QUERY-ANSWER CAUSALITY IN DATABASES AND ITS CONNECTIONS WITH REVERSE REASONING TASKS IN DATA AND KNOWLEDGE MANAGEMENT

by
Babak Salimi

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

at

CARLETON UNIVERSITY

Ottawa, Ontario
December, 2015

Abstract

Causality is an important notion that appears at the foundations of many scientific disciplines, in the practice of technology, and also in our everyday life. Causality is crucial to understanding and managing *uncertainty* in data, information, knowledge, and theories. In data management in particular, there is a need to represent, characterize and compute the causes that explain why certain query results are obtained or not, or why natural semantic conditions, such as integrity constraints, are not satisfied.

The notion of query-answer causality in databases was introduced in [Meliou et al., 2010c]. This notion is shown to be general enough to be applied to a broad class of database-related applications, such as explaining unexpected answers to a query result, diagnosing network malfunctions, data cleaning, hypothetical reasoning [Meliou et al., 2010c, Meliou et al., 2010b, Meliou et al., 2011b, Meliou, et al., 2011c].

In this thesis, we establish and investigate connections between query-answer causality and other important forms of reasoning that appear in data management and knowledge representation, e.g. consistency-based diagnoses [Reiter, 1987], database repairs and consistent query answering [Arenas et al., 1999], abductive diagnosis [Console et al., 1991b, Eiter et al., 1995], and the view-update problem [Buneman et al., 2002, Kimelfeld, 2012a, Kimelfeld, 2012b]. These problems are classified in [Meliou et al., 2011a] as *reverse data management problems*.

The unveiled relationships allow us to obtain new results for query-answer causality and also for the above mentioned related areas. Furthermore, we argue that causality in data management can be seen as a very fundamental concept, to which many other data management problems and notions are connected. In fact, we suggest causality as a unifying framework for reverse data management problems.

Acknowledgements

Foremost, I would like to express my special appreciation and gratitude to my advisor, Professor Dr. Leopoldo Bertossi, who has been a tremendous mentor for me. I would like to thank him for encouraging my research and for allowing me to grow as a research scientist. His advice on both research as well as on my career have been invaluable.

I would also like to thank my committee members, Professor Rene J. Miller, Professor Amy Felty, Professor Douglas J. Howe and Professor Patrick Farrell for serving as my PhD Committee members. I am grateful to them all for making my defense a pleasant and enriching experience; and also for their many useful comments.

I gratefully acknowledge the funding sources that made my Ph.D. work possible. I was funded by an RA through NSERC Discovery, and the NSERC Strategic Network on Business Intelligence (BIN).

To my friends and roommates: thank you for listening, offering me advice, and supporting me through this entire process. Special thanks to Sina Ariyan, Mostafa Milani, Misagh Tavanpour and Sima Jamali. The debates, dinners, and game nights as well as editing advice, financial supports, and general help and friendship are all greatly appreciated.

My gratitude to my grandmother and my parents for their infinite patience and implicit faith in my capabilities is boundless and cannot be expressed in sufficient words. This thesis is especially dedicated to my wonderful (late) mother. **Mom I love you**, I remember your pure love and care for me, and I will never forget.

Table of Contents

List of Figures

Chapter 1

Introduction

The central aim of many disciplines of knowledge, ranging from philosophy through law, psychology, economics, statistics, etc. to computer science, is the elucidation of cause-effect relationships among variables or events. There have been many attempts to define causality that can be traced back at least to Hume (1739), but as yet no consensus has been reached and the debate continues to the present [Halpern, 2015a].

The advent of the *structural model framework* (see [Pearl, 2009] for a discussion and references) has sharpened many of the debate points and placed the study of causality within a much broader context. It has led to several closely-related accounts of *actual causation*,[1] among which the one by [Halpern & Pearl, 2005] (HP from now on) has been most influential.

A distinctive feature of the HP-model is its reliance on *structural equations* to model causal relationships. The use of structural equations as a model for causal relationships was well known before [Pearl, 2009, Pearl, 2010]. However, the details of the framework that have proved so influential are due to Pearl. The structural equations express the effects of *interventions*: "What happens to the outcome if we change the state of the input?". This is a *"what-if question"*. The mirror question: "How can we change the input to achieve a particular outcome?", is a *"how-to question"*, and can be addressed by assessing the results of a collection of what-if questions. These two forms of analysis are at the essence of causal reasoning.

According to the HP approach, a domain is described in terms of random variables and their values. Causal relationships between those variables, say $A, B, ...$, are modeled by a set of structural equations. The set of random variables (with their values) and the associated structural equations is called a *causal model* [Halpern & Pearl, 2005]. In this context, the statement "A is an actual cause for B" claims that there is a set of possible

[1]As opposed to general causal claims such as "smoking causes cancer" which refer to a class of events, actual causation specifies a particular instantiation of a causal relation e.g., Joe's smoking caused his cancer.

interventions (contingencies) on the causal model that makes B *counterfactually* depend on A. That is, had A not happened, B wouldn't have happened.

Example 1.0.1 Suppose that a committee consists of two members, Alice and Bob; and a particular proposal is approved by the committee if Alice or Bob vote for it.

According to the structural model approach, such a scenario can be modeled by means of two boolean random variables A and B, which may take values 0 or 1. They represent Alice and Bob's vote, respectively. Value 1 is a vote for approval. There is an additional boolean random variable, P, representing the decision by the committee. P takes value 1 iff the proposal is approved.

The causal dependencies between these variables are expressed through a function representing that the value of P is determined by the value of $A \vee B$. We may consider the following possible worlds:

(a) A world where Alice votes for the proposal, but Bob votes against, i.e. $A = 1$ and $B = 0$. In this case, $P = 1$. Moreover, P counterfactually depends on the value of A, i.e. had A not been 1, the proposal wouldn't have been approved. So, A is the single actual cause for P.

(b) In another world, both Alice and Bob vote for the proposal, in which P is "over-determined" by A and B, and there is no counterfactual dependence of P upon any of A or B, separately. However, according to the HP-model, both A and B are actual causes for P. For instance, B counterfactually depends on A, in the contingency that A is 0. In other words, the proposal approval counterfactually depends on Bob's positive vote, under the contingency that Alice had not been voted for the proposal. Notice that, this contingency is in correspondence with the "intervention" of setting A to 0 in the model. $\square$

When multiple causes over-determine an outcome (cf. Example 1.0.1(b)) it is natural to consider a graded notion of causation on the basis of their *responsibility*. Building on the HP-model, [Chockler & Halpern, 2004] proposed a formal account for *causal responsibility*. Intuitively, causal responsibility is a numerical function of the minimum number of changes (the interventions) in the model that make a factor a counterfactual cause. The more changes needed, the less responsibility of the factor.

In computer science, most of the work on causality has been developed in the context of knowledge representation, but not much has been done about causality in data management, where there is the need to represent, characterize and compute causes that explain why certain query results are obtained or not, or why natural semantic conditions, such as integrity constraints, are not satisfied. Causality can also be used to explain the contents of a view, i.e. of a predicate with virtual contents that is defined in terms of other physical, materialized relations (tables).

Furthermore, in a world of big, uncertain data, the need to understand data beyond simple query answering, introducing explanations in different forms, to extract high-level information from datasets, has become particularly relevant. Thus, a major research challenge in data management is the development of tools that assist users in explaining observed data-related phenomena.

The study of causality in data management was explicitly started in [Meliou et al., 2010c, Meliou et al., 2010b], and for query results in relational databases. There, the HP-model is adopted and investigated. The definition of query-answer causality is shown to be general enough to be applied to a broad class of database-related applications, e.g. explaining unexpected query-answers, diagnosing network malfunctions, data cleaning, hypothetical reasoning, and data provenance [Meliou et al., 2010c, Meliou et al., 2010b, Meliou et al., 2011b, Meliou, et al., 2011c]. (Provenance of a query on a database is an expression that describes how the answer was derived from the tuples in the database.) In this work we start from this approach, concentrating on causality as defined for relational databases.

1.1 Query-Answer Causality

When querying a database, a user may not always obtain the expected results, and the system could provide some explanations. They could be useful to further understand the data or check if the query is the intended one. The notion of causality-based explanation for a query result was introduced in [Meliou et al., 2010c]. We will refer to this notion as *query-answer causality* (or simply, *QA-causality*).

In this work we will consider only *monotone* queries. For them, the set of answers (monotonically) grows with the database instance.

Intuitively, a database atom (or simply, a tuple) τ is an *actual cause* for an answer $\bar{a}$

to a monotone query $\mathcal{Q}$ from a relational database instance D if there is a "contingent" subset of tuples Γ, accompanying τ, such that, after removing Γ from D, removing τ from $D \smallsetminus \Gamma$ causes $\bar{a}$ to switch from being an answer to being a non-answer (i.e., not being an answer). In other words, τ is an actual cause for $\bar{a}$ if there is an intervention (in the form of tuple deletions) that makes τ pivotal for $\bar{a}$. Here "pivotal" means the query answer counterfactually depends on τ.

Actual causes and contingent tuples are usually restricted to be among a pre-specified set of *endogenous tuples*, which are admissible, possible candidates for causes, as opposed to *exogenous tuples*. Actually, exogenous tuples provide the context or the background theory for the problem, and are considered as external factors that are not of interest to the current problem statement or beyond our control. Since no intervention is conceivable on exogenous tuples, they can not be included in any contingency set or be an actual cause. In other words, it is assumed that they are included in all conceivable hypothetical states of a database.

The endogenous/exogenous partition is application-dependent and captures predetermined factors, such as users' preferences that may affect QA-causal analysis. For example, certain tuples or full tables might be identified as irrelevant (or exogenous) in relation to a particular query at hand (cf. Example 1.1.4) or decided to be exogenous or endogenous a priori, independently from the query. In most of this work, we assume the latter case, unless otherwise stated. Accordingly, the partition is always assumed to be one of the fixed parameters of the QA-causality related problem under consideration.

A cause τ may have different associated contingency sets Γ. Intuitively, the smaller they are the strongest is τ as a cause (it needs less company to undermine the query answer). So, some causes may be stronger than others. This idea is formally captured through the notion of *causal responsibility*, and introduced in [Meliou et al., 2010c]. It reflects the relative degree of actual causation. According to this proposal, the responsibility of a tuple τ for a query answer, $\rho(\tau)$, is defined as the numerical value $\frac{1}{1+|\Gamma|}$, where Γ is a minimum-cardinality contingency set associated to τ. In applications involving large data sets, it is crucial to rank potential causes according to their responsibilities [Meliou et al., 2010c, Meliou et al., 2010b].

Example 1.1.1 Consider an instance D with relations $\mathit{Author}(\mathit{AuName}, \mathit{JName})$ and

$Journal(JName, Topic, \#Paper)$, and contents as below:

Author	AuName	JName
	Joe	TKDE
	John	TKDE
	Tom	TKDE
	John	TODS

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

The conjunctive query:

$$\mathcal{Q}(AuName, Topic): \quad \exists JName\, \exists \#Paper(Author(AuName, JName) \wedge Journal(JName, Topic, \#Paper)) \tag{1.1}$$

has the following answers:

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

Assume $\langle John, XML\rangle$ is an unexpected answer to $\mathcal{Q}$. That is, it is not likely that *John* has a paper on *XML*. Now, we want to compute causes for this unexpected observation. For the moment assume all tuples in D are endogenous.

It holds that *Author(John, TODS)* is an actual cause for $\langle John, XML\rangle$. Actually, it has two contingency sets, namely: $\Gamma_1 = \{Author(John, TKDE)\}$ and $\Gamma_2 = \{Journal(TKDE, XML, 30)\}$. That is, *Author(John, TODS)* is a counterfactual cause for $\langle John, XML\rangle$ in both $D \smallsetminus \Gamma_1$ and $D \smallsetminus \Gamma_2$. Moreover, the responsibility of *Author(John, TODS)* is $\frac{1}{2}$, because its minimum-cardinality contingency sets have size 1.

Tuples *Journal(TKDE, XML, 30)*, *Author(John, TKDE)* and *Journal(TODS, XML,* 32*)* are also actual causes for $\langle John, XML\rangle$, with responsibility $\frac{1}{2}$.

For a more subtle situation, assume only *Author* tuples are endogenous, possibly reflecting the fact that the data in *Journal* table are more reliable than those in the *Author* table. Under this assumption, the only actual causes for $\langle John, XML\rangle$ are *Author(John, TKDE)* and *Author(John, TODS)*. □

View-conditioned causality (vc-causality) was proposed in [Meliou et al., 2010b, Meliou, et al., 2011c] as a restricted form of QA-causality, to determine causes for unexpected query results, but conditioned to the correctness of prior knowledge about view extensions, that do not have to be altered by interventions.

Example 1.1.2 (ex. 1.1.1 cont.) Consider again the answer $\langle \mathit{John}, \mathit{XML} \rangle$ to $\mathcal{Q}$ in (1.1). Suppose this answer is unexpected and likely to be wrong, while all other answers are known to be correct. In this case, it makes sense that, to determine the causes for $\langle \mathit{John}, \mathit{XML} \rangle$, only those contingency sets whose removal does not affect the correct answers are admissible. In other words, the hypothetical states of the database D that do not provide the correct answers (while eliminating $\langle \mathit{John}, \mathit{XML} \rangle$) are not considered. □

Apart from the explicit use of causality, most of the related research on explanations for query results has concentrated on the notion of *data provenance*, which is also known as *lineage*, or *pedigree*. Roughly speaking, the provenance of a query on a database is an expression that describes how the answer was derived from the tuples in the database. See [Buneman et al., 2001, Buneman & Tan, 2007, Cheney et al., 2009a, Cui et al., 2000, Karvounarakis, 2010, Tannen, 2013] for more on provenance, and more recently, [Chapman & Jagadish, 2009, Huang et al., 2008, Riddle et al., 2014, Glavic et al., 2015], for provenance for “non-answers”, which is about tracing back, sometimes through the interplay of tuple annotations, the reasons for *not* obtaining a possibly expected answer.

In the literature, different types of data provenance have been identified and investigated (see [Glavic et al., 2007, Simmhan et al., 2005] for surveys, and also [Glavic & Miller, 2011]). In this work, we are interested in the *minimal-witness-basis* approach, or *why-provenance* [Buneman et al., 2001].[2] We will refer to this form of provenance as *lineage*. Informally, the lineage of an answer $\bar{a}$ to a monotone query $\mathcal{Q}$ from a relational database instance D is defined as the set of all *minimal-witnesses* for $\bar{a}$; where a minimal-witness for $\bar{a}$ is a subset-minimal subinstance $W \subseteq D$, such that $W \models \mathcal{Q}(\bar{a})$.

Example 1.1.3 (ex. 1.1.1 cont.) Consider again the answer $\langle \mathit{John}, \mathit{XML} \rangle$ to $\mathcal{Q}$ in (1.1). This answer has the following minimal-witnesses: $W_1 = \{\mathit{Author}(\mathit{John}, \mathit{TKDE})$, $\mathit{Journal}(\mathit{TKDE}, \mathit{XML}, \mathit{30})\}$ and $W_2 = \{\mathit{Author}(\mathit{John}, \mathit{TODS}), \mathit{Journal}(\mathit{TODS}, \mathit{XML},$ $\mathit{32})\}$. Therefore, the lineage of this answer is $\{W_1, W_2\}$. □

[2] This definition of why-provenance is closely related to PosBool semi-ring [Tannen, 2010] (see also [Green, 2009]), which define provenance in terms of tuple annotations and a set of propagation rules for them.

