICs, deciding causality for BCQs is tractable, but their presence may make complexity grow. More precisely, there are a BCQ and an inclusion dependency for which deciding if a tuple is an actual cause is NP-complete in data.

6 The Shapley Value in Databases

The Shapley value was proposed in game theory by Lloyd Shapley in 1953 [31], to quantify the contribution of a player to a coalition game where players share a wealth function.[2] It has been applied in many disciplines. In particular, it has been investigated in computer science under *algorithmic game theory* [25], and it has been applied to many and different computational problems. The computation of the Shapley value is, in general, intractable. In many scenarios where it is applied its computation turns out to be $\#P$-hard [12,13]. Here, the class $\#P$ contains the problems of *counting* the solutions for problems in NP. A typical problem in the class, actually, hard for the class, is $\#SAT$, about counting the number of satisfying assignments for a propositional formula. Clearly, this problem cannot be easier than SAT, because a solution for $\#SAT$ immediately gives a solution for SAT [1].

[2] The original paper and related ones on the Shapley value can be found in the book edited by Alvin Roth [28]. Shapley and Roth shared the Nobel Prize in Economic Sciences 2012.

Consider a set of players D, and a game function, $\mathcal{G} : \mathcal{P}(D) \longrightarrow \mathbb{R}$, where $\mathcal{P}(D)$ the power set of D. The Shapley value of player p in D es defined by:

$$Shapley(D, \mathcal{G}, p) := \sum_{S \subseteq D \setminus \{p\}} \frac{|S|!(|D| - |S| - 1)!}{|D|!}(\mathcal{G}(S \cup \{p\}) - \mathcal{G}(S)). \quad (6)$$

Notice that here, $|S|!(|D| - |S| - 1)!$ is the number of permutations of D with all players in S coming first, then p, and then all the others. That is, this quantity is the expected contribution of player p under all possible additions of p to a partial random sequence of players followed by a random sequence of the rests of the players. Notice the counterfactual flavor, in that there is a comparison between what happens having p vs. not having it. The Shapley value is the only function that satisfies certain natural properties in relation to games. So, it is a result of a categorical set of axioms or conditions [28].

The Shapley value has been used in knowledge representation, to measure the degree of inconsistency of a propositional knowledge base [17]; in machine learning to provide explanations for the outcomes of classification models on the basis of numerical scores assigned to the participating feature values [21,22]. It has also been applied in data management to measure the contribution of a tuple to a query answer [18,19], which we briefly review in this section.

In databases, the players are tuples in a database D. We also have a Boolean query $\mathcal{Q}$, which becomes a game function, as follows: For $S \subseteq D$, i.e. a subinstance,

$$\mathcal{Q}(S) = \begin{cases} 1 \text{ if } S \models \mathcal{Q}, \\ 0 \text{ if } S \not\models \mathcal{Q}. \end{cases}$$

With these elements we can define the Shapley value of a tuple $\tau \in D$:

$$Shapley(D, \mathcal{Q}, \tau) := \sum_{S \subseteq D \setminus \{\tau\}} \frac{|S|!(|D| - |S| - 1)!}{|D|!}(\mathcal{Q}(S \cup \{\tau\}) - \mathcal{Q}(S)).$$

If the query is *monotone*, i.e. its set of answers never shrinks when new tuples are added to the database, which is the case of conjunctive queries (CQs), among others, the difference $\mathcal{Q}(S \cup \{\tau\}) - \mathcal{Q}(S)$ is always 1 or 0, and the average in the definition of the Shapley value returns a value between 0 and 1. This value quantifies the contribution of tuple τ to the query result. It was introduced and investigated in [18,19], for BCQs and some aggregate queries defined over CQs. We report on some of the findings in the rest of this section. The analysis has been extended to queries with negated atoms in CQs [27].

A main result obtained in [18,19] is about the complexity of computing this Shapley score. The following *Dichotomy Theorem* holds: For $\mathcal{Q}$ a BCQ without self-joins, if $\mathcal{Q}$ is *hierarchical*, then $Shapley(D, \mathcal{Q}, \tau)$ can be computed in polynomial-time (in the size of D); otherwise, the problem is $\#P$-complete.

Here, $\mathcal{Q}$ is hierarchical if for every two existential variables x and y, it holds: (a) $Atoms(x) \subseteq Atoms(y)$, or $Atoms(y) \subseteq Atoms(x)$, or $Atoms(x) \cap Atoms(y) = \emptyset$. For example, $\mathcal{Q} : \exists x \exists y \exists z (R(x, y) \land S(x, z))$, for which $Atoms(x) = \{R(x, y),$

$S(x,z)\}$, $Atoms(y) = \{R(x,y)\}$, $Atoms(z) = \{S(x,z)\}$, is hierarchical. However, $\mathcal{Q}^{nh}: \exists x \exists y(R(x) \land S(x,y) \land T(y))$, for which $Atoms(x) = \{R(x),\ S(x,y)\}$, $Atoms(y) = \{S(x,y), T(y)\}$, is not hierarchical.

