A Characterization of the Complexity of Resilience and Responsibility for Conjunctive Queries

May 5, 2015

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. These applications usually rely on understanding how interventions in a database impact the output of a query. 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. We refer to this problem as “the resilience of a query” and show its connections to the well-studied problems of deletion propagation and causal responsibility. We thus study the complexity of resilience for self-join-free conjunctive queries, and also make several contributions to previous known results for the problems of deletion propagation with source side-effects and causal responsibility. (1) We define the notion of resilience and provide a complete dichotomy for the class of self-join-free conjunctive queries with arbitrary functional dependencies; this dichotomy also extends and generalizes previous tractability results on deletion propagation with source side-effects. (2) We formalize the connection between resilience and causal responsibility, and show that resilience has a larger class of tractable queries than responsibility. (3) We identify a mistake in a previous dichotomy for the problem of causal responsibility and offer a revised characterization based on new, simpler, and more intuitive notions. (4) Finally, we extend the dichotomy for causal responsibility in two ways: (a) we treat cases where the input tables contain functional dependencies, and (b) we compute responsibility for a set 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 explanations for surprising observations (“Why so?” or “Why no?”). Existing work has defined explanations as predicates in a query [8, 32, 35], or as modifications to the input data [22, 23, 30]. In the latter category, the metric of causal responsibility, first introduced by Chockler and Halpern [10], quantifies the contribution of an input tuple to a particular output. One can then derive explanations by ranking input tuples using their responsibilities: tuples with high degree of responsibility are better explanations for a particular query result than tuples with low responsibility [30].

A seemingly unrelated notion, the concept of deletion propagation with source side-effects [7], seeks a minimum set of tuples in the input tables that should be deleted from the database in order to delete a particular tuple from a query. Query results that have a larger set of tuples that need to be deleted are more reliable or more “robust” to changes in the input database than others. This measure of relative importance can provide another type of explanation and allows us to rank the output tuples by their relative robustness.

In this paper, we take a step back and re-examine 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 identifies the smallest number of tuples to delete from the input to make the query false. We will show that characterizing the complexity of this problem also allows us to study the complexities of both deletion propagation with source side-effects and causal responsibility.

Deletion propagation and existing results.

Databases allow users to interact with data through views, which are often conjunctive queries. Views can be used to simplify complex queries, enforce access control policies, and preserve data independence for external applications. Of particular interest is how deletions in the input data affect the view (which is a trivial problem), but also how deletions in the view could be achieved by appropriately chosen deletions in the input data (which is far less trivial). Concretely, the problem of deletion propagation [7, 14] seeks a set Γ of tuples in the input tables that should be deleted from the database in order to delete a particular tuple from the view. Intuitively, this deletion should be achieved with minimal side-effects, where side-effects are defined with either of two objectives: (a) deletion propagation with source side-effects (DP_{source}) seeks a minimum set of input tuples Γ in order
Figure 1: 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 [28] and [27] 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 tuples with wildcards, e.g., $(1,5,7)$. We derive this 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 tuples with wildcards, e.g., $(1,5,7)$.

defining a view over the database $R,S,T$ shown below. To delete tuple $v_1$ from the resulting view with minimum source side-effects, one only needs to remove tuple $t_1$ from the database. Therefore, the optimal solution to $DP_{\text{source}}$ is $\Gamma = \{t_1\}$ with $|\Gamma| = 1$ (see Fig. 1a).

However, the deletion of $t_1$ also removes $v_2$, which is a view side-effect: $\Delta = \{v_2\}$ with $|\Delta| = 1$. The optimal solution to $DP_{\text{view}}$, which minimizes the side-effects on the view (set $\Delta$) is the set of input tuples $\Gamma = \{r_1,r_2\}$: deleting these two tuples removes only $v_1$ from the view but not $v_2$, and thus has no view-side effects, i.e., $\Delta = \emptyset$ with $|\Delta| = 0$ (see Fig. 1d).

Known complexity results. Buneman et al. [7] showed that both variants are in general NP-complete for conjunctive queries containing projections and joins (PJ), whereas they are in PTIME for queries containing only selections and joins (SJ). Later, Cong at al. [11] identified a class of PJ queries, called “key-preserving,” for which both problem variants can be solved in PTIME. According to these two results, the query from Example 1 falls into the general class of NP-complete queries.

In addition, Kimelfeld et al. [28] provided a more refined dichotomy result for the problem of minimal view side-effects for self-join-free conjunctive queries (CQs). This dichotomy leads to more polynomial time cases, as it characterizes the complexity based on a property of the query structure (using the property of “head domination”), rather than high-level database operators (e.g., projections and joins). For example, the query of Example 1 is not head-dominated, which means that $DP_{\text{view}}$ is indeed NP-complete for that query. Later work has also extended the dichotomy result to self-join-free CQs with functional dependencies (FDs) [27].

Causal responsibility and existing results. The problem of causal responsibility [30] seeks, for a given query and a specified input tuple, a minimum set of other input tuples $\Gamma$ that, if deleted would make the tuple of interest “counterfactual,” i.e., the query would be true with that tuple present, or false if the tuple was also deleted. Both problems of resilience and of causal responsibility rely on the notion of minimal interventions in the input database and are thus closely related. However, we will show that resilience is easier (has lower complexity) than responsibility, and provide extensive discussion of the connections among all these related problems.

Example 2 (Resilience & Causal Responsibility). Consider again the query from Example 1 and the output tuple $v_1 = (1,9)$. Applying the substitution $[x,w] / (1,9)$, i.e., substituting the variables $x$ and $w$ with 1 and 9, respectively, we get a query $q(1,9) :\rightarrow R(1,y), S(y,z,w), T(w,9)$. The solution to $DP_{\text{source}}$ for $q$ and tuple $v_1$ is then equivalent to the solution of the resilience problem over the Boolean query $\Gamma' :\rightarrow R'(y), S(y,z,w), T'(w)\wedge R'(y) :\rightarrow R(1,y)$ and $T'(w) :\rightarrow T(w,9)$ shown below. The answer to the resilience problem for $q' = \Gamma' = \{t'_1\}$ with $|\Gamma'| = 1$: deleting tuple $t'_1$ makes the query false (also see Fig. 1b).
The causal responsibility problem requires a tuple in the lineage of the query as additional input. For example, the responsibility of tuple $s_1$ in query $q'$ corresponds to the contingency set $\Gamma = \{s_2, s_3\}$ with $|\Gamma| = 2$. Deleting these two tuples makes $s_1$ a counterfactual cause for $q'$, i.e., the query is true if $s_1$ is present or false, otherwise (also see Fig. 1c).

Known complexity results. Meliou et al. [30] showed that causality of a given tuple can be computed in polynomial time for any conjunctive query. Further, that work presented a dichotomy result for computing causal responsibility for self-join-free conjunctive queries, based on a characterization of a query property called weak linearity. However, in this work, we identify an error in the existing dichotomy which classified certain hard queries into the polynomial class of queries. In particular, we found that the existing notion of “domination” is not sufficient to characterize the dichotomy and provide a refinement of domination called “full domination” that together with a new concept of “triads” solve this issue.

Contributions of our work. In this paper, we study the problem of minimal interventions with respect to a new notion called resilience of a Boolean query, which is a minimum number of input tuples that need to be deleted in order to make the query false. 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 define our results in terms of “resilience” since the notion of resilience has obvious analogies to universally known minimal set cover problems. At the same time, our complexity results on resilience also allow us to study the problem of causal responsibility. We thus state our contributions with respect to both deletion propagation and causal responsibility.

(1) Contributions to deletion propagation. Our results on resilience imply a refinement for the complexity of minimum source side-effects by defining a novel, yet simple and intuitive property of the query structure called “triads.” For the class of self-join-free conjunctive queries, we show that resilience is $\text{NP}$-complete if the query contains this structure, and $\text{PTIME}$ 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 $\text{BP}_{\text{source}}$ can always be solved in $\text{PTIME}$ for the query of Example 1. These results are analogous to the results of [28] 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).

Our work also provides a complete dichotomy result for the class of self-join-free CQs with Functional Dependencies (Section 4). These results are analogous to the results of [27] for the view-side effect problem. At a high-level, we define rewrite steps that are induced by the functional dependencies, and check the resulting query for the presence of triads.

In particular, our dichotomy result on the resilience of a Boolean conjunctive query provides new tractable solutions to the otherwise hard minimum hypergraph vertex cover problem. Our $\text{PTIME}$ classes for resilience define families of hypergraphs for which minimum vertex cover is also always in $\text{PTIME}$. 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.

(2) Contributions to causality. We show that responsibility is a more fine-grained notion than resilience, resulting in higher complexity. In particular, we show query $q_{\text{atts}}$ in Fig. 2b for which resilience is in $\text{PTIME}$ (Corollary 25), whereas responsibility is $\text{NP}$-complete (Proposition 39). The benefit of responsibility is that it allows us to rank input tuples based on their impact to a query, thus making it applicable to applications where this ranking is important, such as providing explanations and data compression (by compressing data with small contributions to an output). In Section 7, we discuss ways to use resilience in these applications, and thus benefit from its reduced complexity compared to responsibility.

In addition, 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 [30] which classified certain hard queries into the polynomial class of queries. In particular, we found 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 helps use solve this issue. In addition, our new results provide two significant extensions to the previous dichotomy: (a) We generalize the notion of responsibility from simple tuples to tuples with wildcards. (b) We show that through a process of query rewrites, our dichotomy results continue to hold in the presence of functional dependencies over the input relations.

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 Section 5 corrects the dichotomy of responsibility and extends it to the case of tuples with wildcards and functional dependencies. Section 6 reviews additional related work, and Section 7 discusses implications, open problems, and future directions.

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., $x = (x_1, \ldots, x_k)$) to denote tuples or ordered sets. A self-join-free conjunctive query (sj-free CQ) is a first-order formula $q(y) = \exists x (A_1 \land \ldots \land A_m)$ where the variables $x = (x_1, \ldots, x_k)$ are called existential variables, $y = (y_1, \ldots, y_l)$ are called the head variables (or free variables), and each atom $A_i$ represents a relation $R_i(z_i)$ where $z_i \subseteq x \cup y$.\footnote{We assume w.l.o.g. that $z_i$ is a tuple of only variables without constants. This is so, because for any constant in the query, we can...}
The term “self-join-free” means that no relation symbol occurs more than once. We write \( \text{var}(A_j) \) 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 the query in Datalog notation by \( q(y): A_1, \ldots, A_m \). For tuple \( t \), we write \( D \models q[t/y] \) to denote that \( t \) is in the query result of the non-Boolean query \( q(y) \) over database \( D \). The set of query results over database \( D \) is denoted by \( q(y)^D \).