A close connection between causality and lineage has been established in [Meliou et al., 2010c]. Roughly speaking, both definitions refer to the same tuples if all tuples in a database are endogenous. For illustration, in Example 1.1.1, the actual causes of $\langle John, XML\rangle$ coincide with the tuples in the union of W_1 and W_2, the minimal-witnesses for this answer obtained in Example 1.1.3. However, causality is a more refined notion that identifies causes for query results on the basis of user-defined criteria, and ranks causes according to their responsibility [Meliou et al., 2010c]. That is, QA-causality can be seen as an additional analytical step beyond the lineage of a query answer. This is particularly relevant when the size of the lineage of a query answer (i.e. the cardinality of the minimal-witness set) is large, and it becomes tedious and impractical to manually examine it [Meliou et al., 2011b].

The following example is borrowed from [Meliou et al., 2010c], and shows the contribution of QA-causality to lineage. (We reuse this example, because we will retake it in Chapter 9, where its results will be compared with those provided by a new form of numerical quantification of causal contribution of a tuple to a query answer that we introduce in that chapter.)

Example 1.1.4 Timothy Walter ("Tim") Burton is an American film director, film producer, writer and animator. He is famous for fantasy, dark, quirky-themed movies. A user wishes to learn more about Burton's movies and queries IMDB to find out genres of movies that he has directed. The IMDB (database) schema includes the following predicates: $Director(Did, Firstname, Lastname)$, $Movie(Mid, Name, Year, Rank)$, $Genre(Mid, Genre)$, $Movie\_Director(Did, Mid)$.

The following conjunctive query asks for the genres of all movies directed by "Burton" from the IMDB dataset:

$$\begin{aligned} \mathcal{Q}(Genre) \,\text{:}\!\!-\quad & \exists\, Did\, \exists Mid\, \exists Firstname\, \exists Name\, \exists Year\, \exists Rank \\ & (Director(Did, Firstname, \mathsf{Burton}) \wedge Movie\_Director(Did, Mid, Year, Rank) \\ & \wedge\, Movie(Mid, Name) \wedge Genre(Mid, Genre)). \end{aligned} \tag{1.2}$$

The followings are some of the answers:

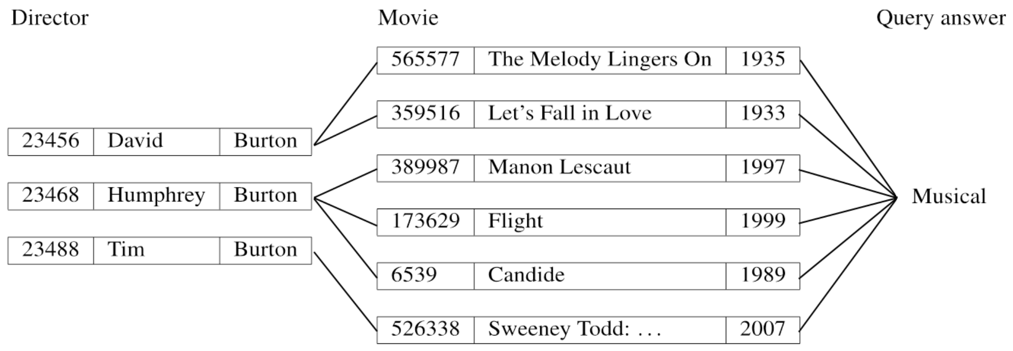

Figure 1.1: Lineage of the answer *Musical* from the *Director* and *Movie* tables

Genre
...
Fantasy
History
Horror
Music
Musical
Mystery
Romance
...

Answers *Fantasy* and *Horror* are quite expected, but *Music* and *Musical* are surprising. So, the user wants to know the reason for these latter answers. Examining the lineage of a surprising answer would be the first step towards finding reasons. However, in this case, the union of the lineages of the answers *Music* and *Musical* contains 137 database tuples, which is rather large to make sense of by the user.

QA-causality provides more informative explanations for surprising answers than raw lineage information. This is done by imposing an exogenous/endogenus partition (to filter out uninteresting elements of the lineage), and rank causes based on their causal responsibility (to avoid less important tuples). We show this only for the answer *Musical*, which has a smaller lineage than *Music*.

A reasonable partition in this setting is to consider tuples in the *Directors* and *Movies* tables as endogenous, and the others as exogenous. This partition reflects the fact that the user is only interested in movies and directors that contribute to the genre Musical.

The lineage of the answer *Musical* from *Directors* and *Movies* tables is depicted in Figure 1.1. In this case, the lineage is represented as a collection of pairs of tuples from the *Director* and *Movie* tables that, together with some exogenous tuples whose presence is implicitly represented by the edges between the *Movies* and *Director* tuples in Figure 1.1), provide minimal-witnesses for the answer.

The actual causes for the answer *Musical* ranked by their responsibility scores are as follows:

ρ	Actual Causes for *Musical*
0.33	*Movie(526338, "Sweeney Todd", 2007)*
0.33	*Director(23456, David, Burton)*
0.33	*Director(23468, Humphrey, Burton)*
0.25	*Director(23488, Tim, Burton)*
0.25	*Movie(359516, "Lets Fall in Love", 1933)*
0.25	*Movie(565577, "The Melody Lingers On", 1935)*
0.20	*Movie(6539, "Candide", 1989)*
0.20	*Movie(173629, "Flight", 1999)*
0.20	*Movie(389987, "Manon Lescaut", 1997)*

In [Meliou et al., 2010c], the responsibility scores are interpreted as follows:
(a) "Sweeney Todd" at the top of the list is the only musical by Tim Burton. They argued that being this tuple at the top of the list shows the power of causal responsibility to explain the unexpected observation.
(b) The next three tuples in the list are for directors with last name "Burton". Those tuples are also interesting, because they indicate that the query may have been ambiguous.
(c) The tuples at the bottom of the list are for movies. They argued that since directors are more interesting than movies to explain the query answer, causal responsibility correctly ranked movies at the bottom. $\square$

1.2 Causality, Model-Based Diagnosis and Database Repairs

Model-based diagnosis [Struss, 2008, sec. 10.3], an area of knowledge representation, addresses the problem of, given the *specification* of a system in some logical formalism and a usually unexpected *observation* about the system, obtaining *explanations* for the observation, in the form of a diagnosis for the unintended behavior. Model-based diagnosis comes in two forms: *consistency-based* [Reiter, 1987] and *abductive diagnosis* [Console et al., 1991b, Eiter et al., 1995]. A comparison between these two approaches has been provided in [Poole, 1989, Console et al., 1991b].

Under consistency-based diagnosis, a diagnosis is a set of abnormality assumptions about the system components, such that the system specification and the observation are consistent under these assumptions. On the other hand, an abductive diagnosis is a set of assumptions that must be discovered so that together with the system specification entail the observation.

In a different direction, a database instance, D, that is expected to satisfy certain *integrity constraints* may fail to do so.[3] In this case, a *repair* of D is a database D' that

[3]Integrity constraints are conditions that capture the semantics of data and are expected to be satisfied by

does satisfy the integrity constraints and *minimally departs* from D. Different forms of minimality can be applied and investigated. A *consistent answer* to a query from D and with respect to the integrity constraints is a query answer that is obtained from all possible repairs, i.e. is invariant or certain under the class of repairs. These notions were introduced in [Arenas et al., 1999].

The three forms of reasoning we have seen so far, namely inferring causes from databases, model-based diagnosis, and consistent query answering (and repairs) are all *non-monotonic* [Salimi & Bertossi, 2014]. For example, a (most responsible) cause for a query result may not be such anymore after the database is updated. Furthermore, they all reflect some sort of *uncertainty* about the information at hand. In this work, we establish natural, precise, useful, and deeper connections between these three reasoning tasks.

More precisely, we unveil a strong connection between computing causes and their responsibilities for conjunctive query answers, on one hand, and computing repairs in databases with respect to *denial constraints*, on the other. (Denial constraints are integrity constraints that prohibit a particular kind of combination of database tuples (cf. Section 2.1).) These computational problems can be reduced to each other. In order to obtain repairs with respect to *a set* of denial constraints from causes, we investigate causes for queries that are *unions of conjunctive queries*, and develop algorithms to compute causes and responsibilities.

We show that inferring and computing actual causes and their responsibilities in a database setting become consistency-based diagnosis reasoning problems and tasks. Actually, a causality-based explanation for a conjunctive query answer can be viewed as a consistency-based diagnosis, where in essence the first-order logical reconstruction of the relational database provides the system description [Reiter, 1984], and the observation is the query answer. We also establish a bidirectional connection between consistency-based diagnosis and repairs. The connection with the latter is used to extend the complexity analysis of QA-causality started in [Meliou et al., 2010c], from conjunctive queries without self-joins to unions of conjunctive queries.

As opposed to consistency-based diagnoses, which is usually practiced with specifications in first-order predicate logic, abductive diagnosis is commonly done with some sort of

a database in order to keep its correspondence with the outside reality it is being modeled (cf. Section 2.1).

logic programming-based specifications [Denecker et al., 2002, Eiter et al., 1997, Gottlob et al., 2010]. In particular, Datalog can be used as specification language, giving rise to Datalog-abduction [Gottlob et al., 2010].

The definition of QA-causality applies to monotone queries [Meliou et al., 2010c, Meliou et al., 2010b]. However, all complexity and algorithmic results in [Meliou et al., 2010c] have been restricted to conjunctive queries. Datalog queries [Ceri et al., 1989, Abiteboul et al., 1995], which are also monotone, but may contain recursion, require investigation in the context of QA-causality. In this work, we establish a relationship between Datalog-abduction and QA-causality, which allows us to obtain complexity results for QA-causality for Datalog queries.

1.3 Causality and Delete-Propagation

We also explore fruitful connections between QA-causality and the classical and important *view-update problem* in databases [Abiteboul et al., 1995], which is about updating a database through views. An important aspect of the problem is that one wants the base relations (also called "the source database" in this context) to change in a minimal way while still producing the intended view-updates. This is a problem of update propagation of tuples, from views to base relations.

The *delete-propagation* problem [Buneman et al., 2002, Cong et al., 2006, Kimelfeld, 2012a, Kimelfeld, 2012b] is a particular case of the view-update problem where only tuple deletions are allowed from the views. If the views are defined by monotone queries, only source deletions can give an account of view deletions. When only a minimal set of deletions (under set inclusion) from the base relations is expected to be performed, we are in the "minimal-source-side-effect" case. The "minimum-source-side-effect" case appears when that set is required to have a minimum-cardinality. In a different case, we may want to minimize the side-effects on the view, requiring that other tuples in the (virtual) view contents are not deleted [Buneman et al., 2002] (the *minimal-viw-side-effect deletion-problem*).

In this research, we provide a precise connection between different variants of the delete-propagation problem and QA-causality. In particular, we show that the minimal-source-side-effect deletion-problem is related to the QA-causality problem. The minimum-source-side-effect deletion-problem is related to the *most-responsible cause problem* (see

[Salimi & Bertossi, 2015a]). The view-side-effect-free deletion-problem turns out to be related to *view-conditioned causality*.

The connection between QA-causality, abductive diagnosis and the delete-propagation problems allows us to adopt and adapt established results for the last two in the context of QA-causality, obtaining some new complexity results for QA-causality.

1.4 QA-Causality and Reverse Data Management

A *data transformation* is a function from an input data source to an output data source [Meliou et al., 2011a]. The natural evolution of data follows the directionality of the transformations, i.e. from source to target. Most data management tasks fall under this forward paradigm from a variety of perspectives: query processing, data integration, data mining, clustering and indexing.

However, the focus of many reasoning tasks in database management, such as causality, abduction, view-updates, mapping inversion in data exchange [Arenas et al., 2010, Fagin, 2007, Arenas et al., 2003b] , data provenance and database repairs, is to invert a *data transformation process*, in order to reason diagnostically back about an observation. Problems of this kind are classified in [Meliou et al., 2011a] as *reverse data management problems*.

The results obtained in this research, and reported in [Bertossi & Salimi, 2014, Salimi & Bertossi, 2014, Salimi & Bertossi, 2015a, Salimi & Bertossi, 2015b, Salimi & Bertossi, 2015c], show that causal reasoning is a central activity in reverse data management problems. In fact, causality can provide a unifying foundation for reverse data engineering process. Having a unifying foundation for these problems opens the playground to adopt (and possibly adapt) the established results from one area to the others. It also opens up the possibility of lifting the established concepts and results from causality to reverse data management.

We should mention that many of these mutually related problems have been treated differently in the literature, and similar results have been reproduced. For instance, as we show in this work, causal responsibility as in [Meliou et al., 2010b], minimal-source-side-effect deletion-problem [Buneman et al., 2002], cardinality-based repairs [Lopatenko & Bertossi, 2007], consistency-based diagnoses from theories expressed in terms of disjunctive positive rules (cf. Section 5.5.3) are very similar problems. Intuitively, in all of them the aim is to make the minimal/minimum changes on the input so that the output enjoys a particular property. In other words, they are sorts of how-to analysis in the causal reasoning terms.

1.5 Degree of Causal Contribution

The notion of causal responsibility in [Meliou et al., 2010c] intends to provide a metric to quantify the causal contribution of a tuple to a query answer. This is a numerical degree of causality, and responsibility-based ranking of causes that is considered as one of the most important contributions of HP-causality to data management [Meliou et al., 2010c, Meliou et al., 2011b].

The basic idea behind causal responsibility is that the smaller the number of tuples are deleted from the database to make a tuple τ a counterfactual cause, the more responsible τ is for the query answer.

A close connection between causal responsibility and other notions in databases, e.g. minimal-source-side-effect deletion-problem and cardinality-based repairs, is unveiled in this work (see [Salimi & Bertossi, 2015a, Salimi & Bertossi, 2015b, Cibele et al., 2016]). The underlying reason for these connections is that these notions are all motivated by the need to perform a minimum number of changes in the database so that the new state of the database has a desired property. Therefore, causal responsibility is indeed an important notion that can capture and unify several problems in data management.

However, in Chapter 9, we argue in technical terms that causal responsibility as introduced in [Meliou et al., 2010c] may only partially fulfil the original intention of plausibly ranking tuples in terms of causal contribution to an answer. We illustrate the point with an informal example.

Example 1.5.1 (ex. 1.1.4 cont.) It turns out that query $\mathcal{Q}$ in (1.2) is ambiguous with respect to the user intention. This is because it is a query about "Burton" rather than "Tim Burton". Looking at the lineage of the answer *Musical* (cf. Figure 1.1), we observe that "Humphrey Burton" (as represented in the corresponding tuple in the *Director* relation) contributes more to the genre "Musical" (the query answer) than "David Burton" and "Tim Burton". This is because he has more movies in that category than the others. In other words, among them, “Humphrey Burton” is contained in more minimal-witnesses for the answer *Musical*. Accordingly, we expect a plausible ranking of tuples in terms of their degree of causal contribution to place “Humphrey Burton” at the top.

However, we already obtained that the three directors (or their tuples) have the same causal responsibility for the query answer; and they all require the same minimum number of changes on the database to become a counterfactual cause for the answer. $\square$

The issue is that, in addition to the “minimum number of changes” required to make a tuple τ a counterfactual cause, there are other factors that have to be taken into account. In this work, specifically in Chapter 9, we define and investigate an intuitive metric, called “*degree of causal contribution*”, that also takes into account factors such as number of minimal-witnesses (of an answer) that τ is contained in. We also show that this new notion enjoys interesting properties. In particular, it does rank “Humphrey Burton” higher than the other two directors.

