The Complexity of Resilience and Responsibility for Self-Join-Free Conjunctive Queries

Cibele Freire
University of Massachusetts
cibelemf@cs.umass.edu

Wolfgang Gatterbauer
Carnegie Mellon University
gatt@cmu.edu

Neil Immerman
University of Massachusetts
immerman@cs.umass.edu

Alexandra Meliou
University of Massachusetts
ameli@cs.umass.edu

ABSTRACT

Several research thrusts in the area of data management have focused on understanding how changes in the data affect the output of a view or standing query. Example applications are explaining query results, propagating updates through views, and anonymizing datasets. An important aspect of this analysis is the problem of *deleting a minimum number of tuples from the input tables* to make a given Boolean query false, which we refer to as "*the resilience of a query.*" In this paper, we study the complexity of resilience for *self-join-free conjunctive queries with arbitrary functional dependencies.* The cornerstone of our work is the novel concept of triads, a simple structural property of a query that leads to the several dichotomy results we show in this paper. The concepts of triads and resilience bridge the connections between the problems of deletion propagation and causal responsibility, and allow us to substantially advance the known complexity results in these topics. Specifically, we show a dichotomy for the complexity of resilience, which identifies previously unknown tractable families for deletion propagation with source side-effects, and we extend this result to account for functional dependencies. Further, we identify a mistake in a previous dichotomy for causal responsibility, and offer a revised characterization based purely on the structural form of the query (presence or absence of triads). Finally, we extend the dichotomy for causal responsibility in two ways: (a) we account for functional dependencies in the input tables, and (b) we compute responsibility for sets of tuples specified via wildcards.

1. INTRODUCTION

As data continues to grow in volume, the results of relational queries become harder to understand, interpret, and debug through manual inspection. Data management research has recognized this fundamental need to derive *explanations for query results* and surprising observations [4, 33–35], to assist users and administrators in better understanding their data and resolving problems effectively and efficiently.

Users

	uid	*name*
u_1	1	Alice
u_2	2	Bob
u_3	3	Charlie

Requests

	type	*details*
r_1	IMAP	email (in)
r_2	SMTP	email (out)
r_3	DB	data access

Access log

	uid	*type*	*server*
a_1	1	IMAP	yoda
a_2	2	DB	yoda
a_3	1	SMTP	yoda
a_4	1	DB	yoda
a_5	3	IMAP	loki
a_6	3	DB	yoda
a_7	2	SMTP	loki
a_8	1	DB	homer

Figure 1: The Computer Science department needs to retire an old server, *yoda*. To perform the migration efficiently, IT should transfer Alice to a different email server, and migrate databases from yoda to one of the other servers.

Example 1 (System migration). *The computer science department needs to retire an old server called "yoda" (see Fig. 1). The IT department needs to understand if and how the server is currently used, to perform the migration to other servers more efficiently. More formally, the administrator wants to understand why the following query q evaluates to* `true`*:*

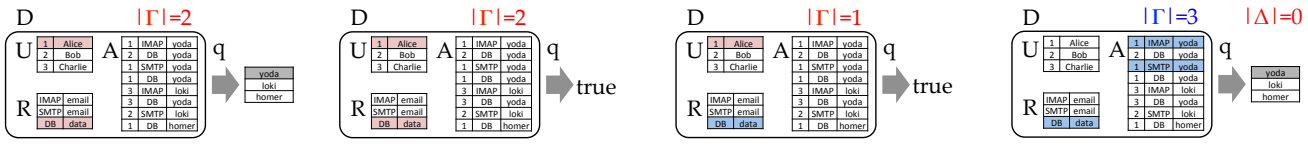
$$q :- Users(x, n), AccessLog(x, y, \text{'yoda'}), Requests(y, d)$$

Detailed analysis of the data reveals that q is `true` *due to (a) email-related requests by Alice, and (b) data access requests by several users. Thus, to perform the migration, the IT department should transfer user Alice to a different email server, and migrate the databases residing on yoda to a different server.*

An explanation that is simple and concise is more likely to be correct and to provide an efficient resolution to a potential problem. In Example 1, there are many changes to the data that could "correct" the outcome of query q, but the smallest *intervention* (removing tuples u_1 and r_3) is the one that provides better clues for the underlying context. Interventions are not explicit recommendations for a particular action (e.g., IT would not actually remove a user), but they give indications for possible actions (e.g., update a user's settings).

There are two closely related research problems that have explicitly studied the impact of such interventions to a query

Proceedings of the VLDB Endowment, Vol. 9, No. 3



(a) Source-side effects: min $|\Gamma|$ (b) Resilience: min $|\Gamma|$ (c) Responsibility: min $|\Gamma|$ (d) View-side effects: min $|\Delta|$

SJ: Queries with selections and joins	PTIME	[7]
PJ: Queries with projections and joins	NP-complete[1]	
"Key-preserving" SPJ queries	PTIME	[11]
All other SPJ queries	NP-complete[1]	
"Triad-free" SPJ queries	PTIME	this paper
All other SPJ queries	NP-complete	
"FD-induced triad-free" SPJ queries	PTIME	this paper
All other SPJ queries	NP-complete	

(e) Source-side effect problem: prior and our dichotomy results

SJ: Queries with selections and joins	PTIME	[7]
PJ: Queries with projections and joins	NP-complete	
"Key-preserving" SPJ queries	PTIME	[11]
All other SPJ queries	NP-complete	
"Head-dominated" SPJ queries	PTIME	[29]
All other SPJ queries	NP-complete	
"Functional head-dominated" SPJ queries	PTIME	[28]
All other SPJ queries	NP-complete	

(f) View-side effect problem: prior dichotomy results

Figure 2: This paper contains dichotomy results for (a) deletion propagation with source-side effects, (b) resilience, and (c) responsibility for causality. Besides others, they imply a complete dichotomy for the source side-effect problem for the class of *self-join-free conjunctive queries in the presence of functional dependencies* (e). Thus, this part of our work is similar in scope to [29] and [28] for the problem of view-side effects (f). We derive these results by analyzing a simpler concept: the resilience of Boolean queries. In addition (not shown in the figure), we provide a correction to a prior dichotomy result for causal responsibility and then extend it in two ways: responsibility for tables with functional dependencies and responsibility for tuples with wildcards, e.g., $(*, 5, 7)$.

output: causal responsibility and deletion propagation. *Causal responsibility* [31] seeks, for a given query q and a specified input tuple t, a minimum set of other input tuples Γ that, if deleted, would make t "counterfactual," i.e., the query would be true with that tuple present, or false if the tuple was also deleted. In Example 1, tuple r_3 becomes counterfactual if tuple u_1 is deleted (Fig. 2c). In contrast, *deletion propagation with source side-effects* [7,14] seeks an overall minimum set of tuples that, if deleted, would remove a particular tuple from the result (e.g., tuples u_1 and r_3 in Fig. 2a).

Unfortunately, existing work in both these problems fails to address scenarios like Example 1 effectively. The issue, at a high level, is that the known results do not lead to efficient solutions: (a) causal responsibility results in inefficiencies because it needs to be computed for every tuple in the input; (b) the known complexity results for deletion propagation with source side-effects (which is the relevant variant here) are too coarse-grained and classify even simple scenarios, such as Example 1, as NP-complete (Fig. 2e).

In this paper, we bridge the connections between these two problems and advance the existing complexity results with improved dichotomies and additional tractable classes. We achieve this by taking a step back and re-examining how particular interventions (*tuple deletions*) in the input of a query impact its output. Specifically, we study how "resilient" a Boolean query is with respect to such interventions. *Resilience* is a variant of deletion propagation for the case of Boolean queries: It identifies the smallest number of tuples to delete from the input to make the query false (e.g., Fig. 2b). A method that provides a solution to resilience can immediately also provide an answer to the *deletion propagation with source-side effects problem* by defining a new Boolean query and database, replacing all head variables in the view with constants of the output tuple. We will show that characterizing the complexity of resilience allows us to study the complexities of *both* deletion propagation with source-side effects and causal responsibility.

The work we present in this paper characterizes the complexity of resilience using the novel concept of a *triad*, a simple query structure that is sufficient to determine a full dichotomy for resilience, in the case of conjunctive queries without self-joins and with possible functional dependencies. The results we present in this paper have three important implications to existing results in *both* deletion propagation and causal responsibility. (1) They advance the known complexity results for deletion propagation with source side-effects by identifying a new large class of tractable cases. (2) They correct and refine the known complexity results on causal responsibility by replacing the complex, procedural criterion of weak linearity with the much simpler, structural criterion of triads. (3) They extend both results to account for possible functional dependencies in the input datasets.

1.1 Contributions of our work

In this paper, we study the problem of minimal interventions with respect to the *resilience* of a Boolean query. Resilience is a variant of deletion propagation with source side-effects, where we seek the minimum number of input tuples that need to be deleted in order to make a Boolean query false. We define our results in terms of "resilience" since the notion of resilience has obvious analogies to universally known minimal set cover problems. In addition, resilience helps bridge the connection between prior work in causal responsibility and deletion propagation with source side-effects.

The core concept of triads. Our first contribution, and the core of our complexity results, is the novel concept of *triads*, a simple structural property of the query hypergraph that elegantly separates the hard from the polynomial queries. A triad is a triple of points with robust connectivity

[1]In this paper we show that a big subclass of PJ queries are in fact in PTIME. The difference arises from the fact that previous work classified query families based on their operators (e.g., projections and joins). In contrast, we characterize query families based on the query structure, leading to additional tractable cases.