Unless otherwise stated, a query in this paper denotes a sj-free Boolean conjunctive query \( q \) (i.e., \( y = \emptyset \)). 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 \textit{true} over the database instance \( D \), and \( D \not\models q \) to denote that \( q \) evaluates to \textit{false}. We call a valuation of all existential variables that is permitted by \( D \) and that makes \( q \) \textit{true}, a witness \( w \).² The set of witnesses of \( D \models q \) can be defined as the set \( \{ w \mid D \models q[w/x] \} \).

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 \textit{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 \textit{exogenous}³ and we write these atoms or relations with a superscript \( \text{“}x\text{”} \). The other atoms, whose tuples may be deleted, are called \textit{endogenous}. We may occasionally attach the superscript \( \text{“}n\text{”} \) 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, \( D = D^x \cup D^n \).

### 2.1 Query resilience

In this paper, we focus on determining the resilience of a query with regard to changes in \( D^x \). Given \( D \models q \), our motivating question is: what is the minimum number of tuples to remove in order to make the query false?

**Definition 3 (Resilience).** Given a query \( q \) and database \( D \), we say that \( (D, k) \in \text{RES}(q) \) if and only if \( D \models q \) and there exists some \( \Gamma \subseteq D^x \) such that \( D - \Gamma \not\models q \) and \( |\Gamma| \leq k \).

In other words, \( (D, k) \in \text{RES}(q) \) means that there is a set of \( k \) or fewer tuples in the endogenous tables of \( D \), the removal of which makes the query false. Observe that since \( q \) is computable in \textit{PTIME}, \( \text{RES}(q) \in \text{NP} \). We will see that there is a dichotomy for all sj-free conjunctive queries: for all such queries \( q \), either \( \text{RES}(q) \in \textit{PTIME} \) or \( \text{RES}(q) \) is \textit{NP}-complete (Theorem 27). We are naturally interested in the optimization version of this decision problem: given \( q \) and \( D \), find the \textit{minimum} \( k \) so that \( (D, k) \in \text{RES}(q) \). A larger \( k \) implies that the query is more “resilient” and requires the deletion of more tuples to change the query output.

In this paper, we focus on Boolean queries, however we can also define the resilience problem for non-Boolean queries as follows:

---

²Notice 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].

³In other words, tuples in these atoms provide context and are outside the scope of possible “interventions” in the spirit of causality [30].

---

### 2.2 Deletion propagation: source side-effects

Deletion propagation in view updates generally refers to non-Boolean queries \( q(y) := A_1, \ldots, A_m \). Next we define the problem [7, 14] formally in our notation:

**Definition 5 (Source side-effects).** Given a query \( q(y) \), database \( D \), and an output tuple \( t \), we say that \( (D, t, k) \in \text{DP}_{\text{source}}(q(y)) \) if and only if \( t \in q(y)^D \) and there exists some \( \Gamma \subseteq D \) such that \( t \notin q(y)^{D-\Gamma} \) and \( |\Gamma| \leq k \).

It is easy to see that there is a homomorphism between resilience and the source-side effect variant of deletion propagation. We have illustrated this correspondence in Example 2 and next describe this transformation more formally.

Given a conjunctive query \( q(y) := A_1, \ldots, A_m \) and a tuple \( t = c \) in the output \( q(y)^D \). We first obtain a Boolean query \( q' \) by deleting the head variables in \( q(y) \). Then we modify the database by applying a filter (selection): for each relation \( R_i(x_i) \) we define a new relation \( R'_i(x_i) := R_i(\theta_i(z_i)) \) with \( x_i \), being the existential variables that occur in \( R_i \), and where the substitution \( \theta_i : y \rightarrow c \) replaces the former head variables with the corresponding constants from \( t \) and keep the existential variables as they are. For example, \( R'(y) := R(1, y) \) in Example 2 (see Fig. 1a and Fig. 1b). This will lead to a new database \( D' = \bigcup_i R'_i \) and a new Boolean query \( q' := A'_1, \ldots, A'_m \), where \( A'_i = R'_i(x_i) \) if \( A_i = R_i(x_i) \), for which the following holds:⁴

**Corollary 6 (Resilience & Source side-effects).** Given a query \( q(y) \), database \( D \), and output tuple \( t \in q(y)^D \), let \( q' \) and \( D' \) be the new Boolean query and new database instance obtained by the above transformation. Then: \( (D, t, k) \in \text{DP}_{\text{source}}(q(y)) \Leftrightarrow (D', k) \in \text{RES}(q') \).

Notice that the same transformation can be used to treat constants in a CQ when considering source side-effects. Thus, by solving the complexity of resilience, we immediately also solve the problem 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

⁴An informal way to describe this transformation of \( D \) at the query level is to first only keep tuples in the lineage of \( t \) and to then delete all columns in atoms that contain constants from \( e \).
have most impact to an output), and problems related to set cover. We proceed to discuss existing results on the complexity of deletion propagation with source side-effects, and explain how our results on the complexity of resilience extend this prior work.

Buneman et al. [7] define a dichotomy for the hardness of \( DP_{\text{source}}(q) \) based on the operations that occur in \( q \), namely, selection, projection, join, union. Specifically, they show that \( DP_{\text{source}}(q(y)) \) is NP-complete for PJ and JU queries (i.e., queries involving projections and joins, or queries involving joins and unions), while it is PTIME for SJ and SPU queries (i.e., queries involving selections and joins, or queries involving selections, projections, and unions only). Later, Cong et al. [11] showed that \( DP_{\text{source}}(q(y)) \) is in PTIME for a SPJ query if all primary keys of the involved relations appear in the head variables \( y \) (a condition called “key preservation”). Notice that the concept of key preservation does not apply to the problem of resilience, as keys are never preserved in Boolean queries.

In this paper, we identify a larger class of SPJ queries for which the problem of resilience – and thus \( DP_{\text{source}}(q(y)) \) – is in PTIME, thus extending all prior results. In Sect. 3, we provide a dichotomy result based on identifying a specific and very intuitive structure in a query, called a triad: queries that contain a triad are NP-complete, whereas those that do not are in PTIME. Our results refine the prior work in the sense that prior results characterize the dichotomy at the level of operators used in the query (e.g., joins, projections), while our result identifies all polynomial cases based on (i) the actual query and (ii) additional schema knowledge of forbidden, “exogenous” tables. In Sect. 4, we extend our results to even include (iii) functional dependencies.

2.3 Deletion propagation: 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.

**Definition 7 (View side-effects).** Given a query \( q(y) \), a database \( D \), and a tuple \( t \) in the view, we say that \((D,t,k) \in DP_{\text{view}}(q(y))\) if and only if \( t \notin q(y)^D \) and there exists some \( \Gamma \subseteq D \) such that \( q(y)^D - t \subset \Gamma \leq k, \) where \( \Delta = (q(y)^D - (q(y)^D - t)) \). In other words, \( \Delta \) is the set of tuples other than \( t \) that were eliminated from the view.

The dichotomy results from Buneman et al. [7] extend to the case of \( DP_{\text{view}}(q) \), and the same is true for key preservation [11]. Later, Kimelfeld et al. [28] refined the dichotomy for the view side-effect problem by providing a characterization that uses the query structure: \( DP_{\text{view}}(q(y)) \) 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. Our work in this paper offers a similar refinement for the dichotomy of \( DP_{\text{source}}(q(y)) \) from the characterization at the operator level to the characterization at the level of query structure, plus knowledge of exogenous (“forbidden”) tables.

**Functional dependencies.** Kimelfeld [27] augmented the dichotomy on \( DP_{\text{view}}(q) \) 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 \( DP_{\text{source}}(q(y)) \): our dichotomy for the case of FDs checks for triads after the query is structurally manipulated through a process we call induced rewrites, which is basically a chase of FDs.

**Multi-tuple deletion.** 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. [29] 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.

2.4 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 \( \Gamma \), called the contingency set, removing of which makes \( t \) a counterfactual cause. Determining actual causalitiy is NP-complete for general formulas [15], but there are families of tractable cases [16]. Specifically, causality is PTIME for all conjunctive queries [30]. 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 minimal contingency set: \( \rho = \frac{1}{1 + \min \Gamma} \). These definitions stem from the work of Halpern and Pearl [20], and Chockler and Halpern [10], and were adapted to queries in our previous work [30]. Even though responsibility \( \rho \) was originally defined as inversely proportional to the size of the contingency set \( \Gamma \), here we alter this definition slightly to draw parallels to the problem of resilience.

**Definition 8 (Responsibility).** Given query \( q \), we say that \((D,t,k) \in RSP(q)\) if and only if \( D \models q \) and there is \( \Gamma \subseteq D^0 \) such that \( D - \Gamma \models q \) and \( |\Gamma| \leq k \). In other words, while |\( \Delta \)|, is less than or equal to the size of the contingency set \( \Gamma \), here we alter this definition slightly to draw parallels to the problem of resilience.

In contrast to resilience, the problem of responsibility is defined for a particular tuple \( t \) in \( D \), and instead of finding a \( \Gamma \) 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 2, the resilience of query \( q' \) has size 1 and contains tuple \( t_1 \). However, the solution to the responsibility problem depends on the chosen tuple: the contingency set of \( s_1 \) has size 2, and this size can be made arbitrarily bigger by adding more tuples in \( S \) with attribute \( W = 7 \). 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 9 (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 \( f : \{0,1\}^* \rightarrow \{0,1\}^* \) such that

\[
\forall w \in \{0,1\}^* (w \in S \equiv f(w) \in T).
\]

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 \( f(w) \in T \)).}
to compute" can be taken as expressible in first-order logic\(^5\).

We say that two problems have equivalent complexity \((S \equiv T)\) if 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 10** \((\text{RES} \leq \text{RSP})\). For any query \(q\), \(\text{RES}(q) \leq \text{RSP}(q)\), i.e., there is a reduction from \(\text{RES}(q)\) to \(\text{RSP}(q)\). Thus, if \(\text{RES}(q)\) is hard (i.e., \(\text{NP}\)-complete) then so is \(\text{RSP}(q)\). Equivalently, if \(\text{RSP}(q)\) is easy (i.e., \(\text{PTIME}\)) then so is \(\text{RES}(q)\).

**Proof of Lemma 10.** Let \(q = \exists x_1, \ldots, x_n A_1(z_1) \land \cdots \land A_m(z_m)\). The reduction from \(\text{RES}(q)\) to \(\text{RSP}(q)\) is as follows: given \((D,k)\), we map it to \((D',t_0,k)\) where \(D'\) consists of the database \(D\) together with unique new values \(A_1, \ldots, A_m\) and the new tuples \(A_1(z_1[a/x]), \ldots, A_m(z_m[a/x])\). In other words, we enter a completely new witness \(a\) for \(q\) that has no values in common with the domain of \(D\).