Chapter 2

Background

2.1 Preliminaries

We consider relational database schemas of the form $\mathcal{S} = (U, \mathcal{P})$, where U is the possibly infinite database domain of *constants* and $\mathcal{P}$ is a finite set of *database predicates*[1] of fixed arities. A database instance D compatible with $\mathcal{S}$ can be seen as a finite set of ground atomic formulas (in databases aka. atoms or tuples), of the form $P(c_1, ..., c_n)$, where $P \in \mathcal{P}$ has arity n, and $c_1, \ldots, c_n \in U$. The *active domain* of an instance D, $Adom(D)$, is the set of constants from U that appear in D.

A *conjunctive query* (CQ) is a formula $\mathcal{Q}(\bar{x})$ of the first-order (FO) logic language, $\mathcal{L}(\mathcal{S})$, associated to $\mathcal{S}$ of the form $\exists \bar{y}(P_1(\bar{s}_1) \wedge \cdots \wedge P_m(\bar{s}_m))$, where the $P_i(\bar{s}_i)$ are atomic formulas, i.e. $P_i \in \mathcal{P}$, and the $\bar{s}_i$ are sequences of terms, i.e. variables or constants. The $\bar{x}$ in $\mathcal{Q}(\bar{x})$ shows all the free variables in the formula, i.e. those not appearing in $\bar{y}$. If $\bar{x}$ is non-empty, the query is *open*. If $\bar{x}$ is empty, the query is *boolean* (a BCQ), i.e. the query is a sentence, in which case, it is true or false in a database, denoted by $D \models \mathcal{Q}$ and $D \not\models \mathcal{Q}$, respectively. A sequence $\bar{c}$ of constants is an answer to an open query $\mathcal{Q}(\bar{x})$ if $D \models \mathcal{Q}[\bar{c}]$, i.e. the query becomes true in D when the variables are replaced by the corresponding constants in $\bar{c}$. We denote the set of all answers to an open conjunctive query $\mathcal{Q}(\bar{x})$ with $\mathcal{Q}(D)$.

A query $\mathcal{Q}$ is *monotone* if for every two instances $D_1 \subseteq D_2$, $\mathcal{Q}(D_1) \subseteq \mathcal{Q}(D_2)$, i.e. the set of answers grows monotonically with the instance. For example, CQs and unions of CQ (UCQs) are monotone queries. Datalog queries [Ceri et al., 1989, Abiteboul et al., 1995], although not FO, are also monotone (cf. Section 2.2.1 for more details). In this work we consider only monotone queries.

An *integrity constraint* is a sentence of language $\mathcal{L}(\mathcal{S})$, and then, may be true or false

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

in an instance for schema $\mathcal{S}$. Given a set IC of integrity constraints, a database instance D is *consistent* if $D \models IC$; otherwise it is said to be *inconsistent*. In this work we assume that sets of integrity constraints are always finite and logically consistent.

A particular class of integrity constraints is formed by *denial constraints* (DCs), which are sentences κ of the form: $\forall \bar{x} \neg (A_1(\bar{x}_1) \wedge \cdots \wedge A_n(\bar{x}_n))$, where $\bar{x} = \bigcup \bar{x}_i$ and each $A_i(\bar{x}_i)$ is a database atom, i.e. predicate $A \in \mathcal{P}$. (The atoms may contain constants.) Denial constraints are exactly the negations of BCQs. Sometimes we use the common representation of DCs as "negative rules" of the form: $\leftarrow A_1(\bar{x}_1), \cdots, A_n(\bar{x}_n)$.

Another class of ICs is formed by *inclusion dependencies* (INDs), which are sentences of the form $\forall \bar{x}(P_i(\bar{x}) \rightarrow \exists \bar{y} P_j(\bar{x}', \bar{y}))$, with $\bar{x}' \cap \bar{y} = \emptyset, \bar{x}' \subseteq \bar{x}$. Another special class of ICs is formed by *functional dependencies* (FDs). For example, $\psi : \forall x \forall y \forall z (P(x,y) \wedge P(x,z) \rightarrow y = z)$ specifies that the second attribute of P functionally depends upon the first. (If A, B are the first and second attributes for P, usual notation for this FD is $\psi : A \rightarrow B$.) Actually, this FD is also a *key dependency* (KD), in the sense that the combination of the attributes on the LHS of the arrow functionally determines all the other attributes of the predicate.

2.2 QA-Causality and its Decision Problems

In this section we review the notion of QA-causality as introduced in [Meliou et al., 2010c]. We also summarize the main decision and computational problems that emerge in this context and the established results for them.

2.2.1 Causality and responsibility

In the rest of this work, unless otherwise stated, we assume that a relational database instance D is split in two disjoint sets, $D = D^n \cup D^x$, where D^n and D^x are the sets of *endogenous* and *exogenous* tuples, respectively.

A tuple $\tau \in D^n$ is a *counterfactual cause* for an answer $\bar{a}$ to $\mathcal{Q}(\bar{x})$ in D if $D \models \mathcal{Q}(\bar{a})$, but $D \smallsetminus \{\tau\} \not\models \mathcal{Q}(\bar{a})$. A tuple $\tau \in D^n$ is an *actual cause* for $\bar{a}$ if there exists $\Gamma \subseteq D^n$, called a *contingency set*, such that τ is a counterfactual cause for $\bar{a}$ in $D \smallsetminus \Gamma$. $\mathit{Causes}(D, \mathcal{Q}(\bar{a}))$ denotes the set of actual causes for $\bar{a}$. If $\mathcal{Q}$ is boolean, $\mathit{Causes}(D, \mathcal{Q})$ contains the causes for answer yes.

Notice that $\mathit{Causes}(D, \mathcal{Q}(\bar{a}))$ is non-empty when $D \models \mathcal{Q}(\bar{a})$, but $D^x \not\models \mathcal{Q}(\bar{a})$, reflecting the fact that endogenous tuples are required for the answer.

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

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

The *causal responsibility* of a tuple τ for answer $\bar{a}$, denoted $\rho_{_{\mathcal{Q}(\bar{a})}}(\tau)$, is $\frac{1}{(|\Gamma|+1)}$, where $|\Gamma|$ is the size of the smallest contingency set for τ. When τ is not an actual cause for $\bar{a}$, no contingency set is associated to τ. In this case, $\rho_{_{\mathcal{Q}(\bar{a})}}(\tau)$ is defined as 0.

Example 2.2.1 Consider $D = D^n = \{R(a_4, a_3), R(a_2, a_1), R(a_3, a_3), S(a_4), S(a_2), S(a_3)\}$, and the query $\mathcal{Q} : \exists x \exists y (S(x) \wedge R(x, y) \wedge S(y))$. It holds: $D \models \mathcal{Q}$.

Tuple $S(a_3)$ is a counterfactual cause for $\mathcal{Q}$. If $S(a_3)$ is removed from D, $\mathcal{Q}$ is not true anymore. Therefore, the responsibility of $S(a_3)$ is 1. Besides, $R(a_4, a_3)$ is an actual cause for $\mathcal{Q}$ with contingency set $\{R(a_3, a_3)\}$. If $R(a_3, a_3)$ is removed from D, $\mathcal{Q}$ is still true, but further removing $R(a_4, a_3)$ makes $\mathcal{Q}$ false. The responsibility of $R(a_4, a_3)$ is $\frac{1}{2}$, because its smallest contingency sets have size 1. Likewise, $R(a_3, a_3)$ and $S(a_4)$ are actual causes for $\mathcal{Q}$ with responsibility $\frac{1}{2}$.

For the same $\mathcal{Q}$, but with $D = \{S(a_3), S(a_4), R(a_4, a_3)\}$, and the partition $D^n = \{S(a_4), S(a_3)\}$ and $D^x = \{R(a_4, a_3)\}$, it turns out that both $S(a_3)$ and $S(a_4)$ are counterfactual causes for $\mathcal{Q}$. □

In [Meliou et al., 2010c], causality for non-query answers is defined on the basis of sets of *potentially missing tuples* that account for the missing answer. Causality for non-answers becomes a variation of causality for answers. In this work we do not consider non-query answers.

We will concentrate mostly on CQs without built-ins. However, the definitions of actual cause and contingency set can be applied without a change to *monotone queries* in general [Meliou et al., 2010c], in particular to *unions of BCQs* (UBCQs), with or without built-ins, and Datalog queries, possibly with recursion.

A Datalog query $\mathcal{Q}(\bar{x})$ is a whole program Π consisting of Horn rules that accesses an underlying extensional database D. We may assume that Π defines an answer-collecting

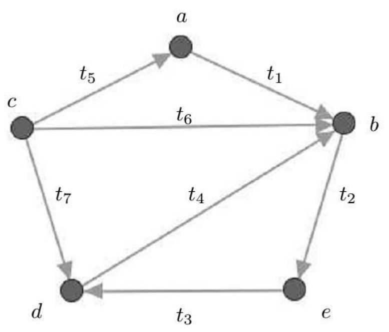

Figure 2.1: Graph G associated to the instance D in Example 2.2.2

predicate $Ans(\bar{x})$ by means of a top rule of the form $Ans(\bar{x}) \leftarrow P_1(\bar{s}_1), \ldots, P_m(\bar{s}_m)$ (i.e., all the predicates in the RHS are defined by other rules in Π or are database predicates for D). Here, the $\bar{s}_i$ are lists of variables or constants, and $\bar{x} \subseteq \bigcup_i \bar{s}_1$.

Now, $\bar{a}$ is an answer to query Π on D when $\Pi \cup D \models Ans(\bar{a})$. Here, entailment ($\models$) means that the RHS belongs to the minimal model of the LHS. So, the extension, $Ans(\Pi \cup D)$, of predicate Ans in the minimal model of the program contains the answers to the query. The Datalog query is boolean if the top answer-predicate is propositional, with a definition of the form $ans \leftarrow P_1(\bar{s}_1), \ldots, P_m(\bar{s}_m)$. In this case, the query is true if $\Pi \cup D \models ans$, equivalently, if ans belongs to the minimal model of $\Pi \cup D$ [Ceri et al., 1989, Abiteboul et al., 1995].

CQs can be expressed as Datalog queries. For example, (1.1) can be expressed in Datalog as:

$$Ans_{\mathcal{Q}}(AuName, Topic) \leftarrow Author(AuName, Jname),\ Journal(JName, Topic, \#Paper).$$

The definition of QA-causality can be applied without any conceptual changes to Datalog queries. In the case of Datalog, we sometimes use the notation $Causes(D, \Pi(\bar{a}))$ or $Causes(D, \Pi)$, depending on whether Π is boolean.

Example 2.2.2 Consider the instance D with a single binary relation E as below (t_1-t_7 are tuple identifiers). Assume all tuples are endogenous.

Instance D can be represented as the directed graph $G(\mathcal{V}, \mathcal{E})$ in Figure 2.1, where $\mathcal{V}$ coincides with the active domain of D (i.e., the set of constants in E), and $\mathcal{E}$ contains an edge (v_1, v_2) iff $E(v_1, v_2) \in D$. The tuple identifiers are used as labels for the corresponding edges in the graph. For simplicity, we will refer to the database tuples through their identifiers.

E	A	B
t_1	a	b
t_2	b	e
t_3	e	d
t_4	d	b
t_5	c	a
t_6	c	b
t_7	c	d

Consider the Datalog query Π:

$$\begin{aligned} Ans(x,y) &\leftarrow P(x,y) \\ P(x,y) &\leftarrow E(x,y) \\ P(x,y) &\leftarrow P(x,z), E(z,y), \end{aligned}$$

which collects pairs of vertices of G that are connected through a path.

It is easy to see that, $\langle c, e\rangle$ is an answer to query Π on D. That is, $\Pi \cup D \models Ans(c,e)$. This is because there are three distinct paths between c and e in G. All tuples except for t_3 are actual causes for this answer i.e., $Causes(E, \Pi(c,e)) = \{t_1, t_2, t_4, t_5, t_6, t_7\}$. Intuitively, this is because all of these tuples contribute to at least one path between c and e. Among them, t_2 has the highest responsibility. Because, t_2 is a counterfactual cause for this answer. $\square$

There are three main computational problems in database causality. For a boolean monotone query $\mathcal{Q}$ and database D:

(a) The *causality problem* (CP) is about computing the actual causes for $\mathcal{Q}$.

(b) The *responsibility problem* (RP) is about computing the responsibility, $\rho_D(\tau)$, of a given actual cause τ.

Since a tuple that is not an actual cause has responsibility 0, this problem subsumes (a).

(c) Computing the *most responsible actual causes* (MRC).

These problems have corresponding *decision versions*. The complexity of these decision problems have been investigated in [Meliou et al., 2010c, Cibele et al., 2016]. Here we recall those results that we will use throughout this work. The first is the causality problem, about deciding whether a tuple is an actual cause for a query answer.

Definition 2.2.1 For a boolean monotone query $\mathcal{Q}$, the *causality decision problem* (*CDP*) is (deciding about membership of):

$$\mathcal{CDP}(\mathcal{Q}) := \{(D, \tau) \mid \tau \in D^n, \text{and } \tau \in \mathit{Causes}(D, \mathcal{Q})\}. \qquad \square$$

This problem is tractable for *CQ*s [Meliou et al., 2010c]. In this work we extend the complexity analysis to UCQs and Datalog queries.

The next is the responsibility problem, about deciding responsibility of a tuple for a query answer being above a given threshold.

Definition 2.2.2 For a boolean monotone query $\mathcal{Q}$, the *responsibility decision problem* (*RDP*) is (deciding about membership of):

$$\mathcal{RDP}(\mathcal{Q}) = \{(D, \tau, v) \mid \tau \in D^n, v \in \{0\} \cup \{\tfrac{1}{k} \mid k \in \mathbb{N}^+\}, D \models \mathcal{Q} \text{ and } \rho_{\mathcal{Q}}(\tau) > v\}. \qquad \square$$

This problem is *NP*-hard for CQs without self-joins [Meliou et al., 2010c], but tractable for *linear* CQs [Meliou et al., 2010c]. Roughly speaking, a CQ is linear if its atoms can be ordered in a way that every variable appears in a continuous sequence of atoms that does not contain a self-join (i.e., a join involving the same predicate), e.g. $\exists x \exists v \exists y \exists u\ (A(x) \wedge S_1(x, v) \wedge S_2(v, y) \wedge R(y, u) \wedge S_3(y, z))$ is linear, but not $\exists x \exists y \exists z\ (A(x) \wedge B(y) \wedge C(z) \wedge W(x, y, z))$, for which RDP is *NP*-complete.[2]

In [Meliou et al., 2010c], the class of CQs for which RDP is tractable is extended to *weakly linear* queries. However, a glitch in proving the tractability result for weakly linear queries has been identified in [Cibele et al., 2016]. In particular, they found that [Meliou et al., 2010c] classifies certain hard queries into tractable cases due to errors in some of the proofs. In addition, they provide a refined characterization of the classes of queries for which, deciding responsibility is tractable, on the basis of a structural property of a query which goes beyond that of linearity.

In this work, we extend the complexity analysis for RDP to CQs (possibly with self-joins), UCQs, Datalog queries. In addition, we study the complexity of the functional, non-decision version of RDP, which is an optimization problem about computing responsibilities.

[2]Computing sizes of minimum contingency sets is reduced to the max-flow/min-cut problem in a network [Meliou et al., 2010c].

We will also define and investigate the decision version of MRC, MRCDP, the *most responsible cause decision problem*. This is a natural problem, because actual causes with the highest responsibility tend to provide most interesting explanations for query answers [Meliou et al., 2010c, Meliou et al., 2010b].