— a sort of super cycle (Def. 13). While previous results in causal responsibility [31] alluded to the presence of such a structure, it was not discovered, nor characterized in prior work. Triads are significant for two reasons: (1) They define a simple structural criterion that determines query complexity. In contrast, previous related results for causal responsibility were procedural, defining a series of transformations to obtain a canonical query form. (2) Our findings so far indicate that triads are a very fundamental and general concept, and we believe they can be used to characterize dichotomies for broader classes of queries. The concept of triads also provides *new tractable solutions to the otherwise hard minimum hypergraph vertex cover problem.* Our PTIME classes for resilience define families of hypergraphs for which minimum vertex cover is also always in PTIME. As such, resilience provides an intuitive definition that can draw analogies to problems even outside the database community. However, these implications are outside the scope of this paper.

Our complexity results on resilience automatically translate to new results in the problem of deletion propagation with source side-effects. Further, we were able to use the concept of triads to improve and extend known results in causal responsibility. Thus, we state our contributions with respect to these two problems.

Contributions to deletion propagation.

Our results on resilience imply a refinement for the complexity of deletion propagation with *minimum source side-effects* ($\mathtt{DP}_{\mathtt{source}}$). Our work advances previous results in two ways:

New tractable cases. The previous known results characterized all PJ queries as NP-complete [7] (Fig. 2e), because they analyzed the complexity at the level of relational operators. In this paper, however, by analyzing the query structure and using the concept of triads, we show additional tractable cases: for the class of self-join-free conjunctive queries, resilience (and by extension $\mathtt{DP}_{\mathtt{source}}$) is NP-complete if the query contains a triad, and PTIME otherwise (Sect. 3). Determining whether a query contains a triad can be done very efficiently, in polynomial time with respect to query complexity. This implies that $\mathtt{DP}_{\mathtt{source}}$ can always be solved in PTIME for the query of Example 1, even though the previously known results [7] classify it as NP-complete. These results are analogous to the results of Kimelfeld et al. [29] for the view-side effect problem. In addition, our dichotomy criterion also allows the specification of "forbidden" tables (called *exogenous* tables) that do not allow deletions. This is an extension to the traditional definition of the deletion propagation problem and affects the complexity of queries in non-obvious ways (defining a table as exogenous can make both easy queries hard, and hard queries easy).

Functional dependencies. Our work also provides a *complete dichotomy result for the class of self-join-free CQs with Functional Dependencies* (Sect. 4). These results are analogous to the results of [28] for the view-side effect problem. At a high-level, we define a new transformation driven by FDs, called *induced rewrite* (Def. 30), and show that it preserves the complexity of resilience (Lemma 31). Our second main result shows a dichotomy for resilience in the presence of FDs: if after applying all induced rewrites, the query contains a triad, then it is NP-complete (Theorem 35).

Contributions to causality.

Causal responsibility is closely related to resilience, but we show that it is a more intricate notion, with higher complexity (Lemma 7). In particular, we show query q_{rats} in Fig. 3b for which resilience is in PTIME (Cor. 23), whereas responsibility is NP-complete (Prop. 37). We discover that triads can help characterize the complexity of responsibility, and advance previous known results in four ways:

Correction of dichotomy. We found that responsibility is a more subtle concept than we previously thought. In particular, we identified an error in the existing dichotomy for responsibility [31], which classified certain hard queries into the polynomial class of queries. The problem is that the existing notion of "domination" is not sufficient to characterize the dichotomy. In Sect. 5, we provide a refinement of domination called "full domination" that solves this issue. In addition, we show that by transforming sj-free CQ into a different normal form, computing responsibility is again NP-complete in the presence of triads and PTIME otherwise (Theorem 49). Using triads makes the dichotomy characterization more elegant than the previous result, as it is based on a simple structural property of the query, rather than a series of transformations.

Functional dependencies. We prove that the existence of triads also determines a dichotomy for responsibility in the presence of FDs (Theorem 51).

Generalization to tuple groups. Further, we show that the last two dichotomies still hold for the responsibility of groups of input tuples, that can be expressed via wild cards. E.g., $R(a, *)$ denotes the set of all tuples $R(a, b)$ for any possible b (Theorems 49 and 51).

Using resilience instead of responsibility. Finally, we show how to harness the reduced complexity of resilience to derive the set of tuples with the highest responsibility for a query (Sect. 5.5). This makes resilience a better choice than responsibility in many practical settings.

Outline.

Section 2 defines all notions mentioned here more formally and discusses the connections of resilience with deletion propagation and causal responsibility. Sections 3 and 4 contain our two main technical contributions for the problem of resilience, while Sect. 5 corrects the dichotomy of responsibility and extends it to the case of tuples with wildcards and functional dependencies. Section 6 reviews the related work, and Sect. 7 discusses implications, open problems, and future directions. Due to space restrictions, some of our proofs are omitted. For these, please refer to the full version of this paper [19].

2. FORMAL SETUP AND CONNECTIONS

This section introduces our notation, defines resilience, and formalizes the connections between the problems of resilience, deletion propagation, and causal responsibility.

General notations. We use boldface (e.g., $\mathbf{x} = (x_1, \ldots, x_k)$) to denote tuples or ordered sets. A *self-join-free conjunctive query* (sj-free CQ) is a first-order formula $q(\mathbf{y}) = \exists \mathbf{x} \, (A_1 \wedge \ldots \wedge A_m)$ where the variables $\mathbf{x} = (x_1, \ldots, x_k)$ are called *existential variables*, $\mathbf{y} = (y_1, \ldots, y_c)$ are called the *head variables* (or free variables), and each atom A_i represents a relation $R_i(\mathbf{z}_i)$ where $\mathbf{z}_i \subseteq \mathbf{x} \cup \mathbf{y}$.[2]

[2]We assume w.l.o.g. that $\mathbf{z}_i$ is a tuple of only variables without constants. This is so, because for any constant in the query, we can first apply a selection on each table and then consider the modified

The term "self-join-free" means that no relation symbol occurs more than once. We write $\texttt{var}(A_j)$ for the set of variables occurring in atom A_j. The database instance is then the union of all tuples in the relations $D = \bigcup_i R_i$. As usual, we abbreviate a non-Boolean query in Datalog notation by $q(\mathbf{y}) :- A_1, \ldots, A_m.$ where q has head variables $\mathbf{y}$. For tuple $\mathbf{t}$ of the same length as $\mathbf{y}$, we write $D \models q[\mathbf{t}/\mathbf{y}]$ to mean that $\mathbf{t}$ is in the query result of the query $q(\mathbf{y})$ over database D. The set of such results is denoted by $q(\mathbf{y})^D$.

Unless otherwise stated, a *query* in this paper denotes a sj-free *Boolean* conjunctive query q (i.e., $\mathbf{y} = \emptyset$). Because we only have sj-free CQ we do not have two atoms referring to the same relation, so we may refer to atoms and relations interchangeably. We write $D \models q$ to denote that the query q evaluates to `true` over the database instance D, and $D \not\models q$ to denote that q evaluates to `false`. We call a valuation of all existential variables that is permitted by D and that makes q `true`, a *witness* $\mathbf{w}$.[3] The set of witnesses of $D \models \exists\mathbf{x}\,(A_1 \wedge \ldots \wedge A_m)$ is the set $\{\mathbf{w} \mid D \models (A_1 \wedge \ldots \wedge A_m)[\mathbf{w}/\mathbf{x}]\}$.

A database instance may contain some "forbidden" tuples that may not be deleted. Since we are interested in the data complexity of resilience, we specify *at the query level* which tables contain tuples that may or may not be deleted. Those atoms from which tuples may not be deleted are called *exogenous*[4] and we write these atoms or relations with a superscript "x". The other atoms, whose tuples may be deleted, are called *endogenous*. We may occasionally attach the superscript "n" to an atom to emphasize that it is endogenous. Moreover, we can refer to a database as a partition of its tables into its exogenous and endogenous parts, $D = D^{\mathrm{x}} \cup D^{\mathrm{n}}$.

2.1 Query resilience

In this paper, we focus on determining the *resilience* of a query with respect to changes in D^{n}. Given $D \models q$, our motivating question is: what is the minimum number of tuples to remove to make the query false? In order to study the complexity of resilience, we focus on the decision problem (rather than the optimization problem).

Definition 2 (Resilience Decision). *Given a query q and database D, we say that $(D, k) \in \texttt{RES}(q)$ if and only if $D \models q$ and there exists some $\Gamma \subseteq D^{\mathrm{n}}$ such that $D - \Gamma \not\models q$ and $|\Gamma| \leq k$.*

In other words, $(D, k) \in \texttt{RES}(q)$ means that there is a set of k or fewer endogenous tuples whose removal makes the query false. Observe that since q is computable in PTIME, $\texttt{RES}(q) \in$ NP. We will show that there is a dichotomy for sj-free CQs: for all such queries q, either $\texttt{RES}(q) \in$ PTIME or $\texttt{RES}(q)$ is NP-complete (Theorem 25).

We next compare resilience with deletion propagation with source side-effects.

2.2 Deletion propagation: source side-effects

Deletion propagation generally refers to non-Boolean queries. Given a non-Boolean query, $q(\mathbf{y}) :- A_1, \ldots, A_m$, and database D, let $\mathbf{t} \in q(\mathbf{y})^D$. The goal is to determine the minimum number of tuples that must be removed from the database, D (the source), so that $\mathbf{t}$ is no longer in the query result [7, 14]. The motivation is that q is a view from which we want to remove $\mathbf{t}$ with minimal change to the database D. We next define the decision version of this problem in our notation:

Definition 3 (Source side-effects Decision). *Given a query $q(\mathbf{y})$, database D, and an output tuple $\mathbf{t} \in q(\mathbf{y})^D$, we say that $(D, t, k) \in \texttt{DP}_{\texttt{source}}(q(\mathbf{y}))$ iff there exists some $\Gamma \subseteq D$ such that $t \notin q(\mathbf{y})^{D-\Gamma}$ and $|\Gamma| \leq k$.*

There is a close correspondence between resilience and source side-effects: Given non-boolean query, $q(\mathbf{y})$, database D, and an output tuple $\mathbf{t} \in q(\mathbf{y})^D$, we can construct the boolean query $q[\mathbf{t}/\mathbf{y}]$ over database D' in which new relations R' are constructed via the substitution $\mathbf{y} \to \mathbf{t}$. It is obvious that for any k, the two problems are equivalent:

$$(D, t, k) \in \texttt{DP}_{\texttt{source}}(q(\mathbf{y})) \quad \Leftrightarrow \quad (D', k) \in \texttt{RES}(q')\ .$$

Proposition 4 (Resilience & Source side-effects). *Every source side-effect problem can be trivially transformed into an equivalent resilience problem.*

Thus, any results on the complexity of resilience, immediately translate to the complexity of deletion propagation with source side-effects. We prefer to present our results using the notion of resilience, as there are several applications beyond view updates that relate to these problems. Examples include robustness of network connectivity (identifying sets of nodes and edges that could disconnect a network), deriving explanations for query results (finding the lineage tuples that have most impact to an output), and problems related to set cover.