Let \(t_0 = A_1(z_1[a/x]), \ldots, A_m(z_m[a/x])\). Then \((D',t_0)\) has the same size as the minimal contingency set for \(q\) in \(D\) and hence \((D,k) \in \text{RES}(q)\) implies \((D',t_0,k) \in \text{RSP}(q)\).

Later we will see a query, \(q_{\text{rats}}\), for which \(\text{RES}(q_{\text{rats}}) \in \text{PTIME}\) (Corollary 25) but \(\text{RSP}(q_{\text{rats}})\) is \(\text{NP}\)-complete (Proposition 39). Thus (assuming \(\text{P} \neq \text{NP}\)), \(\text{RSP}(q)\) is sometimes strictly harder than \(\text{RES}(q)\).

### 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 \(\mathcal{H}(q)\) (Definition 11). In Section 3.1, we begin by showing that the resilience problem for two basic queries, the triangle query \(q_{\triangle}\) and the tripod query \(q_{\triangledown}\) are both \(\text{NP}\)-complete. We then generalize these queries to a feature of hypergraphs that we call a triad (Definition 15), 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 \(\text{RES}(q)\) is \(\text{NP}\)-complete, i.e., determining resilience is hard. Conversely, we show in Section 3.2 that if \(\mathcal{H}(q)\) does not contain any triad, then \(\text{RES}(q)\) is \(\text{PTIME}\). We prove this by showing how to transform a triad-free \(\text{sJ}\)-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 as shown in previous work [30]. The desired dichotomy theorem for the resilience of \(\text{sJ}\)-free conjunctive queries thus follows (Theorem 27).

#### 3.1 Triads make resilience hard

We will define triples of atoms called triads and then prove that if the dual hypergraph of a query \(q\) contains a triad, then the resilience problem \(\text{RES}(q)\) is \(\text{NP}\)-complete.

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:

\[\mathcal{H}(q) = (V, E)\]

where \(V = \{x_1, \ldots, x_n\}\) and \(E = \{\{x_i, x_j, x_k\} | \exists a \in A \text{ s.t. } A(x_i[a/x]) \land A(x_j[a/x]) \land A(x_k[a/x])\}\). The atoms \(A\) are then the nodes of \(\mathcal{H}(q)\). The hyperedges \(\{x_i, x_j, x_k\}\) are connected if there is an atom \(A\) with \(A(x_i[a/x]) \land A(x_j[a/x]) \land A(x_k[a/x])\).

**Definition 11** (Dual Hypergraph \(\mathcal{H}(q)\)). Let \(q = \exists A_1, \ldots, A_m\) be a \(\text{sJ}\)-free CQ. Its dual hypergraph \(\mathcal{H}(q)\) has vertex set \(V = \{A_1, \ldots, A_m\}\). Each variable \(x_i \in \var(q)\) determines the hyperedge consisting of all those atoms in which \(x_i\) occurs: \(\{A_j | x_i \in \var(A_j)\}\).

For example, Fig. 2 shows the dual hypergraphs of four important queries defined in Example 12. 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 a sequence of vertices such that every successive pair is contained in an edge. 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 12** (Important queries). Before we precisely define what a triad is, we identify two hard queries, \(q_{\triangle}, q_{\triangledown}\) together with two intermediate queries, \(q_{\text{rats}}, q_{\text{brats}}\) (see Fig. 2 for drawings of their hypergraphs).

\[
q_{\triangle} : R(x,y), S(y,z), T(z,x)
\]

(Triangle)

\[
q_{\text{rats}} : A(x), R(x,y), S(y,z), T(z,x)
\]

(Rats)

\[
q_{\text{brats}} : A(x), R(x,y), B(y), S(y,z), T(z,x)
\]

(Brats)

\[
q_{\triangledown} : A(x), B(y), C(z), W(x,y,z)
\]

(Tripod)

We now prove that \(q_{\triangle}\) and \(q_{\triangledown}\) are both hard, i.e., their resilience problems are \(\text{NP}\)-complete. This will lead us to the definition of a triad: the hypergraph property that implies hardness. Later we will see that \(q_{\text{rats}}\) is easy for both resilience and responsibility. However, \(q_{\text{rats}}\) is easy for resilience but hard for responsibility.

**Proposition 13** (Triangle \(q_{\triangle}\) is hard). \(\text{RES}(q_{\triangle})\) and \(\text{RSP}(q_{\triangle})\) are \(\text{NP}\)-complete.

**Proof of Proposition 13.** We reduce \(\text{3SAT} \to \text{RES}(q_{\triangle})\). It will then follow that \(\text{RES}(q_{\triangle})\) is \(\text{NP}\)-complete, and thus so is \(\text{RSP}(q_{\triangle})\) by Lemma 10. Let \(\psi\) be a 3CNF formula with \(n\) variables \(v_1, \ldots, v_n\) and \(m\) clauses \(C_1, \ldots, C_m\). Our reduction will map any such \(\psi\) to a pair \((D_\psi, k_\psi)\) where \(D_\psi\) is a database satisfying \(q_{\triangle}\), and

\[
\psi \in \text{3SAT} \iff (D_\psi, k_\psi) \in \text{RES}(q)
\]

In our construction, if \(\psi \in \text{3SAT}\), then the size of each minimum contingency set for \(q_{\triangle}\) in \(D_\psi\) will be \(k_\psi = 6mn\), whereas if \(\psi \notin \text{3SAT}\), then the size of all contingency sets for \(q_{\triangle}\) in \(D_\psi\) will be greater than \(k_\psi\).

Note \(D_\psi \models q_{\triangle}\) iff it contains three pairs \(R(a,b), S(b,c), T(c,a)\). We visualize \(R(a,b)\) as a red edge, \(S(b,c)\) as a green edge and \(T(c,a)\) as a blue edge. Thus each witness \((a,b,c)\) that \(D_\psi \models q_{\triangle}\) is an RGB triangle. (Notice that the edge direction \(a \to b\) drawn in Fig. 3 corresponds to the variable order in \(R\), and analogously for \(S\) and \(T\).) The job of a contingency set for \(q_{\triangle}\) is to remove all RGB triangles.

\[\text{If two atoms } A, B \text{ appear in } q \text{ with the identical set of variables, we can replace } A \text{ by } A \cap B \text{ and delete } B.\]
We have thus constructed this single new RGB triangle because there is no other way.

In other words, a truth assignment to the variables of $D$ contains one circular gadget $G_i$ for each variable $v_i$. The circle consists of $12m$ solid edges, half of them marked $v_i$ and the other half marked $\overline{v}_i$ (see Fig. 3a and Fig. 3b). Note that there are $12m$ RGB triangles and they can be minimally broken by choosing the $6m$ $v_i$ edges or the $6m$ $\overline{v}_i$ edges. Any other way would require more edges removed. Thus, each minimum contingency set for $D_\psi$ corresponds to a truth assignment to the variables of $\psi$. And there will be a minimum contingency set of size $k_\psi = 6mn$ iff $\psi \in \text{3SAT}$.

We complete the construction of $D_\psi$ by adding one RGB triangle for each clause $C_j$. For example, suppose $C_j = v_1 \lor \overline{v}_2 \lor v_3$. The RGB triangle we add consists of a red edge marked $v_1$, a green edge marked $\overline{v}_2$ and a blue edge marked $v_3$ (see Fig. 3c). Note that if the chosen assignment satisfies $C_j$, then all $v_i$ edges are removed, or all $\overline{v}_i$ edges are removed, or all $v_i$ edges are removed. Thus the $C_j$ triangle is automatically removed.

How do we create $C_j$’s RGB triangle? Remember that we have chosen $G_i$ to contain 2 segments for each clause. We use the $j$th odd-numbered segment of $G_i$ to produce the $v_i$ or $\overline{v}_i$ used in the clause-$j$ triangle. The even numbered segments are not used: they serve as buffers to prevent spurious RGB triangles from being created (In Fig. 3b we mark these even segments with frowns: they are sad because they are never used).

More precisely, the red $v_1$-edge from $G_1$ is $(a_{j+1}^{1j+1}, b_{j+1}^{1j+1})$, the green $\overline{v}_2$-edge from $G_2$ is $(b_{j+1}^{2j+1}, c_{j+1}^{2j+1})$, and the blue $v_3$-edge from $G_3$ is $(c_{j+1}^{3j+1}, a_{j+1}^{3j+1})$ (see Fig. 3c).

Now to make this an RGB triangle in $D_\psi$, we identify the two $a$-vertices, the two $b$ vertices and the two $c$ vertices. In other words, $G_i$’s $a$-vertex $a_{j+1}^{1j+1}$ is equal to $G_3$’s $a$-vertex $a_{j+1}^{3j+1}$, i.e., they are the same element of the domain of $D_\psi$.

We have thus constructed $C_j$’s RGB triangle (see Fig. 3c).

The key idea is that these identifications can only create this single new RGB triangle because there is no other way.

Figure 2: The hypergraphs of queries $q_\Delta$, $q_{\text{rats}}$, $q_{\text{brats}}$, $q_T$ (Example 12). $\{R,S,T\}$ is a triad of $q_\Delta$; $\{A,B,C\}$ is a triad of $q_T$.

(b) Gadget $G_i$ is a cycle containing 2$m$ six-node segments with $12m$ RGB triangles. They can all be eliminated by removing the 6$m$ edges marked $v_i$ or the 6$m$ edges marked $\overline{v}_i$. The even numbered segments are sad because they are never used for connecting different gadgets (corresponding to clauses that use several variables); they only separate the odd ones, thus preventing spurious triangles.

(c) For clause $C_j = (v_1 \lor \overline{v}_2 \lor v_3)$, we identify vertices $b_{j+1}^{1j+1} \in G_1$ with $b_{j+1}^{2j+1} \in G_2$; $c_{j+1}^{1j+1} \in G_2$ with $c_{j+1}^{2j+1} \in G_3$ and $a_{j+1}^{3j+1} \in G_3$ with $a_{j+1}^{4j+1} \in G_1$. This RGB triangle will be deleted iff the chosen variable assignment satisfies $C_j$.

Figure 3: Gadget construction for hardness proof for $q_\Delta$. 
to get back to \( G_1 \) from \( G_2 \) in two steps. All other identifications involve different segments and so are at least six steps away. Recall that this is the reason why the even-numbered segments in the \( G_i \)'s are not used: this ensures that no spurious RGB triangles are created.

Thus, as desired, Eq. 1 holds and we have reduced 3SAT to \( \text{RES}(q_\Delta) \). \( \square \)

We next show that the tripod query \( q_T \) is also hard. We do this by reducing the triangle to the tripod. The spirit of this reduction will be used later in our main result.