Definition 2.2.3 For a BCQ $\mathcal{Q}$, the *most responsible cause decision problem* is (membership of):

$$\mathcal{MRCDP}(\mathcal{Q}) = \{(D, \tau) \mid \tau \in D^n \text{ and } 0 < \rho_{_D}(\tau) \text{ is a maximum for} D\}. \qquad \square$$

2.2.2 View-conditioned causality

QA-causality is defined for a fixed query $\mathcal{Q}$ and a fixed answer $\bar{a}$. However, in practice one often has multiple queries and/or multiple answers. For a query with several answers one might interested in causes for a a fixed answer, conditioning to the assumed correctness of the other answers to the query. This form of *conditional causality* was suggested in [Meliou et al., 2010b]. The notion was made precise in [Meliou, et al., 2011c], in a more general, non-relational setting, to give an account of the effect of a tuple on multiple outputs (views). Here we adapt this notion of *view-conditioned causality* to the case of a single query, with possibly several answers. We illustrate the usefulness of this notion with a couple of examples.

Example 2.2.3 Consider relations $\mathit{GroupUser}(\mathit{User}, \mathit{Group})$ and $\mathit{GroupFile}(\mathit{File}, \mathit{Group})$, representing users' memberships of groups and access permissions for groups to files, respectively. A user u can access the file f if u belongs to a group that can access f, i.e. there is some *Group* such that $\mathit{GroupUser}(\mathit{User}, \mathit{Group})$ and $\mathit{GroupFile}(\mathit{File}, \mathit{Group})$.

The following Datalog query collects users and the files they can access:

$$\mathit{Access}(\mathit{User}, \mathit{File}) \leftarrow \mathit{GroupUser}(\mathit{User}, \mathit{Group}), \mathit{GroupFile}(\mathit{File}, \mathit{Group}). \qquad (2.1)$$

Suppose we observe that a particular file is accessible by an unauthorized user (an unexpected answer to the query), while all other users' accesses are authorized (i.e., the rest of the answers to the query are deemed to be correct). We want to find out the causes for this unexpected observation. For this task, contingency sets whose removal do not return the correct answers anymore should not be considered.

This example, originally presented in [Cui et al., 2001] and later used in [Buneman et al., 2002, Kimelfeld, 2012b, Kimelfeld, 2012a], is borrowed from the area of view-updates. We use it here to point to the similarities between the seemingly different problems of view-updates and causality. We elaborate on this in Chapter 8. $\square$

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

Definition 2.2.4 Assume $\mathcal{Q}(\bar{x})$ is a monotone query, and $\mathcal{Q}(D) = \{\bar{a}_1, \ldots, \bar{a}_n\}$.

(a) A tuple $\tau \in D^n$ is called a *view-conditioned counterfactual cause* (vcc-cause) for answer $\bar{a}_1$ to $\mathcal{Q}$ if $D \smallsetminus \{\tau\} \not\models \mathcal{Q}(\bar{a}_1)$, but $D \smallsetminus \{\tau\} \models \mathcal{Q}(\bar{a}_i)$, for $i \in \{2, \ldots, n\}$.

(b) A tuple $\tau \in D^n$ is a *view-conditioned actual cause* (vc-cause) for $\bar{a}_1$ if there exists a contingency set, $\Gamma \subseteq D^n$, such τ is a vcc-cause for $\bar{a}_1$ in $D \smallsetminus \Gamma$.

(c) $\mathit{vc\text{-}Causes}(D, \mathcal{Q}(\bar{a}_1))$ denotes the set of all vc-causes for $\bar{a}_1$.

(d) The *vc-causal responsibility* of a tuple τ for answer $\bar{a}_1$, denoted $vc\text{-}\rho_{\mathcal{Q}(\bar{a}_1)}(\tau)$, is $\frac{1}{(|\Gamma|+1)}$, where $|\Gamma|$ is the size of the smallest contingency set that makes τ a vc-cause for $\bar{a}_1$. $\square$

Notice that Definition 2.2.4 could be generalized by considering that several answers are unexpected and the others are correct. This kind of partition can only affect the admissible contingency sets.

Clearly, vc-causes for an answer $\bar{a}$ to a query are also actual causes for the same query, but not necessarily the other way around: $\mathit{vc\text{-}Causes}(D, Q(\bar{a})) \subseteq \mathit{Causes}(D, Q(\bar{a}))$. Notice that the causal responsibility and the vc-causal responsibility of a tuple as a cause, resp. vc-cause, for a query answer may take different values.

Example 2.2.4 (ex. 2.2.3 cont.) The answer $\langle \mathit{John}, \mathit{XML}\rangle$ does not have any vc-cause. In fact, consider for example the tuple *Author(John, TODS)* that is an actual cause for $\langle \mathit{John}, \mathit{XML}\rangle$, with two contingency sets, Γ_1 and Γ_2. It is easy to verify that none of these contingency sets satisfies the condition in Definition 2.2.4, e.g. the original answer $\langle John, CUBE\rangle$ is not preserved in $D \smallsetminus \Gamma_1$. The same argument can be applied to all actual causes for $\langle John, XML\rangle$. □

Example 2.2.5 (ex. 2.2.3 cont.) Consider the instance D with relations $GroupUser$ and $GroupFiles$, and contents as below:

GroupUser	*User*	*Group*
	Joe	g_1
	Joe	g_2
	John	g_1
	Tom	g_2
	Tom	g_3
	John	g_3

GroupFiles	*File*	*Group*
	f_1	g_1
	f_1	g_3
	f_2	g_2
	f_3	g_3

The query $Access$ in (2.1) has the following answers:

$Access(D)$	*User*	*File*
	Joe	f_1
	Joe	f_2
	Tom	f_1
	Tom	f_2
	Tom	f_3
	John	f_1
	John	f_3

Suppose the access of *Joe* to the file, f_1 (corresponding to the query answer $\langle Joe, f_1\rangle$) is deemed to be unauthorized, while all other users accesses are authorized i.e., the rest answers to the query are correct. So that we are interested in causes for this observation.

It can be verified that $GroupUser(Joe, g_1)$ is the only vcc-cause for this answer. □

The notions of vc-causality and vc-responsibility have corresponding decisions problems, which can be defined in terms similar to those for plain causality and responsibility.

Definition 2.2.5 (a) The *vc-causality decision problem* (VCDP) is about membership of

$$\mathcal{VCDP}(\mathcal{Q}) = \{(D, \bar{a}, \tau) \mid \bar{a} \in \mathcal{Q}(D) \text{ and } \tau \in \textit{vc-Causes}(D, \mathcal{Q}(\bar{a}))\ \}.$$

(b) The *vc-causal responsibility decision problem* (VRDP) is about membership of

$$\mathcal{VRDP}(\mathcal{Q}) = \{(D, \bar{a}, \tau, v) \mid \tau \in D^n, v \in \{0\} \cup \{\tfrac{1}{k} \mid k \in \mathbb{N}^+\},\ D \models \mathcal{Q}(\bar{a}) \text{ and } vc\text{-}\rho_{_\mathcal{Q}}(\tau) > v\}. \qquad \square$$

As Example 2.2.4 shows, sometimes there are no vc-causes. For this reason it makes sense to study the complexity of deciding whether a query answer has a vc-cause or not. This is a relevant problem. For illustration, consider the query $Access$ in Example 2.2.3. The existence of a vc-cause for an unexpected answer (unauthenticated access) to this query, tells us that it is possible to revoke the unauthenticated access without restricting other users' access permissions.

Definition 2.2.6 For a monotone query $\mathcal{Q}$, the *vc-cause existence problem* (VCEP) is (deciding about membership of):

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

2.3 Complexity Classes

We recall some complexity classes [Papadimitriou, 1994] used in this work. FP is the class of functional problems associated to decision problem in the class *PTIME*, i.e. that are solvable in polynomial time. P^{NP} (or Δ_2^P) is the class of decision problems solvable in polynomial time by a machine that makes calls to an NP oracle. For $P^{NP(log(n))}$ the number of calls is logarithmic. It is not known if $P^{NP(log(n))}$ is strictly contained in P^{NP}. $FP^{NP(log(n))}$ is similarly defined.

Chapter 3

State of the Art

This research is mainly related to, and unifies ideas from work on causality, degree of causal responsibility, query result explanations, data provenance.

3.1 Causality

Causality is an active research area mainly in philosophy, law and logic with its own dedicated workshops.[1] The idea of analyzing causation in terms of counterfactual dependencies can be traced back to Hume (1739),[2]. An event B counterfactually depends on event A, if A and B both hold, and had A not happened then B would not have happened. The best-known counterfactual theory of causation in modern times is due to [Lewis, 1972]. [Halpern & Pearl, 2005] presents an structural model approach to actual causality, which relies upon a graph structure called a causal model.

The study of causality in data management was explicitly started in [Meliou et al., 2010a, Meliou et al., 2010c, Meliou et al., 2010b], and for query results in relational databases. There, the HP-model is adopted and investigated. This definition is built upon the HP-model, but simplifies it and does not require a causal model. A general overview of causality in a database context is given in [Meliou et al., 2010b]. The definition of query-answer causality is shown to be general enough to be applied to a broad class of database-related applications, e.g. explaining unexpected query-answers, diagnosing network malfunctions, data cleaning, hypothetical reasoning and data provenance [Meliou et al., 2010c, Meliou et al., 2010b, Meliou et al., 2011b, Meliou, et al., 2011c]. In this work we start from this approach, concentrating on causality as defined for relational databases.

[1]E.g. UAI Workshop in Advances in Causal Inference. Amsterdam, Netherlands, 2015. http://www.homepages.ucl.ac.uk/ ucgtrbd//uai2015_causal/

[2]David Hume (1711-1776) was a Scottish philosopher known for applying empirical standards to notions of causation and necessity.http://www.iep.utm.edu/hume-cau/

A comprehensive complexity analysis for computing and deciding causality and responsibility for CQs is given in [Meliou et al., 2010c, Lopatenko & Bertossi, 2007, Cibele et al., 2016]. They show that for this class of queries, causality can be computed in polynomial time in the size of a database.

For the problem of deciding whether the responsibility of a tuple is greater than a threshold, a dichotomy within CQs queries without self-joins was provided in [Meliou et al., 2010c], where the authors showed that this problem is *NP*-hard for CQs without self-joins, but tractable for *weakly linear* CQs (see Section 2.2.1). However, an error in the proof of this dichotomy theorem has been identified in [Cibele et al., 2016]. The authors provide a refined version of the dichotomy, on the basis of a structural property of a query that goes beyond that of linearity.

View-conditioned causality, introduced in [Meliou, et al., 2011c], extends the notion of causality to give an account of the effect of a tuple on multiple outputs (views). The authors also describe a reduction from the computation of vc-causes and their responsibilities to SAT, which enables the use of SAT-solvers to compute causal contributions.

3.2 Causal Responsibility

The notion of causal responsibility in [Meliou et al., 2010c] intends to provide a metric to quantify the causal contribution, as a numerical degree, of a tuple to a query answer. This responsibility-based ranking is considered as one of the most important contributions of HP-causality to data management [Meliou, et al., 2011c]. Causal responsibility can be traced back to [Chockler & Halpern, 2004], where it is suggested that, for variables A and B, the degree of responsibility of A for B should be $\frac{1}{(N+1)}$, and N is the *minimum number of changes* that have to be made to obtain a situation where B counterfactually depends directly on A.

In [Gerstenberg et al., 2010], it has been argued that people use something similar to the intuition behind the notion of degree of responsibility (in the sense of [Chockler & Halpern, 2004]) to ascribe responsibility. However, more recently it has been discussed that people take into account not only the number of changes required to make A a counterfactual cause for B, but also the number of ways to reach a situation where B counterfactually depends on A [Zultan et al., 2013].

In [Halpern, 2015b], it has been argued that, while causal responsibility (in the sense of [Chockler & Halpern, 2004]) could capture some reasonable intuitions, alternative definitions might be more appropriate in some applications.

Perhaps not surprisingly, causal responsibility has been subject to a great deal of research in the law literature [Wright, 1988, Wright, 2001, Hart & Honore, 1959, Moore, 1999, Feinberg, 1968, Braham & Van Hee, 2009]. However, in none of these works the concept given any numerical qualification, except for that of [Braham & Van Hee, 2009]. They introduced the notion of "degree of causality", on the basis of the concept of Necessary Element of a Sufficient Set (NESS) test. Roughly speaking, A is a cause for B according to the NESS test if there exists a set S of events, each of which actually occurred, S is sufficient for B, and A is necessary for the sufficiency of S [Wright, 1985]. In [Halpern, 2008] it has been shown that in many applications, both the NESS test and the HP-model are equivalent (in the sense that they yield same causes). In fact, the authors argue that in order to define a degree of causation, one should focus on the relative frequency with which an action satisfies the NESS test.

In [Braham & Van Hee, 2009], it has been argued that the notion of responsibility as defined in [Chockler & Halpern, 2004] does not determine the share of an action in bringing about an outcome, but only "the extent to which there are other causes". However, no clear distinction between the two approaches has been established.

3.3 Data Provenance

Data provenance aims to explain how a particular result (in an experiment, simulation, query, workflow, etc.) was derived. The provenance of a query on a database is an expression that describes how the answer was derived from the tuples in the database.

Apart from the explicit use of causality, most of the related research on explanations for query results has concentrated on data provenance [Buneman et al., 2001, Buneman & Tan, 2007, Cheney et al., 2009a, Cui et al., 2000, Karvounarakis, 2010, Tannen, 2013]. Causality has been discussed in relation to data provenance [Meliou et al., 2010c, Meliou et al., 2011b] and even workflow provenance [Cheney, 2010]. Specifically, a close connection between QA-causality and why-provenance (in the sense of [Buneman et al., 2001]) has been established in [Meliou et al., 2010c]. Roughly speaking, both definitions concern the same tuples if all tuples in a database are endogenous. However, causality is a more refined notion that can provide reasons and explanations for wrong or surprising results, by ranking provenance based on the notion of responsibility [Meliou et al., 2011b].

More specifically, causal responsibly provides a natural way to extract interesting parts from the provenance. This will be done by discovering causes within the endogenous tuples, and ranking those causes based on the their responsibility.

In [Meliou et al., 2010c], causal responsibility has been applied to lineage of a query answer to provide explanation in terms of highest responsible causes. In [Meliou et al., 2011b], and more recently in [Bergman et al., 2015], responsibility has been evaluated in the context of error tracing and post-factum cleaning. Causal responsibility also has been suggested in the context of data mining provenance in [Glavic et al., 2013]. In particular, they suggest responsibility in clustering algorithms, as a metric to quantify the influence of an input tuple to a particular cluster in the output. The authors argue that ranking the input tuples in terms of their responsibility aid to asses the quality of the clusters.

3.4 Explanation in Databases

3.4.1 Query-answers

In QA-causality, the goal is to compute actual causes for a query result and rank them according to their causal responsibility. However, in the world of big data, the contribution of an individual tuple to a query result might be very little. Therefore, broader explanations might be more appealing, e.g. explanations given in terms of predicates [Roy & Suciu, 2014] (see [Meliou et al., 2014] for a recent survey).

The current trend in explanations in databases aims to provide answers to complex questions on query outputs, typically for aggregate queries. Here, the explanations are formulated as predicates on the input attributes. Roughly speaking, a predicate X is an explanation of one or more outputs Y, if removal of tuples satisfying predicate X also changes Y. That is, similar to QA-causality, explanations are established through innervations. These explanations are further ranked according to a score that measures how much they affect the outputs [Wu & Madden, 2013, Roy & Suciu, 2014].

Following this approach, in [Wu & Madden, 2013], the Scorpion system is proposed which, finds predicates on the input data as explanations for a labeled set of outlier points in an aggregate query over a single relation. In [Roy & Suciu, 2014], a formal framework is proposed for finding explanations to complex SQL queries over database schemas involving multiple relations and foreign key constraints.