View side-effects. The problem of deletion propagation with view side-effects has a different objective than resilience: it attempts to minimize the changes in the view rather than the source. We describe the existing results for this variant of the problem in Sect. 6.

2.3 Causal responsibility

A tuple t is a *counterfactual cause* for a query if by removing it the query changes from `true` to `false`. A tuple t is an *actual cause* if there exists a set Γ, called the *contingency set*, removing of which makes t a counterfactual cause. Determining actual causality is NP-complete for general formulas [15], but there are families of tractable cases [16]. Specifically, causality is PTIME for all conjunctive queries [31]. Responsibility measures the *degree* of causal contribution of a particular tuple t to the output of a query as a function of the size of a minimum contingency set: $\rho = \frac{1}{1+\min|\Gamma|}$. These definitions stem from the work of Halpern and Pearl [21], and Chockler and Halpern [10], and were adapted to queries in previous work [31]. Even though responsibility (ρ) was originally defined as inversely proportional to the size of the contingency set Γ, here we alter this definition slightly to draw parallels to the problem of resilience.

Definition 5 (Responsibility Decision). *Given query q, we say that $(D, t, k) \in \texttt{RSP}(q)$ if and only if $D \models q$ and there is $\Gamma \subseteq D^{\mathrm{n}}$ such that $D - \Gamma \models q$ and $|\Gamma| \leq k$ but $D - (\Gamma \cup \{t\}) \not\models q$.*

query with a column removed (see the transformation from resilience to source side-effects for details).

[3] Notice that our notion of witness slightly differs from the one commonly seen in provenance literature where a "witness" refers to a subset of the input database records that is sufficient to ensure that a given output tuple appears in the result of a query [9].

[4] In other words, tuples in these atoms provide context and are outside the scope of possible "interventions" in the spirit of causality [21].

In contrast to resilience, the problem of responsibility is defined for *a particular tuple* t in D, and instead of finding a Γ that will leave no witnesses for $D - \Gamma \models q$, we want to preserve only witnesses that involve t, so that there is no witness left for $D - (\Gamma \cup \{t\}) \models q$. This difference, while subtle, is significant, and can lead to different results. In Example 1, the resilience of query q has size 2 and contains tuples u_1 and r_3. However, the solution to the responsibility problem depends on the chosen tuple: the contingency set of u_1 has size 1, while the contingency set of u_2 has size 2. Furthermore, we show that the problems differ in terms of their complexity.

For completeness, we briefly recall the notions of *reduction* and *equivalence* in complexity theory:

Definition 6 (Reduction ($\leq$) and Equivalence ($\equiv$)). *For two decision problems, $S, T \subseteq \{0,1\}^*$, we say that S is* reducible *to T ($S \leq T$) if there is an* easy to compute reduction $f : \{0,1\}^* \to \{0,1\}^*$ *such that*

$$\forall w \in \{0,1\}^* \big(w \in S \Leftrightarrow f(w) \in T\big) .$$

The idea is that the complexity of S is less than or equal to the complexity of T because any membership question for S (i.e., whether $w \in S$) can be easily translated into an equivalent question for T, (i.e., whether $f(w) \in T$). "Easy to compute" can be taken as expressible in first-order logic[5]*. We say that two problems have* equivalent *complexity ($S \equiv T$) iff they are inter-reducible, i.e., $S \leq T$ and $T \leq S$.*

The problem of calculating resilience can always be reduced to the problem of calculating responsibility.

Lemma 7 ($\texttt{RES} \leq \texttt{RSP}$). *For any query q, $\texttt{RES}(q) \leq \texttt{RSP}(q)$, i.e., there is a reduction from $\texttt{RES}(q)$ to $\texttt{RSP}(q)$. Thus, if $\texttt{RES}(q)$ is hard (i.e.,* NP*-complete) then so is $\texttt{RSP}(q)$. Equivalently, if $\texttt{RSP}(q)$ is easy (i.e.,* PTIME*) then so is $\texttt{RES}(q)$.*

Later we will see a query, q_{rats}, for which $\texttt{RES}(q_{\text{rats}}) \in$ PTIME (Cor. 23) but $\texttt{RSP}(q_{\text{rats}})$ is NP-complete (Prop. 37). Thus (assuming P $\neq$ NP), $\texttt{RSP}(q)$ is sometimes strictly harder than $\texttt{RES}(q)$.

3. COMPLEXITY OF RESILIENCE

In this section we study the data complexity of resilience. We prove that the complexity of resilience of a query q can be exactly characterized via a natural property of its *dual hypergraph* $\mathcal{H}(q)$ (Def. 8). In Sect. 3.1, we begin by showing that the resilience problem for two basic queries, the triangle query ($q_\triangle$) and the tripod query (q_{T}) are both NP-complete. We then generalize these queries to a feature of hypergraphs that we call a *triad* (Def. 13), which is a set of 3 atoms that are connected in a special way in $\mathcal{H}(q)$. We then prove that if $\mathcal{H}(q)$ contains a triad, then $\texttt{RES}(q)$ is NP-complete, i.e., determining resilience is hard. Conversely, we show in Sect. 3.2 that if $\mathcal{H}(q)$ does not contain any triad, then $\texttt{RES}(q) \in$ PTIME. We prove this by showing how to transform a triad-free sj-free CQ into a linear query q' of equivalent complexity. The resilience of linear queries can be computed efficiently in polynomial time using a reduction to network flow that was proposed by previous work [31]. Our dichotomy theorem for the resilience of sj-free CQ then follows (Theorem 25).

[5]All reductions in this paper are first-order, i.e., when we write $S \leq T$ we mean $S \leq_{\text{fo}} T$. First-order reductions are natural for the relational database setting and they are more restrictive than logspace reductions, which in turn are more restrictive than polynomial-time reductions ($S \leq_{\text{fo}} T \Rightarrow S \leq_{\log} T \Rightarrow S \leq_{\text{p}} T$) [25].

3.1 Triads make resilience hard

In this section, we present our first main contribution which is the novel concept of triads (Def. 13): we prove that if the dual hypergraph of a query q contains a triad, then the resilience problem $\texttt{RES}(q)$ is NP-complete (Lemma 16). Triads were inspired by a set of queries previously studied in causal responsibility [31], but the particular structure was not discovered, nor characterized in prior work. In order to lead to the concept of triads, we have to review some basic results and queries that were initially introduced in causal responsibility [31]. While these intermediate results seem on the surface similar to those that appeared in prior work, their proofs follow different reductions that are important in understanding the proof of our main result in Lemma 16 [19].

We first define the (dual) hypergraph $\mathcal{H}(q)$ of query q. The hypergraph of a query q is usually defined with its vertices being the variables of q and the hyperedges being the atoms [1]. In this paper we use only the dual hypergraph:

Definition 8 (Dual Hypergraph $\mathcal{H}(q)$). *Let $q :- A_1, \ldots, A_m$ be a sj-free CQ. Its* dual hypergraph $\mathcal{H}(q)$ *has vertex set $V = \{A_1, \ldots, A_m\}$. Each variable $x_i \in \texttt{var}(q)$ determines the hyperedge consisting of all those atoms in which x_i occurs: $e_i = \{A_j \mid x_i \in \texttt{var}(A_j)\}$.*

For example, Fig. 3 shows the dual hypergraphs of four important queries defined in Example 9. In this paper we only consider dual hypergraphs, so we use the shorter term "hypergraph" from now on. In fact we will think of a query and its hypergraph as one and the same thing. Furthermore, when we discuss *vertices*, *edges* and *paths*, we are referring to those objects in the hypergraph of the query under consideration. Thus, a *vertex* is an *atom*, an *edge* is a *variable*, and a *path* is an alternating sequence of vertices and edges, $A_1, x_1, A_2, x_2, \ldots, A_{n-1}, x_{n-1}, A_n$, such that for all i, $x_i \in \texttt{var}(A_i) \cap \texttt{var}(A_{i+1})$, i.e., the hyperedge x_i joins vertices A_i and A_{i+1}. We explicitly list the hyperedges in the path, because more than one hyperedge may join the same pair of vertices.