**Proposition 14 (Tripod \( q_T \) is hard).** \( \text{RES}(q_T) \) and \( \text{RSP}(q_T) \) are NP-complete.

**Proof of Proposition 14.** We reduce \( \text{RES}(q_\Delta) \) to \( \text{RES}(q_T) \). It will then follow that \( \text{RES}(q_T) \) is NP-complete, and thus so is \( \text{RSP}(q_T) \) by Lemma 10. Let \((D,k)\) be an instance of \( \text{RES}(q_\Delta) \). We construct an instance \((D',k)\) of \( \text{RSP}(q_T) \) by constructing relations \( A, B, C \) as copies of \( R, S, T \) from \( D \). Define \( D' = (A, B, C, W) \) as follows:

\[
A = \{ (ab) \mid R(a,b) \in D \} \\
B = \{ (bc) \mid S(b,c) \in D \} \\
C = \{ (ca) \mid T(c,a) \in D \} \\
W = \{ ((ab), (bc), (ac)) \mid a, b, c \in \text{dom}(D) \}
\]

Here, \((ab)\) stands for a new unique domain value resulting from the concatenation of domain values \( a \) and \( b \). Observe that there is a 1:1 correspondence between the witnesses of \( D \models q_\Delta \) and the witnesses of \( D' \models q_T \). Thus, every contingency set for \( q_\Delta \) in \( D \) corresponds to a contingency set of the same size for \( q_T \) in \( D' \). Furthermore no minimum \( \Gamma' \) from \( D' \) needs to choose tuples from \( W \). If \( t = W((ab), (bc), (ac)) \) were in \( \Gamma' \), then we could replace it by \( A((ab)) \), which suffices to remove all the witnesses removed by \( t \). As we will explain later, \( A \) "dominates" \( W \) (Definition 16). It follows that \((D,k) \in \text{RES}(q_\Delta) \iff (D',k) \in \text{RES}(q_T) \). \( \square \)

Even though \( q_\Delta \) and \( q_T \) appear to be quite different, they share a key common structural property which we define next.

**Definition 15 (Triad).** A triad is a set of three endogenous atoms, \( T = \{S_0, S_1, S_2\} \) 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 \( T \).

Observe that atoms \( R, S, T \) form a triad in \( q_\Delta \) and atoms \( A, B, C \) form a triad in \( q_T \) (see Fig. 2): For example, there is a path from \( R \) to \( S \) in \( q_\Delta \) (across hyperedge \( y \)) that uses only variables (here \( y \)) that are not contained in the other atom (here \( y \notin \text{var}(T) \)).

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 16 (Domination).** If a query \( q \) has endogenous atoms \( A, B \) such that \( \text{var}(A) \subseteq \text{var}(B) \), then we say that \( A \) dominates \( B \).\(^7\)

\( \text{We already saw an example in Proposition 14: in } q_T, \text{ each of the atoms } A, B, C \text{ dominates } W. \text{ The following proposition was proved in [30]. Unfortunately however, it was claimed to hold with respect to responsibility rather than resilience. As we will see later, this proposition fails for responsibility (Proposition 39).} \)

**Proposition 17 (Domination for resilience).** Let \( q \) be an \( sj \)-free \( CQ \) and \( q' \) the query resulting from labeling some dominated atoms as exogenous. Then \( \text{RES}(q) \equiv \text{RES}(q') \).

**Proof of Proposition 17.** Let \( \Gamma \) be a minimum contingency set of \( q \) in \( D \). Suppose that atom \( A \) dominates atom \( B \) but there is some tuple \( B(t) \in \Gamma \). Let \( p \) be the projection of \( t \) onto \( \text{var}(A) \). Then we can replace \( B(t) \) by \( A(p) \) and we remove at least as many witnesses that \( D \models q \). It follows as desired, that the complexity of \( \text{RES}(q) \) is unchanged if \( B \) is exogenous, i.e., \( \text{RES}(q) \equiv \text{RES}(q') \).

When studying resilience, we follow the convention that all dominated atoms are exogenous. For example, \( A \) dominates \( R \) and \( S \) in the query \( q_{\text{rats}} \), and \( B \) dominates \( R \) and \( S \) in the query \( q'_{\text{brats}} \). We thus transform the queries so that the dominated atoms are exogenous.

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

By Proposition 17, \( \text{RES}(q_{\text{rats}}) \equiv \text{RES}(q'_{\text{rats}}) \) and \( \text{RES}(q'_{\text{brats}}) \equiv \text{RES}(q_{\text{brats}}) \).

We now prove our first main result.

**Lemma 18 (Triads make \( \text{RES}(q) \) hard).** Let \( q \) be an \( sj \)-free \( CQ \) where all dominated atoms are exogenous. If \( q \) has a triad, then \( \text{RES}(q) \) is NP-complete.

**Proof of Lemma 18.** Let \( q \) be a query with triad \( T = \{S_0, S_1, S_2\} \). We build a reduction from \( \text{RES}(q_{\text{rats}}) \) to \( \text{RES}(q) \). Given any \( D \) that satisfies \( q_{\text{rats}} \), we will produce a database \( D' \) that satisfies \( q \) such that for all \( k \):

\[
(D,k) \in \text{RES}(q_{\text{rats}}) \iff (D',k) \in \text{RES}(q) \tag{2}
\]

We will assume that no variable is shared by all three elements of \( T \) (we can ignore any such variable by setting it to a constant). Our proof splits into two cases:

- **Case 1:** \( \text{var}(S_0), \text{var}(S_1), \text{var}(S_2) \) are pairwise disjoint. Our reduction is similar to the reduction from \( q_{\Delta} \) to \( q_T \) (Proposition 14).

We first define the triad relations in \( D' \):

\[
S_0 = \{ ((ab), \ldots, (ab)) \mid R(a,b) \in D \} \\
S_1 = \{ ((bc), \ldots, (bc)) \mid S(b,c) \in D \} \\
S_2 = \{ ((ca), \ldots, (ca)) \mid T(c,a) \in D \}.
\]

Thus, each tuple of, for example, \( S_0 \) consists of identical entries with value \( (ab) \) for each pair \( R(a,b) \in D \). Thus, \( S_0, S_1, S_2 \) mirror \( R, S, T \), respectively.

To define all the other atoms \( A_i \) of \( D' \), we first partition the variables of \( q \) into 4 disjoint sets: \( \text{var}(q) = \text{var}(S_0) \cup \text{var}(S_1) \cup \text{var}(S_2) \cup V_3 \). Now for each atom \( A_i \), arrange its variables in these four groups. Then define the atom \( A'_i \) of \( D' \) as follows:

\[
A'_i = \{ ((ab), (bc), (ca), (abc)) \mid D \models q_\Delta(a,b,c) \} \tag{3}
\]

For example, all the variables \( v \in \text{var}(S_0) \) are assigned the value \( (ab) \) and all the variables \( v \in V_3 \) are assigned \( (abc) \).
By the definition of triad, there is a path from $S_0$ to $S_1$ not using any edges (variables) from $\var(S_2)$. Thus, any witness that $D' \models q$ which includes occurrences of $\langle ab \rangle$ and $\langle b'c' \rangle$ must have $b = b'$.

Similarly, a path from $S_1$ to $S_2$ guarantees that $c$ is preserved and a path from $S_2$ to $S_0$ guarantees that $a$ is preserved. It follows that the witnesses that $D' \models q$ are essentially identical to the witnesses that $D \models q_{\triangle}(x, y, z)$ (See Fig. 4).

Furthermore, any minimum contingency set only needs tuples from $S_0, S_1$ or $S_2$. Thus the sizes of minimum contingency sets are preserved, i.e., Eq. 2 holds, as desired. Thus $\text{RES}(q)$ is NP-complete.

**Case 2:** $\var(S_i) \cap \var(S_j) \neq \emptyset$ for some $i \neq j$: We generalize the construction from Case 1 as follows. Partition $\var(S_i)$ into those unshared, those shared with $S_{i-1}$, and those shared with $S_{i+1}$ (Addition is mod 3).

We then assign the relations of the triad as follows:

$$\begin{align*}
S_0 &= \{(ab); a; b \mid R(a, b) \in D\} \\
S_1 &= \{(bc); b; c \mid S(b, c) \in D\} \\
S_2 &= \{(ca); c; a \mid T(c, a) \in D\}.
\end{align*}$$

Since none of the $S_i$'s is dominated, in each case both possible values occur, e.g., $a$ and $b$ both occur in the tuples of $S_0$. Thus as in Case 1, $S_0, S_1, S_2$ capture $R, S, T$, respectively. We now partition $\var(q)$ into 7 sets as follows. The key idea is that for each assignment of $x, y, z$ to values $a, b, c$ in $D$, we will make assignments according to that partition.

$$\begin{align*}
\var(S_0) - (\var(S_1) \cup \var(S_2)) &= \langle ab \rangle \\
\var(S_1) - (\var(S_0) \cup \var(S_2)) &= \langle bc \rangle \\
\var(S_2) - (\var(S_0) \cup \var(S_1)) &= \langle ca \rangle \\
\var(q) - (\var(S_0) \cup \var(S_1) \cup \var(S_2)) &= \langle abc \rangle (4) \\
\var(S_2) \cap \var(S_0) &= a \\
\var(S_0) \cap \var(S_1) &= b \\
\var(S_1) \cap \var(S_2) &= c
\end{align*}$$

We then define each other atom $A$ in $D'$ to be the following set of tuples, where the only difference between atoms is which of the 7 members of the partition of variables occurs in $\var(A)$.

$$\{(ab); (bc); (ca); (abc); a; b; c \mid D \models q_{\triangle}(a, b, c)\}$$

By the definition of triad, there is a path from $S_0$ to $S_1$ not using any edges (variables) from $S_2$, i.e., none from $\var(S_2) \cup V_4 \cup V_6$. Thus, any witness including occurrences of some of $\langle ab \rangle, b', \langle b'c' \rangle$ must have $b = b' = b''$. Thus, as in Case 1, the witnesses of $D' \models q$ are essentially identical to the witnesses of $D \models q_{\triangle}$ and we have reduced $\text{RES}(q_{\triangle})$ to $\text{RES}(q)$.

3.2 Polynomial algorithm for linear queries

So far, we 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.

**Definition 19 (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 20 (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, $q := \langle x, y, z \rangle, S(y, z)$ (see Fig. 5).

The responsibility of linear queries is known to be in PTIME and thus by Lemma 10, resilience of linear queries is in PTIME as well.

**Fact 21 (Linear queries in PTIME [30]).** For any linear $s_j$-free CQ $q$, $\text{RSP}(q)$ (and thus also $\text{RES}(q)$) are in PTIME.

The proof of Fact 21 is that $\text{RES}(q)$ may be computed in a natural way using network flow. The same is true for computing the responsibility of tuple $t$ for $D \models q$. In the latter case, we consider each possible extensions, $e$ of $t$ that is a witness of $D \models q$, and use network flow to compute the minimum size contingency set $\Gamma$ for $t$ such that $e$ remains a witness of $D - \Gamma \models q$. The responsibility of $t$ for $D \models q$ is the minimum over all such extensions $e$ of the size of the minimum contingency set that preserves $e$.