These are the same criteria for (in)tractability that apply to evaluation of BCQs over probabilistic databases [33]. However, the same proofs do not apply, at least not straightforwardly. The intractability result uses query $\mathcal{Q}^{nh}$ above, and a reduction from counting independent sets in a bipartite graph.

The dichotomy results can be extended to summation over CQs, with the same conditions and cases. This is because the Shapley value, as an expectation, is linear. Hardness extends to aggregates max, min, and avg over non-hierarchical queries.

For the hard cases, there is, as established in [18,19], an *approximation result*: For every fixed BCQ $\mathcal{Q}$ (or summation over a CQ), there is a *multiplicative fully-polynomial randomized approximation scheme* (FPRAS) [1], A, with, for given ϵ and δ:

$$P(\tau \in D \mid \frac{Shapley(D,\mathcal{Q},\tau)}{1+\epsilon} \leq A(\tau,\epsilon,\delta) \leq (1+\epsilon)Shapley(D,\mathcal{Q},\tau)\}) \geq 1-\delta.$$

A related and popular score, in coalition games and other areas, is the *Banzhaf Power Index*, which is similar to the Shapley value, but the order of players is ignored, by considering subsets of players rather than permutations thereof. It is defined by:

$$Banzhaf(D,\mathcal{Q},\tau) := \frac{1}{2^{|D|-1}} \cdot \sum_{S \subseteq (D \setminus \{\tau\})} (\mathcal{Q}(S \cup \{\tau\}) - \mathcal{Q}(S)).$$

The Banzhaf-index is also difficult to compute; provably #*P*-hard in general. The results in [18,19] carry over to this index when applied to query answering. In [18] it was proved that the causal-effect score of Sect. 3.2 coincides with the Banzhaf-index, which gives to the former an additional justification.

In [9], additional applications of the Shapley value in databases are described.

7 Final Remarks

Explainable data management and explainable AI (XAI) are effervescent areas of research. The relevance of explanations can only grow, as observed from- and due to the legislation and regulations that are being produced and enforced in relation to explainability, transparency and fairness of data management and AI/ML systems.

There are different approaches and methodologies in relation to explanations, with causality, counterfactuals and scores being prominent approaches that have a relevant role to play. Much research is still needed on the use of *contextual, semantic and domain knowledge*. Some approaches may be more appropriate in this direction, and we argue that declarative, logic-based specifications can be successfully exploited [7].

Still fundamental research is needed in relation to the notions of *explanation* and *interpretation*. An always present question is: *What is a good explanation?*. This is not a new question, and in AI (and other areas and disciplines) it has been investigated. In particular in AI, areas such as *diagnosis* and *causality* have much to contribute.

Now, in relation to *explanations scores*, there is still a question to be answered: *What are the desired properties of an explanation score?*. The question makes a lot of sense, and may not be beyond an answer. After all, the general Shapley value emerged from a list of *desiderata* in relation to coalition games, as the only measure that satisfies certain explicit properties [28,31]. Although the Shapley value is being used in XAI, in particular in its *Shap* incarnation, there could be a different and specific set of desired properties of explanation scores that could lead to a still undiscovered explanation score.

Acknowledgments. Part of this work was funded by ANID - Millennium Science Initiative Program - Code ICN17002; and NSERC-DG 2023-04650.