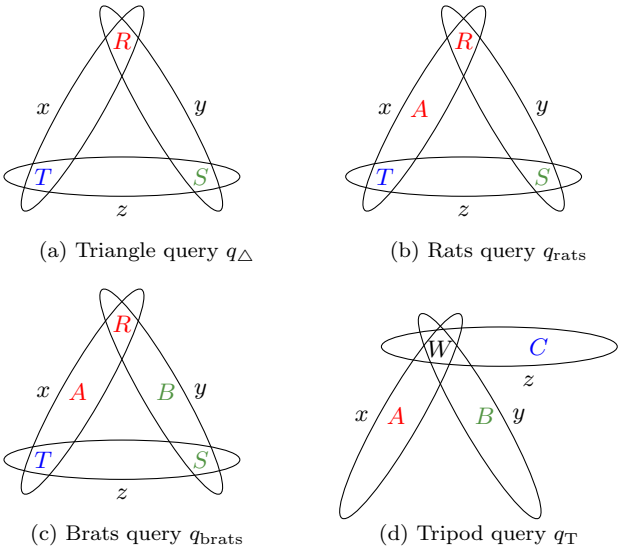
Furthermore, since disconnected components of a query have no effect on each other, each of several disconnected components can be considered independently. We will thus assume throughout that *all queries are connected.* Similarly, WLOG we assume no query contains two atoms with exactly the same set of variables.[6]

Example 9 (Important queries). *Before we precisely define what a triad is, we identify two hard queries, $q_\triangle, q_{\text{T}}$ and two related queries, q_{rats}, q_{brats} (see Fig. 3 for drawings of their hypergraphs).*

$$q_\triangle :- R(x,y), S(y,z), T(z,x) \qquad \textit{(Triangle)}$$
$$q_{rats} :- A(x), R(x,y), S(y,z), T(z,x) \qquad \textit{(Rats)}$$
$$q_{brats} :- A(x), R(x,y), B(y), S(y,z), T(z,x) \qquad \textit{(Brats)}$$
$$q_{\text{T}} :- A(x), B(y), C(z), W(x,y,z) \qquad \textit{(Tripod)}$$

We now prove that $q_\triangle$ and q_{T} are both hard, i.e., their resilience problems are NP-complete. This will lead us to the

[6]If two atoms A, B appear in q with the identical set of variables, we can replace A by $A \cap B$ and delete B.



(a) Triangle query $q_\triangle$ (b) Rats query q_{rats}

(c) Brats query q_{brats} (d) Tripod query q_{T}

Figure 3: Example 9: The hypergraphs of queries $q_\triangle$, q_{rats}, q_{brats}, q_{T}. $\{R, S, T\}$ is a triad of $q_\triangle$; {A,B,C} is a triad of q_{T}.

definition of triads, the hypergraph property that implies hardness. Later, we will see that q_{brats} is easy for both resilience and responsibility. *However, counter to our initial intuition, q_{rats} is easy for resilience but hard for responsibility.*

Proposition 10 (Triangle $q_\triangle$ is hard). $\texttt{RES}(q_\triangle)$ *and* $\texttt{RSP}(q_\triangle)$ *are* NP*-complete.*

We next show that the tripod query q_{T} is also hard. We do this by reducing the triangle to the tripod.

Proposition 11 (Tripod q_{T} is hard). $\texttt{RES}(q_{\text{T}})$ *and* $\texttt{RSP}(q_{\text{T}})$ *are* NP*-complete.*

Understanding the reduction $q_\triangle \leq q_{\text{T}}$ is useful for understanding the proof of our main result. Since we omit the proof of Prop. 11, we provide an example of the reduction:

Example 12. *The reduction $q_\triangle \leq q_{\text{T}}$ maps any pair (D, k) to a pair (D', k') such that $(D, k) \in \texttt{RES}(q_\triangle)$ iff $(D', k') \in \texttt{RES}(q_{\text{T}})$. The mapping produces the tables A, B, C, W from the tables R, S, T. For each tuple $R(a, b) \in D$, we include a tuple $A(\langle ab \rangle)$ into D'. Similarly from tables S and T, we create B and C. Finally, each witness (a, b, c) such that $D \models q_\triangle$ is mapped to the tuple $W(\langle ab \rangle, \langle bc \rangle, \langle ca \rangle) \in D'$.*

This reduction gives a 1:1 correspondence between minimum contingency sets for D and those for D'. We never need to have tuples from W in Γ as their effect would be just to remove a witness (i, j, k) of $D' \models q_{\text{T}}$, what can be done by putting one of $A(i)$, $B(j)$, or $C(k)$ into Γ. We will see that W is (dominated) by A, B, C (Def. 14).

In Fig. 4, the minimum contingency set $\{R(1, 2), S(4, 5)\}$ corresponds to the set $\{A(\langle 12 \rangle), B(\{45\})\}$, which is a minimum contingency set for D'.

While $q_\triangle$ and q_{T} appear to be very different, they share a key common structural property, which we define next.

Definition 13 (triad). *A* triad *is a set of three endogenous atoms, $\mathcal{T} = \{S_0, S_1, S_2\}$ such that for every pair i, j, there is a path from S_i to S_j that uses no variable occurring in the other atom of $\mathcal{T}$.*

R

X	Y
1	2
3	4

S

Y	Z
2	5
2	6
4	5

T

Z	X
5	1
5	3
6	1

A

X
$\langle 12 \rangle$
$\langle 34 \rangle$

B

Y
$\langle 25 \rangle$
$\langle 26 \rangle$
$\langle 45 \rangle$

C

Z
$\langle 51 \rangle$
$\langle 53 \rangle$
$\langle 61 \rangle$

W

X	Y	Z
$\langle 12 \rangle$	$\langle 25 \rangle$	$\langle 51 \rangle$
$\langle 12 \rangle$	$\langle 26 \rangle$	$\langle 61 \rangle$
$\langle 34 \rangle$	$\langle 45 \rangle$	$\langle 53 \rangle$

Figure 4: Database D and database D' defined by the reduction.

Intuitively, a triad is a triple of points with robust connectivity. Observe that atoms R, S, T form a triad in $q_\triangle$ and atoms A, B, C form a triad in q_{T} (see Fig. 3). For example, there is a path from R to S in $q_\triangle$ (across hyperedge y) that uses only variables (here y) that are not contained in the other atom (here $y \notin \texttt{var}(T)$).

A triad is composed of endogenous atoms. Some atoms such as W in q_T are given as endogenous, but are not needed in contingency sets. We will simplify the query by making all such atoms exogenous.

Definition 14 (Domination). *If a query q has endogenous atoms A, B such that $\texttt{var}(A) \subset \texttt{var}(B)$, then we say that A* dominates *B.*[7]

We already saw an example in Example 12: in q_{T}, each of the atoms A, B, C dominates W. The following proposition was proved in [31]. Unfortunately however, it was claimed to hold with respect to *responsibility* rather than *resilience*. As we will see later, this proposition fails for responsibility because the tuple we are computing the responsibility of may interfere with domination (Prop. 37).

Proposition 15 (Domination for resilience). *Let q be an sj-free CQ and q' the query resulting from labeling some dominated atoms as exogenous. Then* $\texttt{RES}(q) \equiv \texttt{RES}(q')$.

When studying resilience, we follow the convention that *all dominated atoms should become exogenous*, and we consider that the normal form of a query. For example, A dominates R and S in the query q_{rats}, and B dominates R and S in the query q_{brats}. We thus transform the queries so that the dominated atoms are exogenous. Exogenous atoms have the superscript "x".

$$q'_{\text{rats}} :- A(x), R^{\text{x}}(x, y), S(y, z), T^{\text{x}}(z, x)$$
$$q'_{\text{brats}} :- A(x), R^{\text{x}}(x, y), B(y), S^{\text{x}}(y, z), T^{\text{x}}(z, x)$$

By Prop. 15, $\texttt{RES}(q_{\text{rats}}) \equiv \texttt{RES}(q'_{\text{rats}})$ and $\texttt{RES}(q_{\text{brats}}) \equiv \texttt{RES}(q'_{\text{brats}})$. We now state our first main result.

Lemma 16 (Triads make $\texttt{RES}(q)$ hard). *Let q be an sj-free CQ where all dominated atoms are exogenous. If q has a triad, then* $\texttt{RES}(q)$ *is* NP*-complete.*

3.2 Polynomial algorithm for linear queries

We just showed that resilience for queries with triads is NP-complete. Next we will prove a strong converse: resilience for triad-free queries is in PTIME. We start by defining a class of queries for which resilience is known to be in PTIME.

[7] Recall that for $A \neq B$, we never have that $\texttt{var}(A) = \texttt{var}(B)$.

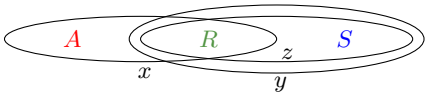


Figure 5: Def. 17: Linear query $q :- A(x), R(x, y, z), S(y, z)$.

Definition 17 (Linear Query). *A query q is* linear *if its atoms may be arranged in a linear order such that each variable occurs in a contiguous sequence of atoms.*

Example 18 (Linear Query). *Geometrically, a query is linear if all of the vertices of its hypergraph can be drawn along a straight line and all of its hyperedges can be drawn as convex regions. For example, the following query is linear, $q :- A(x), R(x, y, z), S(y, z)$ (see Fig. 5).*

The responsibility of linear queries is known to be in PTIME [31] and thus by Lemma 7, resilience of linear queries is in PTIME as well.

Fact 19 (Linear queries in PTIME [31]). *For any linear sj-free CQ q, $\texttt{RSP}(q)$ (and thus also $\texttt{RES}(q)$) are in* PTIME.

The proof of Fact 19 is that $\texttt{RES}(q)$ may be computed in a natural way using network flow. The same is true for computing the responsibility of tuple $\mathbf{t}$ for $D \models q$. In the latter case, we consider each possible extensions, $\mathbf{e}$ of $\mathbf{t}$ that is a witness of $D \models q$, and use network flow to compute the minimum size contingency set Γ for $\mathbf{t}$ such that $\mathbf{e}$ remains a witness of $D - \Gamma \models q$. The responsibility of $\mathbf{t}$ for $D \models q$ is the minimum over all such extensions $\mathbf{e}$ of the size of the minimum contingency set that preserves $\mathbf{e}$.

If all queries without a triad were linear, then this would complete the dichotomy theorem for resilience. While this is not the case, we will show that *any triad-free query can be transformed into a query of equivalent complexity that is linear.*

Recall that when studying resilience, we make atoms which are dominated, exogenous (Prop. 15). This was done, for example, to the rats and brats queries to transform them into the q'_{rats} and q'_{brats} queries. Neither of q'_{rats} or q'_{brats} is linear. However they can be transformed to linear queries without changing their complexity via the following transformation from [31]:

Definition 20 (Dissociation). *Let A^{x} be an exogenous atom in a query q, and $v \in \texttt{var}(q)$ a variable that does not occur in A^{x}. Let q' be the same as q except that we add v to the arguments A^{x}. This transformation is called* dissociation.