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 (Proposition 17). This is done, for example, to the rats and brats queries:

$$\begin{align*}
q'_{\text{rats}} &:= \langle x, y, z \rangle, S(y, z), T^x(z, x) \\
q'_{\text{brats}} &:= \langle x, y, z \rangle, B(y), S^y(z, y), T^y(x, z)
\end{align*}$$

Neither $q'_{\text{rats}}$ nor $q'_{\text{brats}}$ is linear. However they can be transformed to linear queries without changing their complexity via the following transformation slightly modified from [30]:
Definition 22 (dissociation). Let $A^x$ be an exogenous atom in a query $q$, and $v \in \text{var}(q)$ a variable that does not occur in $A$. Let $q'$ be the same as $q$ except that we add $v$ to the arguments $A^x$. This transformation is called dissociation.

Example 23 (Dissociation). The above queries $q'_{\text{rats}}$ and $q'_{\text{brats}}$ have no triads but they are not linear. However, we can apply dissociation to obtain the following linear queries:

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

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

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

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

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

The key fact is that dissociation cannot decrease the complexity of resilience or responsibility.

Lemma 24 (Dissociation never simplifies [30]). If $q'$ is obtained from $q$ through dissociation, then $\text{RES}(q) \leq \text{RES}(q')$.

Proof of Lemma 24. Let $R(z)$ be the atom that has been changed to $R'(z,v)$. We reduce $\text{RES}(q)$ to $\text{RES}(q')$ by mapping $(D, k)$ to $(D', k)$ where $D'$ is the same as $D$ with the exception that we let $R' = \{ t, d \mid R(t) \in D ; d \in \text{dom}(D) \}$. This transformation does not change the witness set nor the contingency sets, because, by the way we formed $R'$ from $R$, the conjunct $R'(z,v)$ places the same restriction on $D'$ that $R(z)$ places on $D$.

The other direction does not hold, i.e., dissociation may strictly increase the complexity of the resilience of a query. It follows from Lemma 24 that if $q$ can be dissociated to a linear query, then $\text{RES}(q) \in \text{PTIME}$. In particular, the above dissociations of $q'_{\text{rats}}$ and $q'_{\text{brats}}$ prove that $\text{RES}(q'_{\text{rats}})$ and $\text{RES}(q'_{\text{brats}})$ are in $\text{PTIME}$. Since the transformations from $q_{\text{rats}}$ to $q'_{\text{rats}}$ and $q_{\text{brats}}$ to $q'_{\text{brats}}$ preserve the complexity of resilience, we thus conclude that $\text{RES}(q_{\text{rats}})$ and $\text{RES}(q_{\text{brats}})$ are easy. Later we will see that $\text{RSP}(q_{\text{rats}})$ is NP-complete (Proposition 39).

Corollary 25. $\text{RES}(q_{\text{rats}})$ and $\text{RES}(q_{\text{brats}})$ are in $\text{PTIME}$.

For responsibility, it is also true that dissociation does not decrease complexity, but the proof is more subtle, and we defer it to Sect. 5.3.

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

Lemma 26 (queries without triads are easy). Let $q$ be an $s$-$j$-free CQ that has no triad. Then $\text{RES}(q)$ is in $\text{PTIME}$.

Proof of Lemma 26. Let $q$ be a triad-free query. We prove by induction on the number of endogenous atoms in $q$ that we can transform it into a linear query by using dissociations. Since dissociations cannot decrease complexity (Lemma 24) and resilience is easy for linear queries (Fact 21), it follows that $\text{RES}(q)$ is in $\text{PTIME}$.

Base case: $q$ has fewer than three endogenous atoms. Consider $S_1, S_2$ the endogenous atoms of $q$. Using dissociation, we add all the variables to all the exogenous atoms. Thus all the exogenous atoms are identical and we can remove all but one, call it $E^x_1$. The resulting query, $q'$, is linear with ordering $S_1, E^x_1, S_2$. Thus $\text{RES}(q) \in \text{PTIME}$.

Inductive case: assume true for triad-free queries with $n$ endogenous atoms. Let $q_{n+1}$ be triad-free and have $n + 1$ endogenous atoms. We now describe a way to linearize these atoms. For each endogenous atom $S_i$, let $c_i$ be the cut of the hypergraph resulting from removing all the variables of $S_i$, i.e., all the hyperedges that touch $S_i$. These cuts are drawn as dotted vertical lines in Fig. 6.

Let $S_1$ and $S_2$ be two endogenous atoms and draw $S_2$ to the right of $S_1$. Now consider a third endogenous atom $S_3$. Since $q_{n+1}$ is connected and has no triads, there is a unique $i \in \{1, 2, 3\}$ such that the cut $c_i$ disconnects the two atoms in $\{S_1, S_2, S_3\} - \{S_i\}$.

Thus we must place $S_1$ between the other two. In other words, there is exactly one place that $S_3$ can be added to the figure: to the left of $S_1$ if $c_1$ separates $S_3$ from $S_2$; in between $S_1$ and $S_2$ if $c_3$ separates $S_1$ from $S_2$; or to the right of $S_2$ if $c_2$ separates $S_1$ from $S_3$.

For example, let $S_1(x, y)$ and $S_2(y, z)$ be the first two endogenous atoms. Let the third be $S_3(z, w)$ which shares a variable with $S_2$. Note that $c_3$ does not separate $S_1$ from $S_2$ and $c_1$ does not separate $S_2$ from $S_3$. Since $q_{n+1}$ has no triad, it must be the case that $c_2$ separates $S_1$ from $S_3$. Thus, the order in this case must be $S_1, S_2, S_3$.

Now add the remaining endogenous atoms one at a time. Since $q_{n+1}$ has no triad, by the above observation, there is exactly one place that each next endogenous atom may be placed. Finally once all the endogenous atoms have been placed, renumber them so left to right they are $S_1, S_2, \ldots, S_{n+1}$.

Define the query $q_{n+1}$ to be the result of removing all the variables in $\text{var}(S_{n+1}) - \text{var}(S_n)$ and removing all the atoms in which any of those removed variables occurred. In Fig. 6, this corresponds to removing everything to the right of $c_n$.

By our inductive hypothesis, there is a query $q_n'$ that is the result of doing some dissociations to $q_n$, and $q_n'$ is linear. Furthermore by our observation above, the ordering of the endogenous atoms remains $S_1, S_2, \ldots, S_n$.

Now, we form $q_{n+1}'$ by first adding back to $q_n$ all the variables and atoms that we removed. Note that we are thus adding back just one endogenous atom, $S_{n+1}$, together with zero or more exogenous atoms, all of which contain some variables in $\text{var}(S_{n+1}) - \text{var}(S_n)$. Finally, to all these exogenous atoms that we have just added back (if any), add all the variables in $\text{var}(S_n) \cup \text{var}(S_{n+1})$, together with any other variables occurring in any of these exogenous atoms.
Thus all the newly readded exogenous atoms are identical and we can combine them into one, call it, $E_n^x$. Note that $c_n$ still separates $E_x^x$ and $S_x^x$ from the rest of the hypergraph.

Thus, we have transformed $q_{n+1}$ to a linear query $q_{n+1}'$ such that $\text{RES}(q_{n+1}) \leq \text{RES}(q_{n+1}')$. Thus $\text{RES}(q_{n+1}) \in \text{PTIME}$ as desired. \Box

3.3 Dichotomy of resilience

Combining Lemma 18 and Lemma 26 leads to our first dichotomy result on the complexity of resilience:

**Theorem 27 (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 $\text{RES}(q)$ is \textsc{NP}-complete, otherwise it is in \text{PTIME}.

Note that it is easy to tell whether $q$ has a triad. Checking whether a given triple is a triad consists of three (linear-time) reachability problems. Thus, we get a \text{PTIME} algorithm by testing all triples of endogenous atoms.

**Corollary 28.** We can check in polynomial time in the size of the query $q$ whether $\text{RES}(q)$ is \textsc{NP}-complete or \text{PTIME}.

4. FUNCTIONAL DEPENDENCIES

Functional dependencies, such as key constraints, restrict the set of allowable data instances. In this section, we will characterize how these restrictions affect the complexity of resilience. Extending Theorem 27, we will show that, in the presence of functional dependencies, a dichotomy for the complexity of resilience still holds.

4.1 FDs can only simplify resilience

We write $\text{RES}(q; \Phi)$ to refer to the resilience problem for query $q$, restricted to databases satisfying the set of FDs $\Phi$. 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 29 (FDs do not increase complexity).** Let $q$ be an sj-free CQ and $\Phi$ a set of functional dependencies. Then $\text{RES}(q; \Phi) \leq \text{RES}(q)$.

**Proof.** The reduction is the identity function. Note that $\text{RES}(q; \Phi)$ is just the restriction of $\text{RES}(q)$ to databases satisfying $\Phi$. Thus, for all databases $D$ that satisfy $(q; \Phi)$, $(D, k) \in \text{RES}(q; \Phi) \iff (D, k) \in \text{RES}(q)$.

**Corollary 30.** If $q$ is an sj-free CQ that has no triad, and therefore $\text{RES}(q)$ is in \text{PTIME}, then $\text{RES}(q; \Phi)$ is also in \text{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 (Proposition 14). However, $q_T$ becomes polynomial when we add the FD $\varphi = x \rightarrow y$.

**Proposition 31 (FDs can reduce complexity).**

$\text{RES}(q_T; \{x \rightarrow y\}) \in \text{PTIME}.$

We will prove Proposition 31 along the way, as we learn about the effect of FD’s. Recall that $q_T := A(x), B(y), C(z), W^x(x, y, z)$. The FD $x \rightarrow y$ damages the tripod $\{A, B, C\}$ because $A$ and $B$ are no longer independent. More explicitly, once we know $x$ we also know $y$. Thus $\text{RES}(q_T; \{x \rightarrow y\}) \equiv \text{RES}(r)$ where $r := A'(x, y), B(y), C(z), W^x(x, y, z)$. Furthermore, since $B$ dominates $A'$ in $r$, $A'$ becomes exogenous: $r' := A'^x(x, y), B(y), C(z), W^x(x, y, z)$. Query $r'$ has no triad and thus is easy.

4.2 Induced rewrites preserve complexity

We call the transformation $(q_T; \{x \rightarrow y\}) \sim (r; \{x \rightarrow y\})$ an induced rewrite. Induced rewrites are key to understanding the effect of FD’s on the complexity of resilience.

**Definition 32 (Induced Rewrite: $\sim$, Closed Query).** Given a set of functional dependencies $\Phi$ and a query $q$, we write $(q; \Phi) \sim (q'; \Phi)$ to mean that $q'$ is the result of adding the dependent variable $u$ to some relation that contains all the determinant variables $v$ for some $v \rightarrow u \in \Phi$. We use $\sim$ to indicate zero or more applications of $\sim$. If $(q; \Phi) \sim (q^*; \Phi)$ and no more induced rewrites can be applied to $(q^*; \Phi)$, then we call $(q^*; \Phi)$ a closed query.