Several research projects in databases have aimed at explaining query answers focusing on interesting applications. For instance, [Khoussainova, 2012] proposed the system PerfXplain, which intends to enable users to ask comparative performance-related questions about either pairs of MapReduce jobs or pairs of MapReduce tasks. The problem of "meaningful rating interpretation (MRI)", in the context of collaborative rating sites, has been introduced and investigated in [Das et al., 2011]. In this direction, they develop methods for aggregating user ratings and tags and extracting meaningful demographics patterns such as "Teenage girls consistently like Woody Allen Movies". Finally, the problem of computing top-k influential variables and top-k explanations in probabilistic databases was studied in [Kanagal et al., 2011]. This study intends to answer questions such as "why is the probability of an output tuple greater than another one?".

We should mention that explanations also has been investigated in the context of description logic (in particular, DL-lite) in [Calvanese et al., 2013, Borgida et al., 2008].

3.4.2 Missing query-answers

The problem of explaining missing query answers (also known as why-not provenance) intends to address the following question: why is a certain tuple not in the result set? This problem has been received substantial attention lately e.g., [Huang et al., 2008, Herschel et al., 2009, Chapman & Jagadish, 2009, Tran & Chan, 2010, Kohler et al., 2013, Riddle et al., 2014, Glavic et al., 2015, ten Cate et al., 2015].

In [Huang et al., 2008, Herschel et al., 2009], a form of provenance for potential answers has been presented, on the basis of tuple insertions or modifications that would yield the missing tuples. Alternatively, [Chapman & Jagadish, 2009] focus on the operator in the query plan that eliminated a specific tuple, and [Tran & Chan, 2010] suggests an approach to automatically generate a modified query whose result includes both the original query's results as well as the missing tuple.

In [Meliou et al., 2010c], causality for non-query answers is defined on the basis of sets of *potentially missing tuples* that account for the missing answer. This study was the first to highlight the symmetry between why and why-not provenance. More recently, [Kohler et al., 2013] proposed a provenance games as a novel approach to unify why and why-not provenance. The underlying idea is to view a query evaluation as a game between two players who argue, whether for a given database D and a query $\mathcal{Q}$, a tuple τ is in $\mathcal{Q}(D)$ or not. Actually, the provenance game as developed in [Kohler et al., 2013] is domain dependant. Constraint provenance games introduced in [Riddle et al., 2014] as a domain independent extension of the provenance game.

In another direction, ontologies have been used to provide high-level explanations for query results in [ten Cate et al., 2015, Glavic et al., 2015]. The idea is to summarize the successful or failed derivation of a query result by using ontology languages.

Chapter 4

Thesis Contributions

4.1 Contributions

This thesis make the following specific contributions:

1. For a boolean conjunctive query and its associated denial constraint (the former being the violation view of the latter), we establish a precise connections (in term of mutual characterizations and computational reductions) between actual causes for the query (being true) and the subset- and cardinality-repairs of the instance with respect to the denial constraint. We obtain causes from repairs.

2. As in the setting in 1., we obtain database repairs from causes. For this, we extend the treatment of causality to unions of conjunctive queries, to represent multiple denial constraints. We characterize an actual cause's responsibility in terms of cardinality-repairs. We provide algorithms to compute causes and their (minimal) contingency sets for unions of conjunctive queries. The causes can be computed in *PTIME*.

3. We establish a precise connection between consistency-based diagnosis for a boolean conjunctive query being unexpectedly true according to a system description, and causes for the query being true. In particular, we show how to compute actual causes, their contingency sets, and responsibilities using the diagnosis characterization. Hitting-set-based algorithmic approaches to diagnosis inspire our algorithmic/complexity approaches to causality. We also establish a bidirectional connection between consistency-based diagnosis and database repairs.

4. We reformulate the causality problems as hitting set problems and vertex cover problems on hypergraphs, which allows us to apply results and techniques for the latter to causality.

Profiting from these connections, we obtain new algorithmic and complexity results for causality, which can be summarized as follows:

(a) Checking minimal contingency sets for conjunctive queries can be done in *PTIME*.

(b) The responsibility (decision) problem for conjunctive queries becomes *NP*-complete.

(c) However, the responsibility problem is fixed-parameter tractable when the parameter is the inverse of the responsibility bound.

(d) For conjunctive queries, the functional problem of computing the causes' responsibilities is $FP^{NP(log(n))}$-complete, and deciding most responsible causes is $P^{NP(log(n))}$-complete. These results also hold for unions of conjunctive queries.

5. The structure of the resulting hitting-set problem (as in item 3.) allows us to obtain efficient parameterized algorithms and good approximation algorithms for computing causes and minimal contingency sets.

6. Consistency-based diagnosis decision problems can be unsolvable [Reiter, 1987]. However, there are decidable classes of FO diagnosis specifications, but there is little research on their complexity. In this work, we take advantage of the connection between causality and consistency-based diagnosis and report on some new complexity results for model-based diagnosis (cf. Contribution 6).

7. We define notions of preferred causes; in particular one based on prioritized repairs [Staworko et al., 2012]. We also propose a finer-granularity approach to causality, at the attribute level rather than at the tuple level, that is based on interventions that are repair actions that replace attribute values by null values.

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

9. We characterize and obtain causes in terms of- and from abductive diagnoses. We profit from these connections to obtain new algorithmic and complexity results for causality, which can be summarized as follows:

(a) Deciding tuple-causality for Datalog queries, possibly recursive, is *NP*-complete in data.

(b) A class of Datalog queries is identified for which deciding causality is tractable in combined complexity.

(c) Deciding whether the causal responsibility of a tuple for a Datalog query-answer is greater than a given threshold is *NP*-complete.

10. We establish a precise connections between delete-propagation for conjunctive queries and QA-causality. More precisely, we show that actual causes, most-responsible causes and vc-causes can be obtained from solutions to different variants of the delete-propagation problem, and vice-versa.

11. We profit from the connection between causality and delete-propagation to obtain new complexity results for them; more precisely:

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

(b) Deciding whether an answer has a vc-cause is *NP*-complete.

(c) Deciding if a tuple is a vc-cause is *NP*-complete in data.

(d) Deciding whether the vc-causal responsibility of a tuple for a query answer is greater than a given threshold is *NP*-complete in data.

These results also hold for unions of conjunctive queries.

12. We show, in technical terms, that causal responsibility as introduced in [Meliou et al., 2010c] may only partially fulfil the original intention of plausibly ranking tuples in terms of causal contribution.

We define and investigate an alternative metric, called "*degree of causal contribution*", which, as we show, gives more intuitive and plausible results than causal responsibility.

4.2 Comparison with Related Work

This section makes comparisons between this thesis' contributions and related work. In this regard, we have to start by saying that the area of causality in data management is quite recent. We can safely say that in started with [Meliou et al., 2010a, Meliou et al., 2010c]. For this reason there is not much research directly related to our own work.

1. The complexity analysis for deciding causality and computing causal responsibility in [Meliou et al., 2010c] is restricted to conjunctive queries and conjunctive queries without self-joins, respectively. Our work extends this complexity analysis in the following ways:

(a) We establish the complexity of deciding causality and responsibility for unions of conjunctive queries, and Datalog queries (cf. Contributions 4 and 9 in Section 4.1).

(b) We provide FPT results for deciding causality and responsibility (cf. Contribution 5);

(c) We study the complexity of the optimization problems of computing responsibility (that involves minimality) and most responsible causes, and classify them in the polynomial hierarchy (cf. Contribution 4).

2. In a very recent work, a connection between causality and attribute-based repairs (aka. cell-based repairs) is investigated in [Debosschere et al., 2015].

In contrast to repairs that insert or delete full tuples, attribute-based repairs are obtained by changing attribute values in database tuples. Actually, the authors define and investigate cell-based causes in terms of attribute-based repairs. As a matter of fact, their approach is inspired by the characterization of causes in terms of repairs in our work (cf. Contributions 1 and 2; also [Salimi & Bertossi, 2014, Salimi & Bertossi, 2015a]).

3. Although not in the framework of database repairs, consistency-based diagnosis techniques have been applied to consistency restoration of a database with respect to a set of violated integrity constraints [Gertz, 1996].

4. In this work, we adapt the notion vc-causality as introduced in [Meliou et al., 2010b] to the case of a single query, possibly with several answers. We establish the close connection between this form of vc-causality and the view side-effect deletion-problem. The possibility of the connection between the two was first pointed out, but not established, in [Meliou et al., 2010b, Kimelfeld, 2012a, Kimelfeld, 2012b]. Furthermore, we settle the complexity of deciding vc-causality and responsibility for conjunctive queries (cf. Contribution 11) for which no complexity result was known.

5. The problem of *resilience*, as defined and investigated in [Cibele et al., 2016], is basically the decision version of the problem of computing most responsible causes as defined and investigated in our work (cf. Contribution 4). For the resilience problem, those authors

provide a dichotomy result for the complexity in the case of conjunctive queries without self-joins.

In our work we establish the complexity of the most responsible causes problem for conjunctive queries in general (see [Salimi & Bertossi, 2015a, Salimi & Bertossi, 2015c]). Therefore, the results reported in [Cibele et al., 2016] are complementary to ours. Furthermore, [Cibele et al., 2016] connects the resilience problem to delete-propagation. This connection is subsumed by our results (cf. Contribution 10, see also [Bertossi & Salimi, 2014, Salimi & Bertossi, 2015b]).

6. Causal responsibility, as introduced in [Meliou et al., 2010c], has its origin in [Chockler & Halpern, 2004], where the notion of degree of responsibility has been introduced upon the HP-model for causation.

In [Halpern, 2015b], in has been argued that, while their proposal captures some reasonable intuitions, it is somehow naive; and claims that some other variants might be more appropriate for certain applications.

Our proposed metric for quantification of causal contribution, the degree of causal contribution, is restricted to the context of QA-causality (cf. Contribution 12), as opposed to the general HP-model. Nevertheless, we think our proposal can be extended to the latter, leading to a more refined and general notion of degree of causal contribution.

In our work we make a clear distinction and comparison between our proposal and that of [Meliou et al., 2010c], which has its origin in [Chockler & Halpern, 2004].

7. We observe a close connection between our "degree of causal contribution" and the indices used as measures of degree of causation in [Braham & Van Hee, 2009]. They introduced the notion of "degree of causality", on the basis of the concept of Necessary Element of a Sufficient Set (NESS) test (Cf. Section 3). The NESS test is a widely accepted account for causation in the law literature. However, our justification and motivation for the degree of causal contribution comes from a different perspective, and builds upon (and is applicable to) HP-causality. For a discussion, see Section 9.3.

In [Braham & Van Hee, 2009], it is argued that the notion of responsibility as defined in [Chockler & Halpern, 2004] does not determine the share of an action in bringing about an outcome, but only “the extent to which there are other causes”. However, no clear distinction between the two approaches has been established. In this work, we make a clear distinction between our proposed metric and that of causal responsibility in [Meliou et al., 2010c].

8. The results obtained in this work and previously reported in [Bertossi & Salimi, 2014, Salimi & Bertossi, 2014, Salimi & Bertossi, 2015a, Salimi & Bertossi, 2015b, Salimi & Bertossi, 2015c] suggest that causal reasoning is a central activity in the *reverse data management problems* identified in [Meliou et al., 2011a]. This category has been defined based on the authors’ observation that these problems invert a data transformation process, in order to reason diagnostically back about an observation. Our work is the first to study these problems from the perspective of causality. In fact, we believe that causal reasoning is the most fundamental problem in reverse data management.

More related work is mentioned and described in the coming chapters, mainly in Chapter 10, after presenting the corresponding technical results.

Chapter 5

Causality, Database Repairs and Consistency-Based Diagnosis

In this chapter, we explore the connection between computing causes and their responsibilities for conjunctive query answers, on one hand, and computing repairs in databases with respect to DCs, on the other. We show that these computational problems can be reduced to each other.

Furthermore, we show that inferring and computing actual causes and their responsibilities in a database setting become diagnosis reasoning problems and tasks. Actually, a causality-based explanation for a conjunctive query answer can be viewed as a diagnosis, where in essence the first-order logical reconstruction of the relational database provides the system description [Reiter, 1984], and the observation is the query answer. We also establish a bidirectional connection between diagnosis and repairs.

We provide proofs for all the results, except for those that are rather straightforward.

5.1 Database Repairs

Integrity constraints (ICs) capture the semantics of data and are expected to be satisfied by a database in order to keep its correspondence with the outside reality it is modelling. For several reasons, databases may become inconsistent with respect to a given set of integrity constraints (ICs).

Consistent query answering (CQA) is the problem of computing from a database those answers to a query that are consistent with respect to certain ICs, that the database as a whole may fail to satisfy. Consistent answers have been characterized as those that are invariant under minimal forms of restoration of the database consistency [Arenas et al., 2003a]. The notion of minimal restoration of consistency was captured in [Arenas et al., 2003a] in terms of database repairs, i.e. new consistent database instances that share the schema.

Given a set IC of integrity constraints, a *subset repair* (simply, S-repair) of a possibly inconsistent instance D for schema $\mathcal{S}$ is an instance D' for $\mathcal{S}$ that satisfies IC and makes $\Delta(D, D') = (D \smallsetminus D') \cup (D' \smallsetminus D)$ minimal under set inclusion. $\mathit{Srep}(D, IC)$ denotes the set of S-repairs of D with respect to IC [Arenas et al., 1999]. Similarly, D' is a *cardinality repair* (simply C-repair) of D if D' satisfies IC and minimizes $|\Delta(D, D')|$. $\mathit{Crep}(D, IC)$ denotes the class of C-repairs of D with respect to IC. C-repairs are always S-repairs. For DCs, S-repairs and C-repairs are obtained from the original instance by deleting an S-minimal, resp. C-minimal , set of tuples.[1]

More generally, different *repair semantics* may be considered to restore consistency with respect to general integrity constraints. They depend on the kind of allowed updates on the database (i.e., tuple insertions/deletions, changes of attribute values), and the minimality conditions on repairs, e.g. subset-minimality, cardinality-minimality, etc.

Given D and IC, a *repair semantics*, rSem, defines a class $\mathit{Rep}^{\mathsf{rSem}}(D, IC)$ of rSem-repairs, which are the intended repairs [Bertossi, 2011, sec. 2.5]. All the elements of $\mathit{Rep}^{\mathsf{rSem}}(D, IC)$ are instances over the same schema of D, and consistent with respect to Σ. If D is already consistent, we expect $\mathit{Rep}^{\mathsf{rSem}}(D, IC)$ to contain D as its only member.

Given a repair semantics, rSem, $\bar{c}$ is a rSem-*consistent answer* to an open query $\mathcal{Q}(\bar{x})$ if $D' \models \mathcal{Q}[\bar{c}]$ for every $D' \in \mathit{Rep}^{\mathsf{rSem}}(D, IC)$. A BCQ is rSem-*consistently true* if it is true in every $D' \in \mathit{Rep}^{\mathsf{rSem}}(D, IC)$. In particular, if $\bar{c}$ is a consistent answer to $\mathcal{Q}(\bar{x})$ with respect to S-repairs, we say it is an *S-consistent answer*. Similarly for *C-consistent answers*. Consistent query answering for DCs under S-repairs was investigated in detail [Chomicki & Marcinkowski, 2005]. C-repairs and consistent query answering under them were investigated in detail in [Lopatenko & Bertossi, 2007]. (Cf. [Bertossi, 2011] for more references.)