Example 21 (Dissociation). *The above queries q'_{rats} and q'_{brats} have no triads but they are not linear. However, by applying certain dissociations, we obtain the following linear queries:*

$$q''_{rats} :- A(x), R^{\text{x}}(x, y, z), S(y, z), T^{\text{x}}(x, y, z)$$
$$q''_{brats} :- A(x), R^{\text{x}}(x, y, z), B(y), S^{\text{x}}(x, y, z), T^{\text{x}}(x, y, z)$$

Note also that q''_{rats} and q''_{brats} have duplicate atoms which we finally delete, without affecting their complexity:

$$q'''_{rats} :- A(x), R^{\text{x}}(x, y, z), S(y, z)$$
$$q'''_{brats} :- A(x), R^{\text{x}}(x, y, z), B(y)$$

The key fact is that *dissociation can increase, but never decrease the complexity of resilience or responsibility.*[8]

Lemma 22 (Dissociation increases complexity [31]). *If q' can be obtained from q through dissociation, then $\texttt{RES}(q) \leq \texttt{RES}(q')$.*

It follows from Lemma 22 that if q can be dissociated into a linear query, then $\texttt{RES}(q) \in$ PTIME. In particular, the above dissociations of q'_{rats} and q'_{brats} prove that $\texttt{RES}(q'_{\text{rats}})$ and $\texttt{RES}(q'_{\text{brats}})$ are in PTIME. Thus, since the transformations from q_{rats} to q'_{rats} and q_{brats} to q'_{brats} preserve the complexity of resilience, we conclude that $\texttt{RES}(q_{\text{rats}})$ and $\texttt{RES}(q_{\text{brats}})$ are easy. Later we will see that, for responsibility, $\texttt{RSP}(q_{\text{brats}}) \in$ PTIME but $\texttt{RSP}(q_{\text{rats}})$ is NP-complete (Prop. 37).

Corollary 23. *$\texttt{RES}(q_{rats})$ and $\texttt{RES}(q_{brats})$ are in* PTIME.

Later we will see that it is also true that dissociation does not decrease the complexity of responsibility, but the proof is more subtle (Sect. 5.3).

Now we are ready to show that $\texttt{RES}(q)$ is easy if q is triad-free. We will show that for every triad-free query, we can linearize the endogenous atoms and use some dissociations to make the exogenous atoms fit into the same order.

Lemma 24 (Queries without triads are easy). *Let q be an sj-free CQ that has no triad. Then $\texttt{RES}(q)$ is in* PTIME.

3.3 Dichotomy of resilience

Combining Lemma 16 and Lemma 24 leads to our first dichotomy result on the complexity of resilience:

Theorem 25 (Dichotomy of resilience). *Let q be an sj-free CQ and let q' be the result of making all dominated atoms exogenous. If q' has a triad, then $\texttt{RES}(q)$ is* NP*-complete, otherwise it is in* PTIME.

Note that it is easy to tell whether q has a triad. Checking whether a given triple of atoms is a triad consists of three reachability problems – is there a path from S_i to S_j not using any of the edges in $\texttt{var}(S_k)$ – and is thus doable in linear time. An exhaustive search of all endogenous triples thus provides a PTIME algorithm:

Corollary 26. *We can check in polynomial time in the size of the query q whether $\texttt{RES}(q)$ is* NP*-complete or* PTIME.

4. FUNCTIONAL DEPENDENCIES

Functional dependencies (FDs), such as key constraints, restrict the set of allowable data instances. In this section, we characterize how these restrictions affect the complexity of resilience. We first show that FDs cannot increase the complexity of the resilience of a query (Prop. 27). Next we introduce a transformation of queries suggested by a given set of FDs call *induced rewrites* (Def. 30). We show that induced rewrites preserve the complexity of resilience (Lemma 31).

[8] For example, the query $\ell :- A(x), W_1^{\text{x}}(x, y), B(y), W_2^{\text{x}}(y, z), C(z)$ is linear, but by dissociating W_1 and W_2, we can transform it into q_{T}.

We call a query *closed* if all possible induced rewrites have been applied (Def.30). We conjectured that induced rewrites capture the full power of FDs with respect to the complexity of resilience, in other words, the complexity of the resilience of a closed query is unchanged if we remove its FDs (Conjecture 33).

We prove that the complexity of resilience for closed queries that have triads is NP-complete (Lemma 34). On the other hand, even without its FDs, we know that a closed query that has no triads has an easy resilience problem (Lemma 24). We thus conclude that in the presence of FDs, the dichotomy – still determined by the presence or absence of triads, but now in the closure of the query – remains in force (Lemma 24). It follows as a corollary that Conjecture 33 holds.

4.1 FDs can only simplify resilience

We write $\texttt{RES}(q;\Phi)$ to refer to the resilience problem for query q, restricted to databases satisfying the set of FDs Φ. Note that since we are always considering conjunctive queries, any particular FD either holds or does not hold on the whole query, so it is not necessary to mention which atom the FD is applied to.

First we observe that FDs cannot make the resilience problem harder:

PROPOSITION 27 (FDS DO NOT INCREASE COMPLEXITY). *Let q be an sj-free CQ and Φ a set of functional dependencies. Then $\texttt{RES}(q;\Phi) \leq \texttt{RES}(q)$.*

COROLLARY 28 (TRIAD-FREE QUERIES ARE STILL EASY). *If q is an sj-free CQ that has no triad, and therefore $\texttt{RES}(q)$ is in PTIME, then $\texttt{RES}(q;\Phi)$ is also in PTIME.*

We next show that for some queries, FDs do in fact reduce the complexity of resilience. Recall that the tripod query, q_{T} is hard (Prop. 11). However, q_{T} becomes polynomial when we add the FD $\varphi = x \to y$.

PROPOSITION 29. $\texttt{RES}(q_{\text{T}};\{x \to y\}) \in \text{PTIME}$.

We will prove Prop. 29 along the way, as we learn about the effect of FDs. Recall that the tripod query q_{T} has the triad $\{A, B, C\}$. Notice that the FD $x \to y$ "disarms" this triad because A and B are no longer independent. More explicitly, once we know x, we also know y. Thus $\texttt{RES}(q_{\text{T}};\{x \to y\}) \equiv \texttt{RES}(r)$ where $r :- A'(x,y), B(y), C(z), W^{\text{x}}(x,y,z)$ (Lemma 31). Furthermore, since B dominates A' in r, A' becomes exogenous: $r' :- A'^{\text{x}}(x,y), B(y), C(z), W^{\text{x}}(x,y,z)$. Query r' has no triad and thus is easy.

4.2 Induced rewrites preserve complexity

We call the transformation $(q_{\text{T}};\{x \to y\}) \rightsquigarrow (r;\{x \to y\})$ an *induced rewrite*[9]. Induced rewrites are key to understanding the effect of FDs on the complexity of resilience.

DEFINITION 30 (INDUCED REWRITE: $\rightsquigarrow$, CLOSED QUERY). *Given a set of functional dependencies Φ and a query q, we write $(q;\Phi) \rightsquigarrow (q';\Phi)$ to mean that q' is the result of adding the dependent variable u to some relation that contains all the determinant variables $\mathbf{v}$ for some $\mathbf{v} \to u \in \Phi$. We use $\overset{\star}{\rightsquigarrow}$ to indicate zero or more applications of $\rightsquigarrow$. If $(q;\Phi) \overset{\star}{\rightsquigarrow} (q^*;\Phi)$ and no more induced rewrites can be applied to $(q^*;\Phi)$, then we call $(q^*;\Phi)$ a* closed query *and we say that $(q^*;\Phi)$ is the* closure *of $(q;\Phi)$.*

[9]Transformations of queries called *rewrites* were defined in [31]. An induced rewrite is a rewrite that is induced by an FD.

This paper began as an attempt to determine whether the dichotomy for responsibility of sj-free CQs [31] continues to hold in the presence of FDs. In studying the effect of FDs, we defined induced rewrites and proved that induced rewrites preserve the complexity of responsibility. We conjectured that once we have reached a closed query, all the effect of the FDs on the complexity of responsibility has been exhausted and thus there is no further change if we delete all the FDs. We were able to prove this conjecture for unary FDs, i.e., those of the form $v \to u$ where v is a single variable.

However we had great difficulty proving this conjecture for all FDs. We studied the responsibility problem more carefully and found that responsibility is quite delicate. In particular, we discovered an error in Lemma 4.10 of [31], namely that Prop. 15 (in the present paper) does not hold for responsibility.

We identified resilience as a better-behaved notion than responsibility and we characterized the complexity of resilience via triads. Once we had done that, we were able to use the notion of triads to prove our conjecture about closed queries and thus prove the dichotomy theorem for resilience in the presence of arbitrary FDs.

With our improved insight from resilience, we went back and proved the dichotomy for responsibility (Theorem 49) and finally showed that it holds as well in the presence of FDs (Theorem 51).

We first show that induced rewrites preserve the complexity of resilience.

LEMMA 31 (INDUCED REWRITES PRESERVE COMPLEXITY). *Let q be a query, Φ a set of functional dependencies, and q' the result of an induced rewrite, i.e., $(q;\Phi) \rightsquigarrow (q';\Phi)$. Then $\texttt{RES}(q';\Phi) \equiv \texttt{RES}(q;\Phi)$.*

It follows immediately that applying any set of induced rewrites preserves the complexity of resilience:

COROLLARY 32. *If $(q;\Phi) \overset{\star}{\rightsquigarrow} (q';\Phi)$, then $\texttt{RES}(q';\Phi) \equiv \texttt{RES}(q;\Phi)$.*

4.3 For closed queries, FDs are superfluous

Recall that our current goal is to determine whether the dichotomy of the complexity of resilience remains true in the presence of FDs. The following is a natural conjecture which would given an affirmative answer to this question.