This paper began as an attempt to determine whether the dichotomy for responsibility of sj-free CQ’s [30] continues to hold in the presence of FD’s. In studying the effect of FD’s, 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 FD’s on the complexity of responsibility has been exhausted and thus there is no further change if we delete all the FD’s. We were able to prove this conjecture for unary FD’s, i.e., those of the form $v \rightarrow u$ where $v$ is a single variable.

However we had great difficulty proving this conjecture for all FD’s. We studied the responsibility problem more carefully and found that responsibility is quite delicate. In particular, we discovered errors in some of the proofs in [30]. 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 FD’s. We give that proof shortly.

With our improved insight from resilience, we went back and proved the dichotomy for responsibility and finally showed that it holds as well in the presence of FD’s. Those proofs are in Sect. 5.

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

**Lemma 33 (Induced Rewrites Preserve Complexity).** Let $q$ be a query, $\Phi$ a set of functional dependencies, and $q'$ the result of an induced rewrite, i.e., $(q; \Phi) \sim (q'; \Phi)$. Then $\text{RES}(q'; \Phi) \equiv \text{RES}(q; \Phi)$.

**Proof.** Let the change from $q$ to $q'$ be the transformation of the atom $B$ to the new atom $B'$ caused by adding variable $u$ to $B$ where $(v \rightarrow u) \in \Phi$ and $v \subseteq \text{var}(B)$.

\*Transformations of queries called rewrites were defined in [30]. An induced rewrite is a rewrite that is induced by an FD.
rewrites preserves the complexity of resilience: the presence of FDs. The following is a natural conjecture:

∀

RES

duced rewrites:

RES

q

\equiv

\triangleq

PTIME

In particular, let Φ be any set of FD’s for which \((q_\Delta, \Phi_0)\) is closed under induced rewrites. Note that since \(q_\Delta\) is closed, there can be no nontrivial unary queries such as \(x \rightarrow y\), nor any nontrivial binary queries such as \(xy \rightarrow z\). In fact, \(\Phi_0\) has no nontrivial queries, i.e., \(\Phi_0 = \emptyset\).

Now recall the reduction from \(\text{RES}(q_\Delta)\) to \(\text{RES}(q^*')\) in the proof of Lemma 18. What that proof did was to encode \(q_\Delta\) into \(q^*\). Using the triad of \(q^*\), \(T = \{S_0, S_1, S_2\}\), we partitioned the variables of \(q^*\) into 7 sets, and for each assignment of \(x, y, z\) to values \(a, b, c\) in \(D\), we made assignments according to that partition (see Eq. 4).

The net effect, is that just as for \(q_\Delta\), since \((q; \Phi)\) is closed, it must be the case that \(D' \models q\). For example, if \(\Phi\) contains the FD, \(u \rightarrow v\) where \(u\) is contained in one of the 7 partitions, then since \((q; \Phi)\) is closed, \(u\) must be in the same partition and thus it has exactly the same value as each of the variables in \(u\).

If \(u\) has variables from at least two different elements of the partition, then either closure implies that all the variables of \(u\) come from a single triad atom, and thus \(v\) does as well, or together the two partitions that meet \(u\) include all three values \(a, b, c\) and thus \(v\) is determined.

Thus, we have shown that the reduction \(f\) is also a reduction from \(\text{RES}(q_\Delta)\) to \(\text{RES}(q^*, \Phi)\) and thus the latter problem is NP-complete. □

### 4.4 Dichotomy of resilience with FDs

Recall that FD’s cannot increase the complexity of resilience and thus if \(q\) has no triad, then \(\text{RES}(q; \Phi) \in \text{PTIME}\) (Corollary 30). Thus, we have succeeded in proving the dichotomy for resilience in the presence of FD’s:

**Theorem 37 (FD Dichotomy).** Let \((q; \Phi)\) be a sf-free conjunctive query with functional dependencies. Let \((q^*; \Phi)\) be its closure under induced rewrites, and such that all dominated atoms of \(q^*\) are marked as exogenous. If \(q^*\) has a triad then \(\text{RES}(q; \Phi)\) is NP-complete. Otherwise, \(\text{RES}(q; \Phi) \in \text{PTIME}\).

Note that we have also proved Conjecture 35:

**Corollary 38 (Induced rewrites suffice).** Let \((q; \Phi)\) be a sf-free conjunctive query with functional dependencies, and let \(q^*\) be the closure of \(q\) under induced rewrites. Then, \(\text{RES}(q; \Phi) \equiv \text{RES}(q^*; \Phi) \equiv \text{RES}(q^*)\).

### 5. Completeness 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 but in the end the final theorems are similar.

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

\[
q_{rats} := A(x), R(x, y), S(y, z), T(z, x)
\]

\[
q'_{rats} := A(x), R^*(x, y), S(y, z), T^*(z, x)
\]

We saw earlier that \(\text{RES}(q_{rats})\) is in \(\text{PTIME}\) (Corollary 25). The reason is that atom \(A\) dominates \(R\) and \(T\) and thus the complexity of \(\text{RES}(q_{rats})\) is unchanged when we make \(R\) and \(T\) exogenous (Proposition 17), i.e., \(\text{RES}(q_{rats}) \equiv \text{RES}(\text{rats}')\).

Obviously \(q'_{rats}\) is triad free. Thus, by Theorem 27, \(\text{RES}(q'_{rats})\) and \(\text{RES}(q_{rats})\) are in \(\text{PTIME}\). We now show, however, that \(\text{RSF}(q_{rats})\) is NP-complete.

**Proposition 39.** \(\text{RSF}(q_{rats})\) is NP-complete.
PROOF. We reduce 3SAT to \( \text{RSP}(\text{qats}) \). Let \( \psi \) be a 3-CNF formula with variables \( v_1, \ldots, v_n \) and clauses \( C_1, \ldots, C_m \). The reduction will map \( \psi \) to \((D, s_0, k)\), with \( s_0 = S(b_0, c_0) \), where we will construct \( D = (A, R, S, T) \) to have a contingency set for \( s_0 \) of size \( k \) iff \( \psi \in \text{3SAT} \). We let \( a_0 \) be the unique element of the domain of \( D \) that joins with \( s_0 \).

In \( \text{qats} \), \( A \) dominates \( R \), but when we are building a contingency set \( \Gamma \) for \( s_0 \), we may require some tuples of the form \((a_0, b)\) from \( R \). Note that these cannot be replaced by the tuple \( a_0 \) from \( A \), because that would remove all the witnesses from \( D - \Gamma \). This explains why \( \text{RSP}(\text{qats}) \) is \( \text{NP} \)-complete, and it is the key idea behind the reduction we now produce.

For each variable \( v_t \) occurring in \( \psi \), we build the gadget \( G_t \) as follows. \( G_t \) consists of 2\( t \) \( y \) values and 2\( t \) \( z \) values, \( b_j^t, c_j^t \), \( 1 \leq j \leq 2t \) where \( t \) is a constant to be specified later. We include the 2\( t \) pairs \( R(a_0, b_j^t) \) and the 2\( t \) pairs \( T(c_j^t, a_0) \), \( 1 \leq j \leq 2t \). (See Fig. 7 where these pairs are drawn as edges from \( a_0 \) to each \( b_j^t \) and from each \( c_j^t \) to \( a_0 \), respectively.)

Next, we include all the pairs \( S(b_j^t, c_j^t) \), \( 1 \leq j \leq 2t \). These are drawn in Fig. 7 as a complete bipartite graph between the vertex sets \( \{b_0, \ldots, b_t^t\} \) and \( \{c_0, \ldots, c_t^t\} \).

Finally we add two matchings of size \( t \) which we name the \( v_t \) matching and the \( \mathcal{v} \) matching:

\[
\begin{align*}
v_t \text{ matching} & : S(b_j^t, c_{j+1}^t), \ldots, S(b_t^t, c_{2t}^t) \\
\mathcal{v} \text{ matching} & : S(b_{t+1}^t, c_1^t), \ldots, S(b_{2t}^t, c_1^t)
\end{align*}
\]

Any minimum contingency set must remove all of the witnesses from \( G_t \). Such a minimum contingency set must remove either all the pairs \( R(a_0, b_j^t) \) or all the pairs \( T(c_j^t, a_0) \) or both the pairs \( T(c_j^t, a_0), T(c_j^t, a_0) \), i.e., one side or the other of the complete bipartite graph. After this, \( t \) witnesses remain. Any minimum contingency set must include the \( v_t \) matching, corresponding to choosing \( v_t \) true, or the \( \mathcal{v} \) matching, corresponding to choosing \( v_t \) false.

So far, we have described the gadgets \( G_1, \ldots, G_n \) and shown that any minimum contingency set for this part of \( D \) corresponds to a truth assignment for the variables \( v_1, \ldots, v_n \). We next introduce the clause gadgets and choose the value \( k \), so that contingency sets for \( D \) of size \( k \) will correspond exactly to truth assignments that satisfy all of the clauses of \( \psi \).

We now describe the clause gadgets. Suppose, for example, that \( C_s = v_1 \lor \mathcal{v} \lor v_3 \). Then 7 of the eight possible truth assignments to \( v_1, v_2, v_3 \) satisfy \( C_s \), i.e., all but the assignment \( \alpha_2 \) (010 in binary). For each of these 7 good assignments, \( \alpha_i \), \( 0 \leq i \leq 2, 2 < i \leq 7 \), we add an element \( a_{s,i} \) to \( A \) and we add the tuples to \( R \) and \( T \) so that \( a_{s,i} \) participates in exactly one witness in each of the three variable gadgets that agree with assignment \( \alpha_i \). For example, assignment \( \alpha_0 \) (110 in binary) makes \( v_1, v_2 \) true and \( v_3 \) false, so \( a_{s,0} \) joins with \( S(b_{t+1}^1, c_{t+r(s,0)}^1), S(b_{t+1}^2, c_{t+r(s,0)}^2), \) and \( S(b_{t+1}^3, c_{t+r(s,0)}^3) \) (See Fig. 8). Here \( r(s, i) \) is a function that chooses a unique element of the matching \( v_i \) or \( \mathcal{v} \) appropriate to assignment \( \alpha_i \) of clause \( s \). The key property of the \( C_s \) gadget is that if the chosen truth assignment satisfies \( C_s \), then we may choose only 6 \( a_{s,i} \)’s from \( A \) for the contingency set. We do not need to worry about the \( a_{s,i} \)’s corresponding to the chosen assignment. However, if the chosen assignment does not satisfy \( C_s \), then all 7 of the \( a_{s,i} \)’s must be chosen.