Example 5.1.1 Consider a database schema $P(X, Y, Z)$ with the functional dependency $X \rightarrow Y$. The inconsistent instance $D = \{P(a, b, c), P(a, c, d), P(a, c, e)\}$, seen as a set of ground atoms, has two S-repairs, $D_1 = P(a, b, c)$ and $D_2 = P(a, c, d), P(a, c, e)$, because the symmetric set differences with D, $\Delta(D, D_1)$ and $\Delta(D, D_2)$, are minimal under set inclusion. However, only for D_2 the cardinality $|(D, D_2)|$ of the symmetric set difference is minimum; and D_2 is the only C-repair.

[1]We will usually say that a set is S-minimal in a class of sets $\mathcal{C}$ if it minimal under set inclusion in $\mathcal{C}$. Similarly, a set is C-minimal if it is minimal in cardinality within $\mathcal{C}$.

The query $P(x, y, z)$ has consistent answers $\langle a, c, d\rangle$ and $\langle a, c, e\rangle$ under the C-repair semantics (they are classic answers in the only C-repair), but none under the S-repair semantics (the two S-repairs share no classic answers). $\square$

5.2 Consistency-Based Diagnosis

Diagnosis is the problem of trying to find what is wrong with some system based on knowledge about the design/structure of the system, possible malfunctions that can occur in the system and observations made of the behaviour of an artifact [Poole, 1988].

A fundamental approach to diagnostic reasoning is called *consistency-based diagnosis* (a form of model-based diagnosis [Struss, 2008, sec. 10.4]) introduced by Reiter [Reiter, 1987]. Starting point of this approach is a description (a model) of a real-world system. Such a model represents the structure of the system; that is, its components and their interrelations. If now the actual behavior of the system conflicts with the expected behavior of the system a diagnostic task has to be performed. This task comprises identifying those components of the system which, when assumed to work abnormally, will account for the difference between expected and the observed behavior.

Formally, consistency-based diagnosis considers problems $\mathcal{M} = (\mathit{SD}, \mathit{COMPS}, \mathit{OBS})$, where SD is the description in logic of the intended properties of a system under the *explicit* assumption that all the *components* in COMPS, are working normally. OBS is a FO sentence that represents the observations. If the system does not behave as expected (as shown by the observations), then the logical theory obtained from $\mathit{SD} \cup \mathit{OBS}$ plus the explicit assumption, say $\bigwedge_{c \in \mathit{COMPS}} \neg Ab(c)$, that the components are indeed behaving normally, becomes inconsistent. Ab is an *abnormality* predicate.[2]

The inconsistency is captured via the *minimal conflict sets* , i.e. those minimal subsets COMPS' of COMPS, such that $\mathit{SD} \cup \mathit{OBS} \cup \{\bigwedge_{c \in \mathit{COMPS}'} \neg Ab(c)\}$ is inconsistent. As expected, different notions of minimality can be used at this point.

A *minimal diagnosis* for $\mathcal{M}$ is a S-minimal Δ of COMPS, such that $\mathit{SD} \cup \mathit{OBS} \cup \{\neg Ab(c) \mid c \in \mathit{COMPS} \smallsetminus \Delta\} \cup \{Ab(c) \mid c \in \Delta\}$ is consistent. That is, consistency is restored by flipping the normality assumption to abnormality for a minimal set of components, and those are the ones considered to be (jointly) faulty. The notion of minimality

[2]Here, and as usual, the atom $Ab(c)$ expresses that component c is (behaving) abnormal(ly).

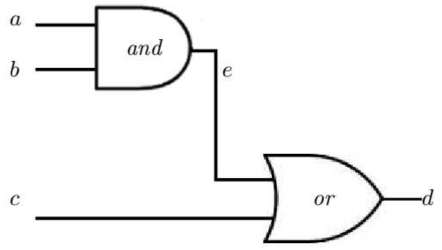

Figure 5.1: A simple circuite with two gates

commonly used is Subset-minimality, i.e. a diagnosis that does not have a proper subset that is a diagnosis. We will use this kind of minimality in relation to diagnosis. Diagnosis can be obtained from conflict sets [Reiter, 1987].

Example 5.2.1 Consider the digital circuit in Figure 5.1. The inputs are $a = 1$, $b = 0$, $c = 1$, but the output is $d = 0$. So, the circuit is not working properly. The diagnosis problem is formulated below as a consistency-based diagnosis problem $\mathcal{M} = (\mathit{SD}, \mathit{COMPS}, \mathit{OBS})$, where $\mathit{COMPS} = \{\mathit{and}, \mathit{or}\}$, SD contains the following FO sentences: $\neg Ab(\mathit{and}) \rightarrow (e \leftrightarrow (a \wedge b))$ and $\neg Ab(\mathit{or}) \rightarrow (d \leftrightarrow (e \vee c))$; and OBS is the FO sentence $a \wedge \neg b \wedge c \wedge \neg d$.

It is easy to verify that $\mathit{SD} \cup \mathit{OBS} \cup \cup \{\neg Ab(\mathit{and}), \neg Ab(\mathit{or})\}$ is inconsistent. $\mathcal{M}$ has two conflict sets: $c_1 = \{\neg Ab(\mathit{and}), \neg Ab(\mathit{or})\}$ and $c_2 = \{\neg Ab(\mathit{or})\}$ from which, only c_2 is S-minimal and forms the single diagnosis of the circuite. □

5.3 Causality and Database Repairs

5.3.1 Actual causes from database repairs

Let $D = D^n \cup D^x$ be an instance for schema $\mathcal{S}$, and $\mathcal{Q} : \exists \bar{x}(P_1(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m))$ a BCQ. $\mathcal{Q}$ may be unexpectedly true, i.e. $D \models \mathcal{Q}$. Now, $\neg\mathcal{Q}$ is logically equivalent to the DC $\kappa(\mathcal{Q})\colon \forall \bar{x} \neg (P_1(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m))$. The requirement that $\neg\mathcal{Q}$ holds can be captured by imposing $\kappa(\mathcal{Q})$ on D. Due to $D \models \mathcal{Q}$, it holds $D \not\models \kappa(\mathcal{Q})$. So, D is inconsistent with respect to $\kappa(\mathcal{Q})$, and could be repaired.

Repairs for (violations of) DCs are obtained by tuple deletions. Intuitively, a tuple that participates in a violation of $\kappa(\mathcal{Q})$ in D is an actual cause for $\mathcal{Q}$. S-minimal sets of tuples like this are expected to correspond to S-repairs for D with respect to $\kappa(\mathcal{Q})$.

More precisely, given an instance $D = D^n \cup D^x$, a BCQ $\mathcal{Q}$, and a tuple $\tau \in D^n$, we consider:

- The class containing the sets of differences between D and those S-repairs that do not contain τ, and are obtained by removing a subset of D^n:

$$\begin{aligned} \mathit{Diff}^s(D, \kappa(\mathcal{Q}), \tau) &= \{D \smallsetminus D' \mid D' \in \mathit{Srep}(D, \kappa(\mathcal{Q})), \\ &\qquad \tau \in (D \smallsetminus D') \subseteq D^n\}. \end{aligned} \tag{5.1}$$

- The class containing the sets of differences between D and those C-repairs that do not contain τ, and are obtained by removing a subset of D^n:

$$\begin{aligned} \mathit{Diff}^c(D, \kappa(\mathcal{Q}), \tau) &= \{D \smallsetminus D' \mid D' \in \mathit{Crep}(D, \kappa(\mathcal{Q})), \\ &\qquad \tau \in (D \smallsetminus D') \subseteq D^n\}. \end{aligned} \tag{5.2}$$

It holds $\mathit{Diff}^c(D, \kappa(\mathcal{Q}), \tau) \subseteq \mathit{Diff}^s(D, \kappa(\mathcal{Q}), \tau)$.

Now, any $\Lambda \in \mathit{Diff}^s(D, \kappa(\mathcal{Q}), \tau)$ can be written as $\Lambda = \Lambda' \cup \{\tau\}$. From the Subset-minimality of S-repairs, it follows that $D \smallsetminus (\Lambda' \cup \{\tau\}) \models \kappa(\mathcal{Q})$, but $D \smallsetminus \Lambda' \models \neg\kappa(\mathcal{Q})$. That is, $D \smallsetminus (\Lambda' \cup \{\tau\}) \not\models \mathcal{Q}$, but $D \smallsetminus \Lambda' \models \mathcal{Q}$. As a consequence, τ is an actual cause for $\mathcal{Q}$ with contingency set Λ'. We have obtained the following result.[3]

Proposition 5.3.1 Given $D = D^n \cup D^x$, and a BCQ $\mathcal{Q}$, $\tau \in D^n$ is an actual cause for $\mathcal{Q}$ iff $\mathit{Diff}^s(D, \kappa(\mathcal{Q}), \tau) \neq \emptyset$. $\square$

Proposition 5.3.2 Given $D = D^n \cup D^x$, a BCQ $\mathcal{Q}$, and $\tau \in D^n$:

(a) If $\mathit{Diff}^s(D, \kappa(\mathcal{Q}), \tau) = \emptyset$, then $\rho_{_D}(\tau) = 0$.

(b) Otherwise, $\rho_{_D}(\tau) = \frac{1}{|\Lambda|}$, where $\Lambda \in \mathit{Diff}^s(D, \kappa(\mathcal{Q}), \tau)$ and there is no $\Lambda' \in \mathit{Diff}^s(D, \kappa(\mathcal{Q}), \tau)$ such that $|\Lambda'| < |\Lambda|$. $\square$

Corollary 5.3.1 Given $D = D^n \cup D^x$ and a BCQ $\mathcal{Q}$: $\tau \in D^n$ is a most responsible actual cause for $\mathcal{Q}$ iff $\mathit{Diff}^c(D, \kappa(\mathcal{Q}), \tau) \neq \emptyset$. $\square$

[3]For database instance D, we will usually denote its subsets with D', Λ, Λ', Δ, Γ, etc.

Example 5.3.1 (ex. 2.2.1 cont.) Consider the same instance D and query $\mathcal{Q}$. In this case, the DC $\kappa(\mathcal{Q})$ is, in Datalog notation, a negative rule: $\leftarrow S(x), R(x,y), S(y)$.

Here, $\mathit{Srep}(D, \kappa(\mathcal{Q})) = \{D_1,\ D_2,\ D_3\}$ and $\mathit{Crep}(D, \kappa(\mathcal{Q})) = \{D_1\}$, with $D_1 = \{R(a_4, a_3),\ R(a_2, a_1),\ R(a_3, a_3),\ S(a_4),\ S(a_2)\}$, $D_2 = \{R(a_2, a_1),\ S(a_4),\ S(a_2),\ S(a_3)\}$, $D_3 = \{R(a_4, a_3),\ R(a_2, a_1),\ S(a_2),\ S(a_3)\}$.

For tuple $R(a_4, a_3)$, $\mathit{Diff}^s(D, \kappa(\mathcal{Q}), R(a_4, a_3)) = \{D \smallsetminus D_2\} = \{\{R(a_4, a_3),\ R(a_3, a_3)\}\}$, which, by Propositions 5.3.1 and 5.3.2, confirms that $R(a_4, a_3)$ is an actual cause, with responsibility $\frac{1}{2}$.

For tuple $S(a_3)$, $\mathit{Diff}^s(D, \kappa(\mathcal{Q}),\ S(a_3)) = \{D \smallsetminus D_1\} = \{S(a_3)\}$. So, $S(a_3)$ is an actual cause with responsibility 1.

Similarly, $R(a_3, a_3)$ is an actual cause with responsibility $\frac{1}{2}$, because $\mathit{Diff}^s(D,\ \kappa(\mathcal{Q}), R(a_3, a_3)) = \{D \smallsetminus D_2,\ D \smallsetminus D_3\} = \{\{R(a_4, a_3),\ R(a_3, a_3)\},\ \{R(a_3, a_3),\ S(a_4)\}\}$.

It holds $\mathit{Diff}^s(D, \kappa(\mathcal{Q}), S(a_2)) = \mathit{Diff}^s(D, \kappa(\mathcal{Q}), R(a_2, a_1)) = \emptyset$, because all repairs contain $S(a_2)$, $R(a_2, a_1)$. This means they do not participate in the violation of $\kappa(\mathcal{Q})$ or contribute to make $\mathcal{Q}$ true. So, they are not actual causes for $\mathcal{Q}$, confirming the result in Example 1.1.1.

$\mathit{Diff}^c(D, \kappa(\mathcal{Q}), S(a_3)) = \{S(a_3)\}$. From Corollary 5.3.1, $S(a_3)$ is the most responsible cause. $\square$

Remark 5.3.1 The results in this section can be easily extended to unions of BCQs. This can be done by associating a DC to each disjunct of the query, and considering the corresponding problems for database repairs with respect to several DCs (Cf. Section 5.3.2.1). $\square$

5.3.2 Database repairs from actual causes

Let us assume in the rest of this section that the database instance is $D = D^n \cup D^x$. We now characterize repairs for inconsistent databases with respect to *a set of* DCs in terms of actual causes, and reduce their computation to computation of causes.[4]

Consider an instance D for schema $\mathcal{S}$, and a set of DCs Σ on $\mathcal{S}$. For each $\kappa \in \Sigma$, say $\kappa\colon \leftarrow A_1(\bar{x}_1), \ldots, A_n(\bar{x}_n)$, consider its associated *violation view* defined by a BCQ,

[4]The characterization results for repairs obtained in this section extend those in [Salimi & Bertossi, 2014] for single DCs.

namely V^κ: $\exists\bar{x}(A_1(\bar{x}_1) \wedge \cdots \wedge A_n(\bar{x}_n))$. The answer *yes* to V^κ shows that κ is violated (i.e., not satisfied) by D.

Next, consider the query that is the union of the individual violation views: $V^\Sigma := \bigvee_{\kappa\in\Sigma} V^\kappa$, a *union of* BCQs (UBCQs). Clearly, D violates (is inconsistent with respect to) Σ iff $D \models V^\Sigma$.

It is easy to verify that D, with $D^x = \emptyset$, is consistent with respect to Σ iff $\mathit{Causes}(D, V^\Sigma) = \emptyset$, i.e. there are no actual causes for V^Σ to be true when all tuples are endogenous.

Now, let us collect all *S-minimal contingency sets* associated with an actual cause τ for V^Σ:

Definition 5.3.1 For $D = D^n \cup D^x$, and Σ a set of DCs:

$$\begin{aligned} \mathit{Cont}(D, V^\Sigma, \tau) \;\; &:= \;\; \{\Gamma \subseteq D^n \mid D \smallsetminus \Gamma \models V^\Sigma,\; D \smallsetminus (\Gamma \cup \{\tau\}) \not\models V^\Sigma, \\ &\qquad \text{and } \forall \Gamma'' \subsetneqq \Gamma,\; D \smallsetminus (\Gamma'' \cup \{\tau\}) \models V^\Sigma\}. \end{aligned} \tag{5.3}$$ $\square$

Notice that for $\Gamma \in \mathit{Cont}(D, V^\Sigma, \tau)$, it holds $\tau \notin \Gamma$. When $D^x = \emptyset$, if $\tau \in \mathit{Causes}(D, V^\Sigma)$ and $\Gamma \in \mathit{Cont}(D, D^n, V^\Sigma, \tau)$, from the definition of actual cause and the S-minimality of Γ, it holds that $\Gamma'' = \Gamma \cup \{\tau\}$ is an S-minimal subset of D with $D \smallsetminus \Gamma'' \not\models V^\Sigma$. So, $D \smallsetminus \Gamma''$ is an S-repair for D. Then, the following holds.