References

1. Arora, S., Barak, B.: Computational Complexity. Cambridge University Press, Cambridge (2009)
2. Arenas, M., Bertossi, L., Chomicki, J. Consistent query answers in inconsistent databases. In: Proceedings of the ACM PODS, pp. 68–79 (1999)
3. Bertossi, L.: Database repairing and consistent query answering. Synthesis Lectures in Data Management. Morgan & Claypool (2011)
4. Bertossi, L., Salimi, B.: From Causes for database queries to repairs and model-based diagnosis and back. Theory Comput. Syst. **61**(1), 191–232 (2017). https://doi.org/10.1007/s00224-016-9718-9
5. Bertossi, L., Salimi, B.: Causes for query answers from databases: datalog abduction, view-updates, and integrity constraints. Int. J. Approximate Reason. **90**, 226–252 (2017)
6. Bertossi, L.: Specifying and computing causes for query answers in databases via database repairs and repair programs. Knowl. Inf. Syst. **63**(1), 199–231 (2021)
7. Bertossi, L.: Declarative approaches to counterfactual explanations for classification. Theory Pract. Logic Program. **23**(3), 559–593 (2023). arXiv Paper 2011.07423
8. Bertossi, L.: Attribution-scores and causal counterfactuals as explanations in artificial intelligence. In: Reasoning Web: Causality, Explanations and Declarative Knowledge. Springer LNCS 13759 (2023). https://doi.org/10.1007/978-3-031-31414-8_1
9. Bertossi, L., Kimelfeld, B., Livshits, E., Monet, M.: The Shapley Value in Database Management. ACM SIGMOD Rec. **52**(2), 6–17 (2023)
10. Buneman, P., Khanna, S., Tan, W.C.: Why and where: a characterization of data provenance. Proceedings of ICDT, pp. 316–330 (2001)
11. Chockler, H., Halpern, J.: Responsibility and blame: a structural-model approach. J. Artif. Intell. Res. **22**, 93–115 (2004)
12. Deng, X., Papadimitriou, C.: On the complexity of cooperative solution concepts. Math. Oper. Res. **19**(2), 257–266 (1994)
13. Faigle, U., Kern, W.: The Shapley value for cooperative games under precedence constraints. Int. J. Game Theory **21**, 249–266 (1992)
14. Halpern, J., Pearl, J.: Causes and explanations: a structural-model approach. Part I: Causes British J. Philos. Sci. **56**(4), 843–887 (2005)
15. Halpern, J.Y.: A modification of the halpern-pearl definition of causality. In: Proceedings of IJCAI, pp. 3022–3033 (2015)
16. Holland, P.W.: Statistics and causal inference. J. Am. Statist. Assoc. **81**(396), 945–960 (1986)
17. Hunter, A., Konieczny, S.: On the measure of conflicts: Shapley inconsistency values. Artif. Intell. **174**(14), 1007–1026 (2010)

18. Livshits, E., Bertossi, L., Kimelfeld, B., Sebag, M.: The Shapley value of tuples in query answering. Logical Methods Comput. Sci. **17**(3), 22.1-22.33 (2021)
19. Livshits, E., Bertossi, L., Kimelfeld, B., Sebag, M.: Query games in databases. ACM SIGMOD Rec. **50**(1), 78–85 (2021)
20. Lopatenko, A., Bertossi, L.: Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 179–193. Springer, Heidelberg (2006). https://doi.org/10.1007/11965893_13
21. Lundberg, S., et al.: From local explanations to global understanding with explainable AI for trees. Nat. Mach. Intell. **2**(1), 2522–5839 (2020)
22. Lundberg, S., Lee, S.: A unified approach to interpreting model predictions. In: Proceedings of Advances in Neural Information Processing Systems, pp. 4765–4774 (2017)
23. Meliou, A., Gatterbauer, W., Moore, K.F., Suciu, D.: The complexity of causality and responsibility for query answers and non-answers. In: Proceedings of VLDB, pp. 34–41 (2010)
24. Meliou, A., Gatterbauer, W., Halpern, J.Y., Koch, C., Moore, K.F., Suciu, D.: Causality in databases. IEEE Data Eng. Bull. **33**(3), 59–67 (2010)
25. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.V. (eds.): Algorithmic Game Theory. Cambridge University Press (2007)
26. Pearl, J.: Causality: Models, Reasoning and Inference, 2nd edn. Cambridge University Press, Cambridge (2009)
27. Reshef, A., Kimelfeld, B., Livshits, E.: The impact of negation on the complexity of the Shapley value in conjunctive queries. In: Proceedings of PODS, pp. 285–297 (2020)
28. Roth, A.E. (ed.): The Shapley Value: Essays in Honor of Lloyd S. Cambridge University Press, Shapley (1988)
29. Rubin, D.B.: Estimating causal effects of treatments in randomized and nonrandomized studies. J. Educ. Psychol. **66**, 688–701 (1974)
30. Salimi, B., Bertossi, L., Suciu, D., Van den Broeck, G.: Quantifying causal effects on query answering in databases. In: Proceedings of the 8th USENIX Workshop on the Theory and Practice of Provenance (TaPP) (2016)
31. Shapley, L.S.: A value for n-person games. Contrib. Theory Games **2**(28), 307–317 (1953)
32. Struss, P.: Model-based problem solving. In: Handbook of Knowledge Representation, Chap. 4. Elsevier, pp. 395–465 (2008)
33. Suciu, D., Olteanu, D., Re, C., Koch, C.: Probabilistic Databases. Morgan & Claypool, Synthesis Lectures on Data Management (2011)