![Figure 7: The qats variable \( v_t \) gadget, \( G_t \): dotted lines are sad pairs which will never be chosen in minimum contingency sets.](image)

![Figure 8: The qats clause gadget corresponding to clause \( C_s = v_1 \lor \mathcal{v} \lor v_3 \). \( A(a_{s,6}) \) is omitted from \( \Gamma \) if the chosen truth assignment happens to be \( \alpha_6 = \{(v_1, 1), (v_2, 1), (v_3, 0)\} \).](image)
We can let $t = 8m$ and $k = 2nt + 6m = (16n + 6)m$, in which case we can let $r(s,i) = 8(s-1)+i$. Our construction insures that $(D, s, k) \in \text{RSP}(q_{rats})$ iff $\psi \in 3\text{SAT}$. 

The proof of Proposition 39 shows that domination does not work the same way for responsibility as it does for resilience. In particular, the analogy of Proposition 17 for responsibility does not hold.

We next show that some version of domination still works for responsibility. Recall the query $q_{rats}$. Note that $\text{var}(A) \subseteq \text{var}(R)$ and $\text{var}(B) \subseteq \text{var}(R)$ and that in fact each variable of $R$ is contained in $\text{var}(D)$ for some atom that dominates $R$.

$$q_{rats} := A(x), R(x,y), B(y), S(y,z), T(z,x)$$

$$q_{rats}^+: := A(x), R^+(x,y), B(y), S(y,z), T(z,x)$$

**Lemma 40.** The complexity of responsibility for $q_{rats}$ is unchanged if we make $R$ exogenous, i.e.,

$$\text{RSP}(q_{rats}) \equiv \text{RSP}(q_{rats}^+)$$.

Proof. Let $D \models q_{rats}$ and let $t$ be a tuple that joins with a witness that $D \models q_{rats}$. We will show that there is a minimum contingency set $\Gamma$ for $t$ that contains no pairs from $R$. Let $\Gamma$ be a minimum contingency set for $t$ that contains as few pairs from $R$ as possible. Suppose that $R(a_1, b_1) \in \Gamma$. Let $j$ be a witness that $(D - \Gamma) \models q_{rats}$ and let $a_0, b_0, c_0$ be the projection of $j$ onto components $x, y, z$, respectively. Thus, $A(a_0), R(a_0, b_0)$ and $B(b_0)$ are all in $D - \Gamma$. In particular, $R(a_1, b_1) \neq R(a_0, b_0)$. Let $\Gamma'$ be the result of replacing $R(a_1, b_1)$ by $A(a_1)$ if $a_1 \neq a_0$, and by $B(b_1)$ otherwise, in which case $b_1 \neq b_0$. Thus $\Gamma'$ is still a minimum contingency set for $t$ and it contains fewer pairs from $R$, contradicting the fact that $\Gamma$ had the fewest possible such pairs. Thus, pairs from $R$ are never needed in any minimum contingency set for $t$. Thus, as claimed, the complexity of $\text{RSP}(q_{rats})$ is unchanged when we make $R$ exogenous.

We are now ready to define full domination, the version of domination that works for responsibility the way that ordinary domination works for resilience. The idea is that every relevant variable from the fully dominated atom $F$ must be contained in some smaller atom $A$ such that $\text{var}(A) \subseteq \text{var}(F)$.

**Definition 41 (Full Domination).** Let $F$ be an atom of query $q$. $F$ is fully dominated iff there exist strict subsets of $\text{var}(F)$, $V_1, \ldots, V_m$, such that $I$ is the set of $F$’s isolated variables, i.e., they occur in no atom besides $F$, $\text{var}(F) = V_1 \cup \ldots \cup V_m \cup I$, and each $V_i$ is the variable set for some endogenous atom $A_i$ in $q$, i.e., $V_i = \text{var}(A_i)$.

For example $R$ is fully dominated in $q_{rats}$:

$$\text{var}(R) = \{x\} \cup \{y\}.$$ 

Note that $I = \emptyset$ because $R$ has no isolated variables. On the other hand, $R$ is not fully dominated in $q_{rats}$ because $y$ is not covered by any smaller atom.

Note that isolated variables can affect the resilience and responsibility of queries. For example, in the $q_{rats}$ query, the pair $R(a,b)$ might appear in some minimum contingency set $\Gamma$ for some database $D$. However, in the modified query, $q := A(x), R'(x,y,w), S(y,z), T(z,x)$, $R'$ has isolated variable $w$. Suppose that $R'(a,b,d_1)$ and $R'(a,b,d_2)$ are the tuples in $D'$ that start with $a,b$. Then the single pair $R(a,b) \in \Gamma$ might be replaced by the two triples, $R'(a,b,d_1), R'(a,b,d_2)$, in which case it is larger and might no longer be minimum.

We now show, however, that fully dominated atoms may be made exogenous and, at the same time, their isolated variables may be removed.

**Lemma 42.** Let $F$ be a fully dominated atom in an $sJ$-free CQ $q$. Let $q'$ be the modified query in which $F$ is replaced by $F^*$, an exogenous atom whose isolated variables are any that have been removed. Then $\text{RSP}(q) \equiv \text{RSP}(q')$.

Proof. We have to show that $\text{RSP}(q) \leq \text{RSP}(q')$ and $\text{RSP}(q') \leq \text{RSP}(q)$. Suppose we are given $(D,t,k)$ where $t$ is a tuple that joins with a witness that $D \models q$. Let $D', t'$ be the modifications of $D, t$ in which we project out $F'$’s isolated variables if any. Conversely, if we are given $(D', t', k')$ then we let $(D, t)$ be the database and tuple resulting from adding back any isolated variables of $F'$, populated as fixed constants. There are two cases. In each case we will show how to choose values of $k, k'$ such that

$$(D, t, k) \in \text{RSP}(q) \iff (D', t', k') \in \text{RSP}(q')$$ (6)

**Case 1:** $t$ is not a tuple of $F$. We show that as in the proof of Lemma 40, there is no need to include any tuples from $F$ in a minimum contingency set $\Gamma$ for $D$. As in that proof, we let $j$ be a witness that $(D - \Gamma) \models q$ and suppose that $F(f) \in \Gamma$. Thus, $j$ and $f$ must differ on some non-isolated variable $v$ of $F$. Let $A$ be the atom that covers $v$ and we can replace $F(f)$ by the tuple $\pi_{\text{var}(A)}(f)$ of $A$. Thus, the sizes of the minimum contingency sets on the two sides are identical and letting $k = k'$, Eq. 6 holds.

**Case 2:** $t$ is a tuple of $F$. In this case, some tuples of $F$ may need to be in $\Gamma$. Let $I$ be the isolated variables of $F$ and let $W = \{ f \in F \mid f \neq t \land \pi_I(f) = \pi_I(t) \}$. These are the tuples of $F$ which agree with $t$ on all but the isolated variables of $F$. $W$ must be in every contingency set for $D, t$. Thus, we let $k = k' + |W|$ and Eq. 6 holds.

**5.1 Triads and hardness**

Now that we have established that full domination works for responsibility, 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 has no isolated variables. 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 $\text{var}(S_1) \subseteq \text{var}(S_2)$ because removing $\text{var}(S_2)$ would isolate $S_1$. Thus $(R, S, T)$ is the unique triad of $q_{rats}$.

On the other hand, $R$ is fully dominated in $q_{rats}$, so we transform it to triad-free $q_{rats}^+$.

$$q_{rats}^+ := A(x), R^+(x,y), B(y), S(y,z), T(z,x)$$.

We now show that $\text{RSP}(q)$ is NP-complete if $q$ has a triad (Lemma 43). Then we will show that otherwise $\text{RSP}(q) \in \text{PTime}$ (Corollary 50). The proofs will take the same form as for resilience, however the proof of Lemma 43 is a bit more subtle than the analogous result for resilience.

**Lemma 43.** Let $q$ be an $sJ$-free CQ where all fully-dominated atoms are exogenous. If $q$ has a triad, then $\text{RSP}(q)$ is NP-complete.
There is no endogenous atom A such that \( \text{var}(A) \subseteq \text{var}(S_i) \cap \text{var}(S_j) \), for some \( i \neq j \). We will show that \( \text{RSP}(q) \leq \text{RSP}(q) \).

Given \( D, t, k \) we must produce \( D', t', k' \) such that
\[
(D, t, k) \in \text{RSP}(q) \iff (D', t', k') \in \text{RSP}(q). \tag{7}
\]

Note that we may assume that \( t = R(a_0, b_0) \) for some values \( a_0, b_0 \), i.e., that \( t \) is a tuple from \( R \), because this case was proved hard in Proposition 13.

In this case, we construct \( D' \) exactly as we did in Lemma 18 (Cases 1 or 2). The only difference is that we must define \( t' \) from \( t \). This is easy: recall that \( t = R(a_0, b_0) \). We let \( t' = S_0((a_0, b_0), a_0, b_0) \), i.e., the corresponding tuple of \( S_0 \).

Thus, we have exactly simulated \( q \), in \( q \), so Eq. 7 holds.

Case 2: There is an endogenous atom \( A \) and some \( i \neq j \), such that \( \text{var}(A) \subseteq \text{var}(S_i) \cap \text{var}(S_j) \), but only for a unique pair \( i \neq j \). We show that \( \text{RSP}(q) \leq \text{RSP}(q) \). Let the pair be \( 0,2 \), i.e., \( \text{var}(A) \subseteq \text{var}(S_0) \cap \text{var}(S_2) \).

Again, we are given \( D, t, k \), where \( t = R(a_0, b_0) \). We produce \( D', t', k' \), but now such that,
\[
(D, t, k) \in \text{RSP}(q) \iff (D', t', k') \in \text{RSP}(q). \tag{8}
\]

We produce \( D' \) and \( t' \) exactly as in Case 1, and we again have that all the witnesses and minimum contingency sets for \( q \) wrt \( D, t, k \) are preserved for \( q \) wrt \( D', t', k' \). Thus Eq. 8 holds.

Finally, we are left with,
\[\text{Case 3: There are endogenous atoms } A, B \text{ such that } \text{WLOG } \text{var}(A) \subseteq \text{var}(S_0) \cap \text{var}(S_2), \text{ and } \text{var}(B) \subseteq \text{var}(S_0) \cap \text{var}(S_1).\]

We know that \( S_0 \) is not fully dominated. Thus, there must exist a non-isolated variable \( w \in \text{var}(S_0) \) such that \( w \not\in \text{var}(A) \cup \text{var}(B) \). Again, since \( S_0 \) is not fully dominated, there must be an endogenous atom \( C \neq S_0 \) such that \( w \in \text{var}(C) \). Thus we have located a tripod sitting in the hypergraph of \( q \) (see Fig. 9). It thus follow from Proposition 14, that \( \text{RSP}(q) \) is NP complete as well.