CONJECTURE 33 (INDUCED REWRITES SUFFICE). *Let $(q^*;\Phi)$ be a closed query, i.e., it is closed under induced rewrites. Then $\texttt{RES}(q^*;\Phi) \equiv \texttt{RES}(q^*)$.*

It is fairly easy to see that Conjecture 33 holds when all the FDs in Φ are unary, i.e., of the form $v \to u$, with u a single variable. However we were stumped about how to prove this for general FDs. This lead to our more careful analysis of the complexity of responsibility, our definition of resilience, and our characterization of the complexity of resilience via triads (Theorem 25). Now we will use that analysis to prove that the complexity of a closed query is NP-complete if it contains a triad, and in PTIME otherwise. Thus Conjecture 33 is true and the dichotomy for the complexity of resilience remains true in the presence of FDs.

LEMMA 34 (CLOSED QUERIES WITH TRIADS ARE HARD). *Let $(q^*;\Phi)$ be a closed sj-free CQ all of whose dominated atoms are exogenous. If q^* has a triad, then $\texttt{RES}(q^*;\Phi)$ is NP-complete.*

4.4 Dichotomy of resilience with FDs

Recall that FDs cannot increase the complexity of resilience and thus if q has no triad, then $\texttt{RES}(q;\Phi) \in \mathsf{PTIME}$ (Cor. 28). Thus, we have succeeded in proving the dichotomy for resilience in the presence of FDs:

THEOREM 35 (FD DICHOTOMY). *Let $(q;\Phi)$ be a sf-free CQ with functional dependencies. Let (q^*,Φ) be its closure under induced rewrites, and such that all dominated atoms of q^* are exogenous. If q^* has a triad then $\texttt{RES}(q;\Phi)$ is* NP*-complete. Otherwise, $\texttt{RES}(q;\Phi) \in \mathsf{PTIME}$.*

Note that we have also proved Conjecture 33:

COROLLARY 36 (INDUCED REWRITES SUFFICE). *Let $(q;\Phi)$ be an sj-free CQ with functional dependencies, and let q^* be the closure of q under induced rewrites. Then, $\texttt{RES}(q;\Phi) \equiv \texttt{RES}(q^*;\Phi) \equiv \texttt{RES}(q^*)$.*

5. COMPLEXITY OF RESPONSIBILITY

We now develop and prove the analogous characterizations of the complexity of responsibility. As we will see, responsibility is a bit more delicate than resilience; yet, in the end, the final theorems are similar.

We first concentrate on the difference between resilience and responsibility. Recall the following two queries:

$$q_{\text{rats}} :- A(x), R(x,y), S(y,z), T(z,x)$$
$$q'_{\text{rats}} :- A(x), R^{\text{x}}(x,y), S(y,z), T^{\text{x}}(z,x)$$

We saw earlier that $\texttt{RES}(q_{\text{rats}})$ is in PTIME (Cor. 23). The reason is that atom A dominates R and T and thus the complexity of $\texttt{RES}(q_{\text{rats}})$ is unchanged when we make R and T exogenous (Prop. 15), i.e., $\texttt{RES}(q_{\text{rats}}) \equiv \texttt{RES}(q'_{\text{rats}})$. Obviously q'_{rats} is triad-free. Thus, by Theorem 25, $\texttt{RES}(q'_{\text{rats}})$ and $\texttt{RES}(q_{\text{rats}})$ are in PTIME. We now show, however, that $\texttt{RSP}(q_{\text{rats}})$ is NP-complete.

PROPOSITION 37. $\texttt{RSP}(q_{rats})$ *is* NP*-complete.*

The proof of Prop. 37 shows that domination does not work the same way for responsibility as it does for resilience. In particular, the analogy of Prop. 15 (Domination for Resilience) does not hold for responsibility.

We next show that a modified version of domination still works for responsibility. Recall the query q_{brats} and define the query $q_{\text{br}^x\text{ats}}$ as follows:

$$q_{\text{br}^x\text{ats}} :- A(x), R^{\text{x}}(x,y), B(y), S(y,z), T(z,x)\ .$$

Notice that $\texttt{var}(A) \subset \texttt{var}(R)$ and $\texttt{var}(B) \subset \texttt{var}(R)$ and that also $\texttt{var}(R) \subseteq \texttt{var}(A) \cup \texttt{var}(B)$.

PROPOSITION 38 ($\texttt{RSP}(q_{\text{BRATS}})$). *The complexity of responsibility for q_{brats} is unchanged if we make R exogenous, i.e., $\texttt{RSP}(q_{brats}) \equiv \texttt{RSP}(q_{br^x ats})$.*

We are now ready to formalize "*full domination*", the version of domination that works for responsibility the way that domination works for resilience. For example, in the query q_{brats}, the relation R is fully dominated because every variable in $\texttt{var}(R)$ is "covered" by some other endogenous relation (Prop. 38).[10] Here are three more examples where R is fully dominated (s_1, s_2, s_3) and one where it is not (n_4):

$$s_1 :- A(x), R(x,y,w), B(y), S(y,z), T(z,x)$$
$$s_2 :- A(x), R(x,y,w), Q^{\text{x}}(w), B(y), S(y,z), T(z,x)$$
$$s_3 :- A(x), R(x,y,w), Q^{\text{x}}(w,x), B(y), S(y,z), T(z,x)$$
$$n_4 :- A(x), R(x,y,w), Q^{\text{x}}(w,z), B(y), S(y,z), T(z,x)$$

In a query q, we call a variable $w \in \texttt{var}(R)$ "*solitary*" if it cannot reach another endogenous atom without following one of the edges in $\texttt{var}(R) - \{w\}$. Then, in each of s_1, s_2, s_3, the variable w is solitary, but w is not solitary in n_4.

DEFINITION 39 (FULL DOMINATION). *Let F be an atom of query q. F is* fully dominated *iff for all non-solitary variables $y \in \texttt{var}(F)$ there is another atom A such that $y \in \texttt{var}(A) \subset \texttt{var}(F)$.*

Observe that relation R is fully dominated in q_{brats}, as well as in s_1, s_2, s_3, but not in n_4. On the other hand, R is not fully dominated in q_{rats} because y is connected to $S(y,z)$ and thus not solitary and not covered by any smaller atom.

We now show that fully dominated atoms may be made exogenous.

LEMMA 40 (FULL DOMINATION). *Let F be a fully dominated atom in an sj-free CQ q. Let q' be the modified query in which F is made exogenous. Then $\texttt{RSP}(q) \equiv \texttt{RSP}(q')$.*

5.1 Triads and hardness

Now that we have established that fully dominated atoms can be made exogenous without changing the complexity of the responsibility problem of a query, we proceed to prove a complexity dichotomy for responsibility.

When studying responsibility, we will insist from now on that every fully dominated atom is exogenous, and analogously to the resilience case, this will be considered the normal form of a query. For example, q_{rats} has no fully dominated atoms, so it is already in its normal form and it has a triad: $\{R, S, T\}$. Note that we cannot have two elements in a triad such that $\texttt{var}(S_1) \subset \texttt{var}(S_2)$ because removing $\texttt{var}(S_2)$ would isolate S_1. Thus $\{R, S, T\}$ is the unique triad of q_{rats}. On the other hand, R is fully dominated in q_{brats}, so we transform it to triad-free $q_{\text{br}^x\text{ats}}$:

$$q_{\text{br}^x\text{ats}} :- A(x), R^{\text{x}}(x,y), B(y), S(y,z), T(z,x)\ .$$

We now show that $\texttt{RSP}(q)$ is NP-complete if q has a triad.

LEMMA 41 (TRIADS MAKE $\texttt{RSP}(q)$ HARD). *Let q be an sj-free CQ where all fully dominated atoms are exogenous. If q has a triad, then $\texttt{RSP}(q)$ is* NP*-complete.*

5.2 The polynomial case

As we saw in the previous section, the presence of triads in a query makes the responsibility problem NP-complete. In the responsibility setting, we require full domination to make an atom exogenous. This means that more atoms may remain endogenous, so there can be more triads. The query q_{rats} is an example: for resilience we use domination and after applying domination, q_{rats} has no triads and

[10] Contrast this with the definition of domination (Def. 14) which only requires that some subset of the variables is covered by another relation.

thus $\texttt{RES}(q_{\text{rats}}) \in \mathsf{PTIME}$. However, if we may only apply full domination, then q_{rats} keeps the triad R, S, T and thus $\texttt{RSP}(q_{\text{rats}})$ is NP-complete.

We now want to prove the polynomial case for responsibility. Recall that in the proof of Lemma 24, we showed the following:

COROLLARY 42. *Let q be a CQ that has no triad. Then we can transform q, via a series of dissociations, to a linear query q'.*

Then, since dissociations cannot make the resilience problem of a sj-free CQ easier (Lemma 22), it followed that $\texttt{RES}(q) \in \mathsf{PTIME}$ for any such triad-free query, q.

To prove that for any triad-free, sj-free CQ q, $\texttt{RSP}(q) \in \mathsf{PTIME}$, it suffices to prove that dissociations cannot make the responsibility problem of such queries easier. As we see next, there is a surprising complication to this proof, which gives us an unexpected bonus result.

5.3 A generalization of responsibility

We want to prove that if q' is obtained from q through dissociation, then $\texttt{RSP}(q) \leq \texttt{RSP}(q')$. In the proof of the similar result for resilience we did the following. We let $R^{\text{x}}(\mathbf{z})$ be the atom that was changed to $R^{\text{x}\prime}(\mathbf{z}, v)$. We then reduced $\texttt{RES}(q)$ to $\texttt{RES}(q')$ by mapping (D, k) to (D', k) where D' is the same as D with the exception that we let $R' = \{(\mathbf{t}, d) \mid R(\mathbf{t}) \in D; d \in \text{dom}(D)\}$. This transformation does not change the witness set nor the contingency sets, because, by the way we formed R' from R, the conjunct $R'(\mathbf{z}, v)$ places the same restriction on D' that $R(\mathbf{z})$ places on D.