Proposition 5.3.3 For an instance D, with $D^x = \emptyset$, and a set DCs Σ: $D' \subseteq D$ is an S-repair for D with respect to Σ iff, for every $\tau \in D \smallsetminus D'$: $\tau \in \mathit{Causes}(D, V^\Sigma)$ and $D \smallsetminus (D' \cup \{\tau\}) \in \mathit{Cont}(D, V^\Sigma, \tau)$. $\square$

To establish a connection between most responsible actual causes and C-repairs, assume that $D^x = \emptyset$, and collect the *most responsible actual causes* for V^Σ:

Definition 5.3.2 For an instance D with $D^x = \emptyset$:

$$\begin{aligned} \mathit{MRC}(D, V^\Sigma) \;\; &:= \;\; \{\tau \in D \mid \tau \in \mathit{Causes}(D, V^\Sigma),\; \nexists \tau' \in \mathit{Causes}(D, V^\Sigma) \\ &\qquad \text{with } \rho_{_D}(\tau') > \rho_{_D}(\tau)\}. \end{aligned} \tag{5.4}$$ $\square$

Proposition 5.3.4 For instance D, with $D^x = \emptyset$, and set of DCs Σ: $D' \subseteq D$ is a C-repair for D with respect to Σ iff, for every $\tau \in D \smallsetminus D'$: $\tau \in \mathit{MRC}(D, V^\Sigma)$ and $D \smallsetminus (D' \cup \{\tau\}) \in \mathit{Cont}(D, V^\Sigma, \tau)$. $\square$

Actual causes for V^{Σ}, with their contingency sets, account for the violation of some $\kappa \in \Sigma$. Removing those tuples from D should remove the inconsistency. From Propositions 5.3.3 and 5.3.4 we obtain:

Corollary 5.3.2 Given an instance D and a set DCs Σ, the instance obtained from D by removing an actual cause, resp. a most responsible actual cause, for V^{Σ} together with any of its S-minimal, resp. C-minimal, contingency sets forms an S-repair, resp. a C-repair, for D with respect to Σ. $\square$

Example 5.3.2 Consider $D = \{P(a), P(e), Q(a,b), R(a,c)\}$ and $\Sigma = \{\kappa_1, \kappa_2\}$, with $\kappa_1 \colon \leftarrow P(x), Q(x,y)$ and $\kappa_2 \colon \leftarrow P(x), R(x,y)$.

The violation views are $V^{\kappa_1} \colon \exists xy(P(x) \wedge Q(x,y))$ and $V^{\kappa_2} \colon \exists xy(P(x) \wedge R(x,y))$. For $V^{\Sigma} := V^{\kappa_1} \vee V^{\kappa_2}$, $D \models V^{\Sigma}$ and D is inconsistent with respect to Σ.

Now assume all tuples are endogenous. Its holds $\mathit{Causes}(D, V^{\Sigma}) = \{P(a), Q(a,b), R(a,c)\}$, and its elements are associated with sets of S-minimal contingency sets, as follows: $\mathit{Cont}(D, V^{\Sigma}, Q(a,b)) = \{\{R(a,c)\}\}$, $\mathit{Cont}(D, V^{\Sigma}, R(a,c)) = \{\{Q(a,b)\}\}$, and $\mathit{Cont}(D, V^{\Sigma}, P(a)) = \{\emptyset\}$.

From Corollary 5.3.2, and $\mathit{Cont}(D, V^{\Sigma}, R(a,c)) = \{\{Q(a,b)\}\}$, $D_1 = D \smallsetminus (\{R(a,c)\} \cup \{Q(a,b)\}) = \{P(a), P(e)\}$ is an S-repair. So is $D_2 = D \smallsetminus (\{P(a)\} \cup \emptyset) = \{P(e), Q(a,b), R(a,c)\}$. These are the only S-repairs.

Furthermore, $\mathit{MRC}(D, V^{\Sigma}) = \{P(a)\}$. From Corollary 5.3.2, we obtain that D_2 is also a C-repair for D. $\square$

An actual cause τ with any of its S-minimal contingency sets determines a unique S-repair. The last example shows that, with different combinations of a cause and one of its contingency sets, we may obtain the same repair (e.g. for the first two Cont sets). So, we may have more minimal contingency sets than minimal repairs. However, we may still have exponentially many minimal contingency sets, so as we may have exponentially many minimal repairs of an instance with respect to DCs, as the following example shows.[5]

Example 5.3.3 Consider $D = \{R(1,0), R(1,1), \ldots, R(n,0), R(n,1), S(1), S(0)\}$ and the DC $\kappa : \leftarrow R(x,y), R(x,z), S(y), S(z)$. D is inconsistent with respect to κ. There are

[5]Cf. [Arenas et al., 2003b] for an example of the latter that uses key constraints, which are DCs with inequalities (with violation views that contain inequality).

exponentially many S-repairs of D: $D' = D \smallsetminus \{S(0)\}$, $D'' = D \smallsetminus \{S(1)\}$, $D_1 = D \smallsetminus \{R(1,0),\ldots,R(n,0)\}$, ..., $D_{2^n} = D \smallsetminus \{R(1,1),\ldots,R(n,1)\}$. The C-repairs are only D' and D''.

For the BCQ V^κ associated to κ, $D \models V^\kappa$, and $S(1)$ and $S(0)$ are actual causes for V^κ (counterfactual causes with responsibility 1). All tuples in R are actual causes, each with exponentially many S-minimal contingency sets. For example, $R(1,0)$ has the S-minimal contingency set $\{R(2,0),\ldots,R(n,0)\}$, among exponentially many others (any set built with just one element from each of the pairs $\{R(2,0),R(2,1)\}$, ..., $\{R(n,0),R(n,1)\}$ is one). $\square$

5.3.2.1 Causes for unions of conjunctive queries

If we want to compute repairs with respect to sets of DCs from causes for UBCQs using, say Corollary 5.3.2, we first need an algorithm for computing the actual causes and their (minimal) contingency sets for UBCQs. These algorithms could be used as a first stage of the computation of S-repairs and C-repairs with respect to sets of DCs. However, these algorithms, which we develop in Section 5.3.2.2), are also interesting and useful *per se*.

The *PTIME* algorithm for computing actual causes in [Meliou et al., 2010c] is for single conjunctive queries, but does not compute the actual causes' contingency sets. Actually, doing the latter increases the complexity, because *deciding responsibility*[6] of actual causes is NP-hard [Meliou et al., 2010c] (which would be tractable if we could efficiently compute all (minimal) contingency sets).[7] In principle, an algorithm for responsibilities can be used to compute C-minimal contingency sets, by iterating over all candidates, but Example 5.3.3 shows that there can be exponentially many of them.

We first concentrate on the problem of computing actual causes for UBCQs, without their contingency sets, which requires some notation.

Definition 5.3.3 Given $\mathcal{Q} = C_1 \vee \cdots \vee C_k$, each C_i a BCQ, and an instance D:

(a) $\mathfrak{S}(D)$ is the collection of all S-minimal subsets of D that satisfy a disjunct C_i of $\mathcal{Q}$.

[6]For a precise formulation, see Definition 2.2.2.

[7]Actually, [Meliou et al., 2010c] presents a *PTIME* algorithm for computing responsibilities for a restricted class of CQs.

(b) $\mathfrak{S}^n(D)$ consists of the S-minimal subsets Λ of D^n for which there exists a $\Lambda' \in \mathfrak{S}(D)$ with $\Lambda \subseteq \Lambda'$ and $\Lambda \smallsetminus \Lambda' \subseteq D^x$. □

$\mathfrak{S}^n(D)$ contains all S-minimal sets of endogenous tuples that simultaneously (and possibly accompanied by exogenous tuples) make the query true. It is easy to see that $\mathfrak{S}(D)$ and $\mathfrak{S}^n(D)$ can be computed in polynomial time in the size of D.

Now, generalizing a result for CQs in [Meliou et al., 2010c], actual causes for a UBCQs can be computed in *PTIME* in the size of D without computing contingency sets. We formulate this results in terms of the corresponding *causality decision problem* (CDP).

Proposition 5.3.5 Given $D = D^x \cup D^n$, $\tau \in D$, and a UBCQ $\mathcal{Q}$:

(a) τ is an actual cause for $\mathcal{Q}$ iff there is $\Lambda \in \mathfrak{S}^n(D)$ with $\tau \in \Lambda$.

(b) The *causality decision problem* (about membership of)

$$\mathcal{CDP} := \{(D, \tau) \mid \tau \in D^n, \text{ and } \tau \in \mathit{Causes}(D, \mathcal{Q})\}$$

belongs to $PTIME$.

Proof: (a) Assume $\mathfrak{S}(D) = \{\Lambda_1, \ldots, \Lambda_m\}$, and there exists a $\Lambda \in \mathfrak{S}^n(D)$ with $\tau \in \Lambda$. Consider a set $\Gamma \subseteq D^n$ such that, for all $\Lambda_i \in \mathfrak{S}^n(D)$ where $\Lambda_i \neq \Lambda$, $\Gamma \cap \Lambda_i \neq \emptyset$ and $\Gamma \cap \Lambda = \emptyset$. With such a Γ, τ is an actual cause for $\mathcal{Q}$ with contingency set Γ. So, it is good enough to prove that such Γ always exists. In fact, since all subsets of $\mathfrak{S}^n(D)$ are S-minimal , then, for each $\Lambda_i \in \mathfrak{S}^n(D)$ with $\Lambda_i \neq \Lambda$, $\Lambda_i \cap \Lambda = \emptyset$. Therefore, Γ can be obtained from the set of difference between each Λ_i and Λ.

Now, if τ is an actual cause for $\mathcal{Q}$, then there exist an S-minimal $\Gamma \in D^n$, such that $D \smallsetminus (\Gamma \cup \{\tau\}) \not\models \mathcal{Q}$, but $D \smallsetminus \Gamma \models \mathcal{Q}$. This implies that there exists an S-minimal Λ of D, such that $\tau \in \Lambda$ and $\Lambda \models \mathcal{Q}$. Due to the Subset-minimality of Γ, it is easy to see that τ is included in a subset of $\mathfrak{S}^n(D)$.

(b) This is a simple generalization of the proof of the same result for single conjunctive queries found in [Meliou et al., 2010c]. □

Example 5.3.4 (ex. 5.3.2 cont.) Consider the query $\mathcal{Q}\colon\ \exists xy(P(x) \wedge Q(x,y)) \vee \exists xy(P(x) \wedge R(x,y))$, and assume that for D, $D^n = \{P(a), R(a,c)\}$ and $D^x = \{P(e), Q(a,b)\}$. It holds $\mathfrak{S}(D) = \{\{P(a), Q(a,b)\}, \{P(a),\ R(a,c)\}\}$. Since $\{P(a)\} \subseteq \{P(a),\ R(a,c)\}$, $\mathfrak{S}^n(D) = \{\{P(a)\}\}$. So, $P(a)$ is the only actual cause for $\mathcal{Q}$. $\square$

5.3.2.2 Contingency sets for unions of conjunctive queries

It is possible to develop a (naive) algorithm that accepts as input an instance $D = D^n \cup D^x$, and a UBCQ $\mathcal{Q}$, and returns $\mathit{Causes}(D, \mathcal{Q})$; and also, for each $\tau \in \mathit{Causes}(D, \mathcal{Q})$, its (set of) S-minimal contingency sets $\mathit{Cont}(D, \mathcal{Q}, \tau)$.

The basis for the algorithm is a correspondence between the actual causes for $\mathcal{Q}$ with their contingency sets and a *hitting-set problem*.[8] More precisely, for a fixed UBCQ $\mathcal{Q}$, consider the *hitting-set framework*

$$\mathfrak{H}^n(D) = \langle D^n, \mathfrak{S}^n(D)\rangle, \tag{5.5}$$

with $\mathfrak{S}^n(D)$ as in Definition 5.3.3. Different computational and decision problems are based on $\mathfrak{H}^n(D)$, and we will confront some below. Notice that hitting sets (HSs) are all subsets of D^n.

The S-minimal HSs for $\mathfrak{H}^n(D)$ correspond to actual causes with their S-minimal contingency set for $\mathcal{Q}$. Most responsible causes for $\mathcal{Q}$ are in correspondence with HSs for $\mathfrak{H}^n(D)$. This is formalized as follows:

Proposition 5.3.6 For $D = D^x \cup D^n$, $\tau \in D$, and a UBCQ $\mathcal{Q}$:

(a) τ is an actual cause for $\mathcal{Q}$ with S-minimal contingency set Γ iff $\Gamma \cup \{\tau\}$ is an S-minimal HS for $\mathfrak{H}^n(D)$.

(b) τ is a most responsible actual cause for $\mathcal{Q}$ with C-minimal contingency set Γ iff $\Gamma \cup \{\tau\}$ is a minimum HS for $\mathfrak{H}^n(D)$.

Proof: The proof is similar to that of part (a) of Proposition 5.3.5. $\square$

Example 5.3.5 (ex. 5.3.2 and 5.3.4 cont.) D and $\mathcal{Q}$ are as before, but now all tuples are endogenous. Here, $\mathfrak{S}(D) = \mathfrak{S}^n(D) = \{\{P(a), Q(a,b)\}, \{P(a),\ R(a,c)\}\}$. $\mathfrak{H}^n(D)$ has

[8]If $\mathcal{C}$ is a collection of non-empty subsets of a set S, a subset $S' \subseteq S$ is a *hitting set* for $\mathcal{C}$ if, for every $C \in \mathcal{C}$, $C \cap S' \neq \emptyset$. S' is an S-minimal HS if no proper subset of it is also an HS. S is a minimum HS is it has minimum-cardinality.

two S-minimal HSs: $H_1 = \{P(a)\}$ and $H_2 = \{Q(a,b), R(a,c)\}$. Each of them implicitly contains an actual cause (any of its elements) with an S-minimal contingency set (what's left after removing the actual cause). H_1 is also the C-minimal hitting set, and contains the most responsible actual cause, $P(a)$. $\square$

Remark 5.3.2 For $\mathfrak{H}^n(D) = \langle D^n, \mathfrak{S}^n(D)\rangle$, $\mathfrak{S}^n(D)$ can be computed in *PTIME*, and its elements are bounded in size by $|\mathcal{Q}|$, which is the maximum number of atoms in one of $\mathcal{Q}$'s disjuncts. This is a special kind of hitting-set problems. For example, deciding if there is a hitting set of size at most k as been called the *d-hitting-set problem* [Niedermeier, 2003], and d is the bound on the size of the sets in the set class. In our case, d would be $|\mathcal{Q}|$. $\square$

5.3.2.3 Causality, repairs, and consistent answers

Corollary 5.3.2 and Proposition 5.3.6 can be used to compute repairs. If the classes of S- and C-minimal HSs for $\mathfrak{H}^n(D)$ (with $D^n = D$) are available, computing S- and C-repairs will be in *PTIME* in the sizes of those classes. However, it is well known that computing minimal HSs is a complex problem. Actually, as Example 5.3.3 implicitly shows, we can have exponentially many of them in $|D|$; so as exponentially many minimal repairs for D with respect to a denial constraint. We can see that the complexity of contingency sets computation is in line with the complexities of computing hitting sets and repairs.

As Corollary 5.3.2 and Proposition 5.3.6 show, the computation of causes, contingency sets, and most responsible causes via minimal/minimum HS computation can be used to compute repairs and decide about repair questions. Since the HS problems in our case are of the d-hitting set kind, good algorithms and approximations for the latter (cf. Section 5.5.2) could be used in the context of repairs.

In the rest of this section we consider an instance D whose tuples are all endogenous, and a set Σ of DCs. For the disjunctive violation view V^Σ, the following result is obtained from Propositions 5.3.3 and 5.3.4, and Corollary 5.3.2.

Corollary 5.3.3 For an instance D, with $D^x = \emptyset$, and set DCs Σ, it holds:

(a) For every $\tau \in \mathit{Causes}(D, V^\Sigma)$, there is an S-repair that does not contain τ.