5.2 The polynomial case

As we saw in the previous section, the presence of triads in a query makes its 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 \( q_{\text{rats}} \) is an example: for resilience we use domination and after applying domination, \( q_{\text{rats}} \) has no triads and thus \( \text{RSP}(q_{\text{rats}}) \in \text{PTIME} \). However, if we may only apply full domination, then \( q_{\text{rats}} \) keeps the triad \( R, S, T \) and thus \( \text{RSP}(q_{\text{rats}}) \) is NP-complete.

We now want to prove the polynomial case for responsibility. Recall that in the proof of Lemma 26, we showed the following

**Corollary 44.** Let \( q \) be a conjunctive query 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 an sj-free CQ easier (Lemma 24), it followed that \( \text{RES}(q) \in \text{PTIME} \) for any such triad-free query, \( q \).

To prove that for any triad-free, sj-free CQ, \( q, \text{RSP} \in \text{PTIME} \), 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 \( \text{RSP}(q) \leq \text{RSP}(q') \). In the proof of the similar result for resilience we did the following. We let \( R^X(z) \) be the atom that was changed to \( R^X(z, v) \). We then reduced \( \text{RES}(q) \) to \( \text{RES}(q') \) by mapping \( (D, k) \) to \( (D', k) \) where \( D' \) is the same as \( D \) with the exception that we let \( R^X = \{ t, d \mid R(t) \in D, d \in \text{dom}(D) \} \). This transformation does not change the witness set nor the contingency sets, because, by the way we formed \( R' \) from \( R \), the conjunct \( R'(z, v) \) places the same restriction on \( D' \) that \( R(z) \) places on \( D \).

This proof goes through fine for responsibility except in one case, namely if the tuple \( t \) that we are computing the responsibility of belongs to \( R \), the exogenous relation to which we have added the new variable, \( v \).

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

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

**Definition 45 (tuples with wildcards).** Let \( D \) be a database containing a relation, \( R(x_1, \ldots, x_n) \). Let \( \tau = (s_1, \ldots, s_n) \) be a tuple such that each \( s_i \in \text{dom}(D) \cup \{ * \} \), i.e., \( \tau \) may have elements in the domain in some coordinates and the wildcard, \( * \), in others. We call \( \tau \) a tuple with wildcards. We say that a tuple \( (a_1, \ldots, a_n) \in R \) matches \( \tau \) iff for all \( i \),

\[1\] 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.
\( \alpha_i = s_i \) or \( s_i = \ast \). When \( D \) and \( R \) are understood, \( \tau \) represents a set of tuples from \( R \), \( \langle \tau \rangle = \{ a \in R \mid a \text{ matches } \tau \} \).

For example, the tuple with wildcard, \((a, \ast)\), 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 46 (\( \text{RSP}^\ast \)).** Let \( D \) be a database containing a relation, \( R \), \( q \) a query for \( D \) and \( \tau \) a tuple with wildcards. Then \((D, \tau, k) \in \text{RSP}^\ast(q)\) iff there exists a contingency set \( \Gamma \) of size \( k \) such that \((D - \Gamma) ⊑ q \) and \((D - (\Gamma ∪ \langle \tau \rangle)) \not\ni q \).

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

**Corollary 47.** Let \( q \) be an \( s_j \)-free \( \text{CQ} \) all of whose fully dominated atoms are exogenous. If \( q \) has a triad then \( \text{RSP}^\ast(q) \) is \( \text{NP} \)-complete.

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

**Lemma 48.** If \( q' \) is obtained from \( q \) through dissociation, then \( \text{RSP}^\ast(q) \leq \text{RSP}^\ast(q') \).

Furthermore, linear queries are still easy for responsibility:

**Lemma 49.** For any linear \( sj \)-free \( \text{CQ} \) \( q \), \( \text{RSP}^\ast(q) \) is in \( \text{PTIME} \).

**Proof.** The proof is a small modification of the proof for Fact 21. We again consider each possible extension, \( \epsilon \) of an element of \( \langle \tau \rangle \) that is a witness of \( D ⊑ q \), and use network flow to compute the minimum size contingency set \( \Gamma \) for \( \langle \tau \rangle \) such that \( \epsilon \) remains a witness that \( D - \Gamma \ni q \).

The responsibility of \( t \) for \( D \ni q \) is the minimum over all such extensions \( \epsilon \) of the size of the minimum contingency set that preserves \( \epsilon \). This involves using network flow to compute a min cut where we assign 0 to each edge corresponding to a tuple in \( \langle \tau \rangle \), \( \infty \), to the remaining edges of \( \epsilon \), and 1 to all other edges.

**Corollary 50.** If \( q \) has no triad, then \( \text{RSP}^\ast(q) \) can be made linear by using dissociations, and is thus in \( \text{PTIME} \). Therefore so is \( \text{RSP}(q) \).

We have thus proved our desired dichotomy for responsibility, and as a bonus, we have proved it for responsibility with wildcardss as well:

**Theorem 51.** Let \( q \) be an \( sj \)-free \( \text{CQ} \), and let \( q' \) be the result of making all fully dominated atoms exogenous. If \( \text{R}(q') \) contains a triad then \( \text{RSP}(q) \) and \( \text{RSP}^\ast(q) \) are \( \text{NP} \)-complete. Otherwise, \( \text{RSP}(q) \) and \( \text{RSP}^\ast(q) \) are \( \text{PTIME} \).

It follows from Corollary 50 and Corollary 47 that \( \text{RSP}^\ast(q) \equiv \text{RSP}(q) \) for all \( sj \)-free \( \text{CQ} \), \( q \). Note that it is not at all clear how one would build a reduction from \( \text{RSP}^\ast(q) \) to \( \text{RSP}(q) \). However, our characterization of the complexity of \( \text{RSP}(q) \) and \( \text{RSP}^\ast(q) \) gives us this result: After all fully dominated atoms are made exogenous, if there is a triad, then \( \text{RSP}(q) \) is \( \text{NP} \)-complete, thus so is \( \text{RSP}^\ast(q) \). If there is no triad, then \( \text{RSP}^\ast(q) \in \text{PTIME} \), thus so is \( \text{RSP}(q) \):

**Corollary 52.** For all \( sj \)-free \( \text{CQ} \), \( q \), we have \( \text{RSP}(q) \equiv \text{RSP}^\ast(q) \).

### 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 53.** Let \( (q, \Phi) \) be a \( sj \)-free conjunctive query with functional dependencies. Let \( (q^*, \Phi) \) be its closure under induced rewrites, and such that all fully dominated atoms of \( q^* \) are marked as exogenous. If \( q^* \) has a triad then \( \text{RSP}(q; \Phi) \) is \( \text{NP} \)-complete. Otherwise, \( \text{RSP}(q; \Phi) \in \text{PTIME} \).

**Proof.** Since FDs only make \( \text{RSP}(q) \) easier, we know that if \( q^* \) has no triad then \( \text{RSP}(q^*) \) is easy, thus so is \( \text{RSP}(q^*; \Phi) \) and thus equivalently \( \text{RSP}(q; \Phi) \). For the converse, we just show that just as in the proof of Lemma 36, the reduction, \( f \), from one of \( \text{RSP}(q_{\triangledown}), \text{RSP}(q_{\text{rats}}), \text{RSP}(q_{\text{tr}}) \) to \( \text{RSP}(q) \) which we built in Lemma 43 always produces databases, \( D' \) that satisfy \( \Phi \). The proof is almost exactly as in Lemma 36. Note that in the proof of Lemma 43, we use the same reduction in all three cases, i.e., no matter if we are reducing from \( \text{RSP}(q_{\triangledown}), \text{RSP}(q_{\text{rats}}), \) or \( \text{RSP}(q_{\text{tr}}) \).

### 6. RELATED WORK

Sections 1 and 2 have extensively discussed prior work and the connections between resilience, deletion propagation and responsibility [7, 11, 27, 28]. 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, 19]. 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, 25]. 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 special cases of view updates.

**Causality.** The study of causality is important in many areas other than databases, for example in Artificial Intelligence and philosophy. Although an intuitive concept, it is difficult to formally define causality and many authors have presented possible definitions of causality. In our prior work, the notions of causality and responsibility were strongly inspired by the work of Halpern and Pearl [10, 20]. Causal reasoning is based on the idea of interventions: understand how changes of input variables affect an outcome, and thus relates in spirit to resilience. In the case of resilience, the intervention is the deletion of input tuples. In Section 7 we provide some additional discussion on how resilience can address some applications of causality, and it has the benefit that it is easier to compute than responsibility.

**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 [30, 31], 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 \cite{32,35}. 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 \cite{2,4,5,17,26,33}. Finally, the problem of explaining missing query results \cite{8,21,23,34} is a problem analogous to deletion propagation, but in this case, we want to add, rather than remove tuples from the view. In this paper, we focused the definition of resilience with respect to tuple deletions; extending it to handle other kinds of updates is the topic of future work.

7. DISCUSSION AND OUTLOOK

Summary. This paper presents dichotomy results for the resilience and responsibility of self-join-free conjunctive queries. Our results extend and generalize previous complexity results on the problem of deletion propagation with source side-effects and causal responsibility.

Using resilience for responsibility. We have noted that resilience also has implications in other, more general applications. One such application is deriving the most likely causes of query results: Responsibility provides a measure of the causal contribution of an input tuple to a query output. Ranking input tuples based on their responsibilities is used in our prior work \cite{30} to identify likely causes: tuples at the top of the ranking are the most likely causes, whereas tuples low in the ranking are less likely. Even though resilience cannot derive an equivalent to the responsibility ranking, it can identify the top tuples in this ranking. An optimal resilience set of tuples $\Gamma$ is guaranteed to comprise of tuples with the highest responsibility values. This is easy to see: If there were a tuple $t \notin \Gamma$ with a higher responsibility value, then there would exist $\Gamma'$ such that $\{t\} \cup \Gamma'$ would be a solution to resilience, and $|\Gamma'| < |\Gamma| - 1$, which is a contradiction.

Therefore, all tuples in a resilience set are at the top of the responsibility ranking. However, there may be additional tuples with equivalent responsibility that are not part of the selected resilience set $\Gamma$. These can also be derived with a simple algorithm: For each tuple $t \notin \Gamma$, remove $t$ from the dataset and solve for resilience. If the solution is $|\Gamma'| < |\Gamma|$, this means that all tuples in $\{t\} \cup \Gamma$ have equal responsibilities. This way, each tuple can be classified as being one of the top-ranked responsibility tuples or not. This observation has interesting implications: even though responsibility is harder to compute that resilience (Section 2.4), one can still derive the top causes by solving resilience instead of responsibility. We intend to investigate this application in future work.

8. REFERENCES

\begin{thebibliography}{1}
\bibitem{5} G. Bender, L. Kot, and J. Gehrke. Explainable security for relational databases. SIGMOD, pp. 1411–1422, 2014.
\end{thebibliography}