This proof goes through fine for responsibility except in one case, namely if the tuple $\mathbf{t}$ that we are computing the responsibility of belongs to R, the exogenous relation to which we have added the new variable v.[11]

When $\mathbf{t} \in R$, we would like to transform it to $\mathbf{t'} \in R'$ by appending a value, a_i, corresponding to the new variable, v. However, this will change responsibility in an unclear way. In particular, the responsibility of $\mathbf{t}$ does not correspond to the responsibility of $(\mathbf{t}, a)$ for any particular a. It rather corresponds to the responsibility of $(\mathbf{t}, a)$ for *all possible* a's.

To solve our problem, we need to generalize the notion of responsibility to include wildcards.

DEFINITION 43 (TUPLES WITH WILDCARDS). *Let D be a database containing a relation, $R(x_1, \ldots, x_c)$. Let $\tau = (s_1, \ldots, s_c)$ be a tuple such that each $s_i \in dom(D) \cup \{*\}$, i.e., τ may have elements in the domain in some attributes and the wildcard $*$ in others. We call τ a "*tuple with wildcards*." We say that a tuple $(a_1, \ldots, a_c) \in R$ "*matches*" τ iff for all i, $a_i = s_i$ or $s_i = *$. When D and R are understood, τ represents a set of tuples from R, $\langle \tau \rangle = \{\mathbf{a} \in R \mid \mathbf{a}$ matches $\tau\}$.*

For example, the tuple with wildcard $(a, *)$ matches all pairs from R whose first coordinate is a. We generalize responsibility to allow us to compute the responsibility of a set of tuples denoted by a tuple with wildcards:

DEFINITION 44 ($\texttt{RSP}^*$). *Let D be a database containing a relation R, q a query for D, and τ a tuple with wildcards. Then $(D, \tau, k) \in \texttt{RSP}^*(q)$ iff there exists a contingency set Γ of size k such that $(D - \Gamma) \models q$ and $(D - (\Gamma \cup \langle \tau \rangle)) \not\models q$.*

[11] The reader may wonder why we might need to compute the responsibility of an exogenous tuple. The answer is that the tuple originally might have come from an endogenous relation which we transformed to an exogenous one using full domination.

Since $\texttt{RSP}^*(q)$ is just a generalization of $\texttt{RSP}(q)$, it is immediate that $\texttt{RSP}(q) \leq \texttt{RSP}^*(q)$. Thus, $\texttt{RSP}^*(q)$ is NP-complete whenever $\texttt{RSP}(q)$ is:

COROLLARY 45 ($\texttt{RSP}^*$ HARDNESS). *Let q be an sj-free CQ all of whose fully dominated atoms are exogenous. If q has a triad then $\texttt{RSP}^*(q)$ is NP-complete.*

From our previous discussion, it now follows that dissociation does not make $\texttt{RSP}^*(q)$ easier:

LEMMA 46 (DISSOCIATION AND $\texttt{RSP}^*$). *If q' is obtained from q through dissociation, then $\texttt{RSP}^*(q) \leq \texttt{RSP}^*(q')$.*

Furthermore, linear queries are still easy for responsibility:

LEMMA 47 (LINEAR QUERIES AND $\texttt{RSP}^*$). *For any linear sj-free CQ q, $\texttt{RSP}^*(q)$ is in PTIME.*

COROLLARY 48. *If q has no triad, then $\texttt{RSP}^*(q)$ can be made linear by using dissociations, and is thus in PTIME. Therefore so is $\texttt{RSP}(q)$.*

We have thus proved our desired dichotomy for responsibility, and as a bonus, we have proved it for groups of tuples with wildcards as well:

THEOREM 49 (RESPONSIBILITY DICHOTOMY). *Let q be an sj-free CQ, and let q' be the result of making all fully dominated atoms exogenous. If q' contains a triad then $\texttt{RSP}(q)$ and $\texttt{RSP}^*(q)$ are NP-complete. Otherwise, $\texttt{RSP}(q)$ and $\texttt{RSP}^*(q)$ are PTIME.*

It follows from Cor. 48 and Cor. 45 that $\texttt{RSP}^*(q) \equiv \texttt{RSP}(q)$ for all sj-free CQ, q. Note that it is not at all clear how one would build a reduction from $\texttt{RSP}^*(q)$ to $\texttt{RSP}(q)$. However, our characterization of the complexity of $\texttt{RSP}(q)$ and $\texttt{RSP}^*(q)$ gives us this result: After all fully dominated atoms are made exogenous, if there is a triad, then $\texttt{RSP}(q)$ is NP-complete, thus so is $\texttt{RSP}^*(q)$. If there is no triad, then $\texttt{RSP}^*(q) \in \mathsf{PTIME}$, thus so is $\texttt{RSP}(q)$:

COROLLARY 50. *For all sj-free CQ q, $\texttt{RSP}(q) \equiv \texttt{RSP}^*(q)$.*

5.4 Dichotomy for responsibility with FDs

Our final theorem is that the dichotomy for responsibility continues to hold in the presence of FDs:

THEOREM 51 (FD RESPONSIBILITY DICHOTOMY). *Let $(q; \Phi)$ be an sj-free CQ with functional dependencies. Let (q^*, Φ) be its closure under induced rewrites, and such that all fully dominated atoms of q^* are exogenous. If q^* has a triad then $\texttt{RSP}(q; \Phi)$ is NP-complete. Otherwise, $\texttt{RSP}(q; \Phi) \in \mathsf{PTIME}$.*

5.5 Using resilience to compute responsibility more efficiently

We now show that in applications where we wish to find those tuples of highest responsibility, we can find them more efficiently by computing resilience instead of responsibility.

Responsibility provides a measure of the causal contribution of an input tuple to a query output. In prior work [31, 32], in order to identify likely causes, we ranked input tuples based on their responsibilities: tuples at the top of the ranking are the most likely causes, whereas tuples low in the ranking are less likely. Producing this ranking entails computing the responsibility of every tuple in the database

that is a cause for the query. This is computationally expensive, and, ultimately, unnecessary: Since most applications only care about the top-ranked causes, we only need to find the set S_ρ consisting of the tuples of highest responsibility. Computing the responsibility of other tuples is unnecessary. Using this insight, we can employ resilience to compute S_ρ more efficiently than by calculating the responsibility of every tuple in the database.

Even though resilience is strictly easier to compute than responsibility, we can compute S_ρ, the set of tuples of highest responsibility, by repeatedly computing resilience. The first observation is that any minimum contingency set for resilience is contained in S_ρ.

PROPOSITION 52. *Let S_ρ be the set of tuples of highest responsibility for database D and Boolean query q. Let Γ be a minimum contingency set for (q, D). Then all tuples in Γ have maximum responsibility for $D \models q$, i.e., $\Gamma \subseteq S_\rho$.*

PROOF. Let q, D, S_ρ, Γ be as in the statement of the proposition. Let $k = |\Gamma|$. Let $\mathbf{t}$ be any element of Γ. Note that $\Gamma - \{\mathbf{t}\}$ is a contingency set of size $k-1$ for the responsibility of $(q, D, \mathbf{t})$. Suppose for the sake of contradiction that some tuple $\mathbf{t'}$ had strictly greater responsibility than $\mathbf{t}$. Then there must be a contingency set Γ' for the responsibility of $(q, D, \mathbf{t'})$ such that $|\Gamma'| < k-1$. However, this means that $\Gamma' \cup \{\mathbf{t'}\}$ is a contingency set for the resilience of (q, D) of size less than k, contradicting the fact that Γ is a minimum contingency set. □

Therefore, all tuples in a minimum contingency set for resilience have maximum responsibility. However, there may be additional tuples with maximum responsibility that are not part of the selected resilience set Γ. These can also be derived by a simple algorithm based on the following observation.

OBSERVATION 53. *Let q, D, S_ρ, Γ, k be as in the proof of Prop. 52 and let $\mathbf{t'}$ be any tuple in D. Let Γ' be a minimum contingency set for the resilience of $(q, D - \{\mathbf{t'}\})$. Then $\mathbf{t'} \in S_\rho$ iff $|\Gamma'| = k-1$. Furthermore, if $|\Gamma'| = k-1$ then $\Gamma' \subseteq S_\rho$.*

Thus, even though responsibility is harder to compute than resilience (Lemma 7), the following algorithm computes the set of tuples of maximum responsibility by repeatedly computing resilience.

ALGORITHM 54. *(Compute max responsibility set)*
1. *Let C be the set of causes of $D \models q$*
2. *Let Γ be a minimum contingency set for (q, D)*
3. *$k := |\Gamma|$; $S := \Gamma$*
4. ***for each** $\mathbf{c} \in C - S$:*
5. *Let Γ' be a minimum contingency set for $(q, D - \{\mathbf{c}\})$*
6. ***if** $|\Gamma'| = k-1$: $S := S \cup \Gamma' \cup \{\mathbf{c}\}$*
7. ***return**(S)*

6. RELATED WORK

Sections 1 and 2 have extensively discussed prior work and the connections between resilience, deletion propagation and responsibility [7, 11]. In this section, we discuss additional related work.

Data provenance. Data provenance studies formalisms that can characterize the relation between the input and the output of a given query [6,9,13,20]. Among the kinds of provenance, "Why-provenance" is the most closely related to resilience in databases. The motivation behind Why-provenance is to find the "witnesses" for the query answer, i.e., the tuples or group of tuples in the input that can produce the answer. Resilience, searches to find a *minimum* set of input tuples that can make a query false.

View updates. The view update problem is a classical problem studied in the database literature [3, 11, 12, 14, 18, 26]. In its general form, the problem consists of finding the set of operations that should be applied to the database in order to obtain a certain modification in the view. Resilience and deletion propagation are a special cases of view updates.

Deletion propagation: view side-effects. The problem of deletion propagation with view side-effects does not directly relate to our results, because it has a different objective than resilience: it attempts to minimize the changes in the view rather than the source.

The complexity results from Buneman et al. [7] extend to the case of $\mathtt{DP}_{\mathtt{view}}(q)$, and the same is true for key preservation [11]. Later, Kimelfeld et al. [29] defined a dichotomy for the view side-effect problem by providing a characterization that uses the query structure: $\mathtt{DP}_{\mathtt{view}}(q(\mathbf{y}))$ is PTIME for queries that are *head dominated*, and NP-complete otherwise. Head domination checks for the components of the query that are connected by the existential variables, where all head variables contained in the atoms of that component appear in a single atom in the query.

Kimelfeld [28] augmented the dichotomy on $\mathtt{DP}_{\mathtt{view}}(q)$ for cases where functional dependencies (FDs) hold over the data instance D. The tractability condition for this case checks whether the query has *functional head domination*, which is an extension of the notion of head domination. We provide similar extensions in this paper for the problem of $\mathtt{DP}_{\mathtt{source}}(q(\mathbf{y}))$: our dichotomy for the case of FDs checks for triads after the query is structurally manipulated through a process we call *induced rewrites*.

Cong et al. [11] also studied a variant of deletion propagation that aims to remove a group of tuples from the view. Their results classify all conjunctive queries as NP-complete, but recently, Kimelfeld et al. [30] provided a trichotomy for the class of sj-free CQs that extends the notion of head domination, classifying queries into PTIME, k-approximable in PTIME, and NP-complete.

Explanations in Databases. Providing explanations to query answers is important because it can help identify inconsistencies and errors in the data, as well as understand the data and queries that operate on it. Causality can provide a framework for explanations of query results [31, 32], but it relies on the computation of responsibility, which is a harder problem than resilience. Other work on explanations also applies interventions, but on the queries instead of the data [34, 37]. These approaches, try to understand how the deletion, addition, or modification of predicates may affect the result of a query. There are also other approaches on deriving explanations that focus on specific database applications [2, 4, 5, 17, 27, 35]. Finally, the problem of explaining *missing* query results [8, 22–24, 36] is a problem analogous to deletion propagation, but in this case, we want to add, rather than remove tuples from the view.

7. DISCUSSION AND OUTLOOK

Summary. This paper presents dichotomy results for the resilience and responsibility of sj-free conjunctive queries.

Our results extend and generalize previous complexity results on the problem of *deletion propagation with source side-effects* and causal responsibility.

Approximation for resilience of sj-free conjunctive queries. The dichotomy results we establish in this work define sets of queries for which we can solve resilience in polynomial time, and sets of queries for which the problem is NP-complete. We cannot hope to find an efficient algorithm for the latter, unless P = NP, but we can look for an approximation for the optimal solution. In particular, a constant factor approximation might be also useful for finding a good approximation for the responsibility problem (see Section 5.5).

Conjunctive queries with self-joins. In order to complete the study of the complexity of resilience for conjunctive queries, we need to investigate the complexity of queries with self-joins. It is known that the problem is NP-complete for a query as simple as $q :- S(x), R(x, y), S(y)$ [31]. We suspect that the insights using triads to characterize the complexity of resilience in the absence of self-joins may still be useful in the presence of self-joins.

Unions of conjunctive queries. It would also be quite interesting to understand the complexity of computing the resilience for queries that are unions of conjunctive queries, i.e., disjunctions of conjunctions. This is a natural extension which we started to explore when trying to generalize our results about resilience to responsibility. In particular, there is a natural way to view the responsibility of a query as the resilience of a union of related queries.

Acknowledgements. We thank Dan Suciu for valuable input and inspiring discussions. This material is based upon work supported by the National Science Foundation under grants CCF-1115448, IIS-1421322, and IIS-1453543.

8. REFERENCES

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

[2] D. Agarwal, D. Barman, D. Gunopulos, N. E. Young, F. Korn, and D. Srivastava. Efficient and effective explanation of change in hierarchical summaries. In *KDD*, pp. 6–15, 2007.

[3] F. Bancilhon and N. Spyratos. Update semantics of relational views. *ACM TODS*, 6(4):557–575, Dec. 1981.

[4] D. Barman, F. Korn, D. Srivastava, D. Gunopulos, N. E. Young, and D. Agarwal. Parsimonious explanations of change in hierarchical data. In *ICDE*, pp. 1273–1275, 2007.

[5] G. Bender, L. Kot, and J. Gehrke. Explainable security for relational databases. *SIGMOD*, pp. 1411–1422, 2014.

[6] P. Buneman, S. Khanna, and W. C. Tan. Why and where: A characterization of data provenance. In *ICDT*, pp. 316–330, 2001.

[7] P. Buneman, S. Khanna, and W.-C. Tan. On propagation of deletions and annotations through views. In *PODS*, pp. 150–158, 2002.

[8] A. Chapman and H. V. Jagadish. Why not? In *SIGMOD*, pp. 523–534, 2009.

[9] J. Cheney, L. Chiticariu, and W. C. Tan. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4):379–474, 2009.

[10] H. Chockler and J. Y. Halpern. Responsibility and blame: A structural-model approach. *J. Artif. Intell. Res. (JAIR)*, 22:93–115, 2004.

[11] G. Cong, W. Fan, F. Geerts, J. Li, and J. Luo. On the complexity of view update analysis and its application to annotation propagation. *IEEE TKDE*, 24(3):506–519, 2012.

[12] S. S. Cosmadakis and C. H. Papadimitriou. Updates of relational views. *J. ACM*, 31(4):742–760, Sept. 1984.

[13] Y. Cui, J. Widom, and J. L. Wiener. Tracing the lineage of view data in a warehousing environment. *ACM TODS*, 25(2):179–227, 2000.

[14] U. Dayal and P. A. Bernstein. On the correct translation of update operations on relational views. *ACM TODS*, 7(3):381–416, 1982.

[15] T. Eiter and T. Lukasiewicz. Complexity results for structure-based causality. *Artif. Intell.*, 142(1):53–89, 2002.

[16] T. Eiter and T. Lukasiewicz. Causes and explanations in the structural-model approach: Tractable cases. *Artif. Intell.*, 170(6-7):542–580, 2006.

[17] D. Fabbri and K. LeFevre. Explanation-based auditing. *PVLDB*, 5(1):1–12, 2011.

[18] R. Fagin, J. D. Ullman, and M. Y. Vardi. On the semantics of updates in databases. In *PODS*, pp. 352–365, 1983.

[19] C. Freire, W. Gatterbauer, N. Immerman, and A. Meliou. A characterization of the complexity of resilience and responsibility for self-join-free conjunctive queries. *arXiv*, 1507.00674:1 – 36, 2015.

[20] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, pp. 31–40, 2007.

[21] J. Y. Halpern and J. Pearl. Causes and explanations: A structural-model approach. Part I: Causes. *Brit. J. Phil. Sci.*, 56:843–887, 2005.

[22] M. Herschel and M. A. Hernández. Explaining missing answers to SPJUA queries. *PVLDB*, 3(1):185–196, 2010.

[23] M. Herschel, M. A. Hernández, and W. C. Tan. Artemis: A system for analyzing missing answers. *PVLDB*, 2(2):1550–1553, 2009.

[24] J. Huang, T. Chen, A. Doan, and J. F. Naughton. On the provenance of non-answers to queries over extracted data. *PVLDB*, 1(1):736–747, 2008.

[25] N. Immerman. *Descriptive Complexity*. Springer, 1999.

[26] A. M. Keller. Algorithms for translating view updates to database updates for views involving selections, projections, and joins. In *PODS*, pp. 154–163, 1985.

[27] N. Khoussainova, M. Balazinska, and D. Suciu. Perfxplain: debugging mapreduce job performance. *PVLDB*, 5(7):598–609, 2012.

[28] B. Kimelfeld. A dichotomy in the complexity of deletion propagation with functional dependencies. In *PODS*, pp. 191–202, 2012.

[29] B. Kimelfeld, J. Vondrák, and R. Williams. Maximizing conjunctive views in deletion propagation. *ACM TODS*, 37(4):24:1–24:37, 2012.

[30] B. Kimelfeld, J. Vondrák, and D. P. Woodruff. Multi-tuple deletion propagation: Approximations and complexity. *PVLDB*, 6(13):1558–1569, 2013.

[31] A. Meliou, W. Gatterbauer, K. F. Moore, and D. Suciu. The complexity of causality and responsibility for query answers and non-answers. *PVLDB*, 4(1):34–45, 2010.

[32] A. Meliou, W. Gatterbauer, S. Nath, and D. Suciu. Tracing data errors with view-conditioned causality. In *SIGMOD*, pp. 505–516, 2011.

[33] A. Meliou, S. Roy, and D. Suciu. Causality and explanations in databases. *PVLDB*, 7(13):1715–1716, 2014.

[34] S. Roy and D. Suciu. A formal approach to finding explanations for database queries. In *SIGMOD*, pp. 1579–1590, 2014.

[35] S. Thirumuruganathan, M. Das, S. Desai, S. Amer-Yahia, G. Das, and C. Yu. Maprat: meaningful explanation, interactive exploration and geo-visualization of collaborative ratings. *PVLDB*, 5(12):1986–1989, 2012.

[36] Q. T. Tran and C.-Y. Chan. How to conquer why-not questions. In *SIGMOD*, pp. 15–26, 2010.

[37] E. Wu and S. Madden. Scorpion: Explaining away outliers in aggregate queries. *PVLDB*, 6(8):553–564, 2013.