(b) For every $\tau \in \mathit{MRC}(D, V^\Sigma)$, there is a C-repair that does not contain τ.

(c) For every $D' \in \mathit{Srep}(D,\Sigma)$ and $D'' \in \mathit{Crep}(D,\Sigma)$, it holds $D \smallsetminus D' \subseteq \mathit{Causes}(D, V^{\Sigma})$ and $D \smallsetminus D'' \subseteq \mathit{MRC}(D, V^{\Sigma})$. □

For a projection-free, and a possibly non-boolean CQ $\mathcal{Q}$, we are interested in its consistent answers from D with respect to Σ. For example, for $\mathcal{Q}(x,y,z)\colon\ R(x,y) \wedge S(y,z)$, the S-consistent (C-consistent) answers would be of the form $\langle a,b,c\rangle$, where $R(a,b)$ and $S(b,c)$ belong to all S-repairs (C-repairs) of D.

From Corollary 5.3.3, $\langle a,b,c\rangle$ is an S-consistent (resp. C-consistent) answer iff $R(a,b)$ and $S(b,c)$ belong to D, but they are not actual causes (resp. most responsible actual causes) for V^{Σ}.

The following simple result and its corollary will be useful in Section 5.5.

Proposition 5.3.7 For an instance D, with $D^x = \emptyset$, a set of DCs Σ, and a projection-free CQ $\mathcal{Q}(\bar{x})\colon\ P_1(\bar{x}_1) \wedge \cdots \wedge P_k(\bar{x}_k)$:

(a) $\bar{c}$ is an S-consistent answer iff, for each i, $P_i(\bar{c}_i) \in (D \smallsetminus \mathit{Causes}(D, V^{\Sigma}))$.

(b) $\bar{c}$ is a C-consistent answer iff, for each i, $P_i(\bar{c}_i) \in (D \smallsetminus \mathit{MRC}(D, V^{\Sigma}))$. □

Example 5.3.6 (ex. 5.3.2 cont.) Consider $\mathcal{Q}(x)\ :\ \ P(x)$. We had $\mathit{Causes}(D, V^{\Sigma}) = \{P(a),\ Q(a,b),\ R(a,c)\}$, $\mathit{MRC}(D, V^{\Sigma}) = \{P(a)\}$. Then, $\langle a\rangle$ is both an S- and a C-consistent answer. □

Notice that Proposition 5.3.7 can easily be extended to conjunctions of ground atomic queries.

Corollary 5.3.4 Given D, a set of DCs Σ, the ground atomic query $\mathcal{Q}$: $P(c)$ is C-consistently true iff $P(c) \in D$ and it is not a most responsible cause for V^{Σ}. □

Example 5.3.7 For $D = \{P(a,b), R(b,c), R(a,d)\}$ and the DC $\kappa : \leftarrow\ P(x,y), R(y,z)$, we obtain: $\mathit{Causes}(D, V^{\kappa}) = \mathit{MRC}(D, V^{\kappa}) = \{P(a,b), R(b,c)\}$.

From Proposition 5.3.7, the ground atomic query $\mathcal{Q}$: $R(a,d)$ is both S- and C-consistently true in D with respect to κ, because, $D \smallsetminus \mathit{Causes}(D, V^{\kappa}) = D \smallsetminus \mathit{MRC}(D, V^{\kappa}) = \{R(a,d)\}$. □

The CQs considered in Proposition 5.3.7 and its Corollary 5.3.4 are not particularly interesting *per se*, but we will use those results to obtain new complexity results for causality later on, e.g. Theorem 5.5.3.

5.4 Consistency-Based Diagnosis: Causes and Repairs

Let $D = D^n \cup D^x$ be an instance for schema $\mathcal{S}$, and $\mathcal{Q}\colon \exists \bar{x}(P_1(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m))$, a BCQ. Assume $\mathcal{Q}$ is, possibly unexpectedly, true in D. So, for the associated DC $\kappa(\mathcal{Q})$: $\forall \bar{x} \neg(P_1(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m))$, $D \not\models \kappa(\mathcal{Q})$. $\mathcal{Q}$ is our *observation*, for which we want to find explanations, using a consistency-based diagnosis approach.

For each predicate $P \in \mathcal{P}$, we introduce predicate Ab_P, with the same arity as P. Intuitively, a tuple in its extension is *abnormal* for P. The "system description", SD, includes, among other elements, the original database, expressed in logical terms, and the DC as true "under normal conditions".

More precisely, we consider the following *diagnosis problem*, $\mathcal{M} = (SD, D^n, \mathcal{Q})$, associated to $\mathcal{Q}$. The FO system description, SD, contains the following elements:

(a) $Th(D)$, which is Reiter's logical reconstruction of D as a FO theory [Reiter, 1984] (cf. Example 5.4.1).

(b) Sentence $\kappa(\mathcal{Q})^{Ab}$, which is $\kappa(\mathcal{Q})$ rewritten as follows:

$$\kappa(\mathcal{Q})^{Ab}\colon\ \forall \bar{x} \neg(P_1(\bar{x}_1) \wedge \neg Ab_{P_1}(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m) \wedge \neg Ab_{P_m}(\bar{x}_m)). \tag{5.6}$$

This formula can be refined by applying the abnormality predicate, Ab, to endogenous tuples only. For this we need to use additional auxiliary predicates End_P, with the same arity of $P \in \mathcal{S}$, which contain the endogenous tuples in P's extension (see Example 5.4.1).

(c) The inclusion dependencies: $\forall \bar{x}(Ab_P(\bar{x}) \rightarrow P(\bar{x}))$, $\forall \bar{x}(End_P(\bar{x}) \rightarrow P(\bar{x}))$, and $\forall \bar{x}(Ab_P(\bar{x}) \rightarrow End_P(\bar{x}))$, for each $P \in \mathcal{P}$.

The last entry, $\mathcal{Q}$, in $\mathcal{M}$ is the "observation", which together with *SD* will produce and inconsistent theory, because we make the initial and explicit assumption that all the abnormality predicates are empty (equivalently, that all tuples are normal), i.e. we consider, for each predicate P, the sentence[9]

$$\forall \bar{x}(Ab_P(\bar{x}) \rightarrow \textbf{false}), \tag{5.7}$$

[9]Notice that these can also be seen as DCs, since they can be written as $\forall \bar{x} \neg Ab_P(\bar{x})$.

where, **false** is a propositional atom that is always false.

The second entry in $\mathcal{M}$ is D^n. This is the set of "components" that we can use to try to restore consistency, in this case, by (minimally) changing the abnormality condition on tuples in D^n. In other words, the rules (5.7) are subject to qualifications: some endogenous tuples may be abnormal. Each diagnosis shows an S-minimal set of endogenous tuples that are abnormal.

Example 5.4.1 (ex. 2.2.1 cont.) For the instance $D = \{S(a_3),\, S(a_4),\, R(a_4, a_3)\}$, with $D^n = \{S(a_4),\, S(a_3)\}$, consider the diagnostic problem $\mathcal{M} = (SD, \{S(a_4), S(a_3)\},\, \mathcal{Q})$, with SD containing:

(a) Predicate completion axioms plus *unique names assumption*:

$$\forall xy(R(x,y) \leftrightarrow x = a_4 \wedge y = a_3), \quad \forall x(S(x) \leftrightarrow x = a_3 \vee x = a_4), \tag{5.8}$$

$$\forall xy(End_R(x,y) \leftrightarrow \textbf{false}), \quad \forall x(End_S(x) \leftrightarrow x = a_3 \vee x = a_4), \tag{5.9}$$

$$a_4 \neq a_3. \tag{5.10}$$

(b) The denial constraint qualified by non-abnormality, $\kappa(\mathcal{Q})^{Ab}$:

$$\begin{aligned}\forall xy\neg \quad &(S(x) \wedge End_S(x) \wedge \neg Ab_S(x) \wedge R(x,y) \wedge End_R(x,y) \wedge \neg Ab_R(x,y)\\ &\wedge S(y) \wedge End_S(y) \wedge \neg Ab_S(y)).\end{aligned} \tag{5.11}$$

In diagnosis formalizations this formula would be usually presented as:

$$\begin{aligned}\forall xy(\quad &\neg Ab_S(x) \wedge \neg Ab_R(x,y) \wedge \neg Ab_S(y) \longrightarrow\\ &\neg(S(x) \wedge End_S(x) \wedge R(x,y) \wedge End_R(x,y) \wedge\ S(y) \wedge End_S(y)).\end{aligned}$$

That is, under the normality assumption, the "system" behaves as intended; in this case, there are no (endogenous) violations of the denial constraint. This main formula in the diagnosis specification can also be written as a disjunctive positive rule:

$$\begin{aligned}\forall xy(S(x) \wedge End_S(x) \wedge R(x,y) \wedge End_R(x,y) \wedge\ S(y) \wedge End_S(y) \longrightarrow&\\ Ab_S(x) \vee Ab_R(x,y) \vee Ab_S(y)).&\end{aligned} \tag{5.12}$$

(c) Abnormality/endogenousity predicates are in correspondence to the database schema, and only endogenous tuples can be abnormal:

$$\forall xy(Ab_R(x,y) \rightarrow R(x,y)), \quad \forall x(Ab_S(x) \rightarrow S(x)), \tag{5.13}$$

$$\forall xy(End_R(x,y) \rightarrow R(x,y)), \quad \forall x(End_S(x) \rightarrow S(x)), \tag{5.14}$$

$$\forall xy(Ab_R(x,y) \rightarrow End_R(x,y)), \ \forall x(Ab_S(x) \rightarrow End_S(x)). \tag{5.15}$$

The assumption is that there are not abnormal tuples:

$$\forall xy(Ab_R(x,y) \rightarrow \textbf{false}), \ \forall x(Ab_S(x) \rightarrow \textbf{false}). \tag{5.16}$$

The FO theory formed by (5.8) - (5.16) is inconsistent. □

Now, in more general terms, the observation is $\mathcal{Q}$ (being true), obtained by evaluating query $\mathcal{Q}$ on (theory of) D. In this case, $D \not\models \kappa(\mathcal{Q})$. Since all the abnormality predicates are assumed to be empty, $\kappa(\mathcal{Q})$ is equivalent to $\kappa(\mathcal{Q})^{Ab}$, which also becomes false wrt D. As a consequence, $SD \cup \{(5.7)\} \cup \{\mathcal{Q}\}$ is an inconsistent FO theory. A diagnosis is a set of endogenous tuples that, by becoming abnormal, restore consistency.

Definition 5.4.1 (a) A *diagnosis* for $\mathcal{M}$ is a $\Delta \subseteq D^n$, such that

$$SD \ \cup \ \{Ab_P(\bar{c}) \mid P(\bar{c}) \in \Delta\} \ \cup \ \{\neg Ab_P(\bar{c}) \mid P(\bar{c}) \in D \smallsetminus \Delta\} \ \cup \ \{\mathcal{Q}\}$$

is consistent.

(b) $Diag^s(\mathcal{M}, \tau)$ denotes the set of S-minimal diagnoses for $\mathcal{M}$ that contain tuple $\tau \in D^n$.

(c) $Diag^c(\mathcal{M}, \tau)$ denotes the set of C-minimal diagnoses in $Diag^s(\mathcal{M}, \tau)$. □

Example 5.4.2 (ex. 5.4.1 cont.) The theory can be made consistent by giving up (5.16), and making S-minimal sets of endogenous tuples abnormal.

$\mathcal{M}$ has two S-minimal diagnosis: $\Delta_1 = \{S(a_3)\}$ and $\Delta_4 = \{S(a_4)\}$. The first one corresponds to replacing the second formula in (5.16) by $\forall x(Ab_S(x) \land x \neq a_3 \rightarrow \textbf{false})$, obtaining now a consistent theory.

Here, $Diag^s(\mathcal{M}, S(a_3)) = Diag^c(\mathcal{M}, S(a_3)) = \{\{S(a_3)\}\}$, and $Diag^s(\mathcal{M}, S(a_4)) = Diag^c(\mathcal{M}, S(a_4)) = \{\{\ S(a_4)\}\}$. □

By definition, $\mathit{Diag}^c(\mathcal{M},\tau) \subseteq \mathit{Diag}^s(\mathcal{M},\tau)$. Diagnoses for $\mathcal{M}$ and actual causes for $\mathcal{Q}$ are related.

Proposition 5.4.1 Consider $D = D^n \cup D^x$, a BCQ $\mathcal{Q}$, and the diagnosis problem $\mathcal{M}$ associated to $\mathcal{Q}$. Tuple $\tau \in D^n$ is an actual cause for $\mathcal{Q}$ iff $\mathit{Diag}^s(\mathcal{M},\tau) \neq \emptyset$. $\square$

The responsibility of an actual cause τ is determined by the cardinality of the diagnoses in $\mathit{Diag}^c(\mathcal{M},\tau)$.

Proposition 5.4.2 For $D = D^n \cup D^x$, a BCQ $\mathcal{Q}$, the associated diagnosis problem $\mathcal{M}$, and a tuple $\tau \in D^n$, it holds:
(a) $\rho_{_D}(\tau) = 0$ iff $\mathit{Diag}^c(\mathcal{M},\tau) = \emptyset$.
(b) Otherwise, $\rho_{_D}(\tau) = \frac{1}{|\Delta|}$, where $\Delta \in \mathit{Diag}^c(\mathcal{M},\tau)$. $\square$

For the proofs of Propositions 5.4.1 and 5.4.2, it is easy to verify that the conflict sets of $\mathcal{M}$ coincide with the sets in $\mathfrak{S}(D^n)$ (cf. Definition 5.3.3). The results are obtained from the characterization of minimal diagnosis as minimal hitting sets of sets of conflict sets (cf. Section 2.1 and [Reiter, 1987]) and Proposition 5.3.6.

Example 5.4.3 (ex. 5.4.2 cont.) From Propositions 5.4.1 and 5.4.2, $S(a_3)$ and $S(a_4)$ are actual cases, with responsibility 1. $\square$

In consistency-based diagnosis, minimal diagnoses can be obtained as S-minimal HSs of the collection of S-minimal *conflict sets* (cf. Section 2.1) [Reiter, 1987]. In our case, conflict sets are S-minimal sets of endogenous tuples that, if not abnormal (only endogenous ones can be abnormal), and together, and possibly in combination with exogenous tuples, make (5.6) false.

It is easy to verify that the conflict sets of $\mathcal{M}$ coincide with the sets in $\mathfrak{S}(D^n)$ (cf. Definition 5.3.3 and Remark 5.3.2). As a consequence, conflict sets for $\mathcal{M}$ can be computed in *PTIME*, the HSs for $\mathcal{M}$ contain actual causes for $\mathcal{Q}$, and the HS problem for the diagnosis problems is of the d-hitting-set kind.

The connection between consistency-based diagnosis and causality allows us, in principle, to apply techniques for the former, e.g. [Feldman et al., 2010, Mozetic & Holzbaur, 1994], to the latter.

Example 5.4.4 (ex. 5.4.1 cont.) The diagnosis problem $\mathcal{M} = (SD, \{S(a_4), S(a_3)\}, \mathcal{Q})$ gives rise to the HS framework $\mathfrak{H}^n(D) = \langle\{S(a_4), S(a_3)\}, \{\{(S(a_3), S(a_4)\}\}\rangle$, with $\{S(a_3), S(a_4)\}$ corresponding to the conflict set $c = \{S(a_4), S(a_3)\}$.

$\mathfrak{H}^n(D)$ has two minimum HSs: $\{S(a_3)\}$ and $\{S(a_4)\}$, which are the S-minimal diagnosis for $\mathcal{M}$. Then, the two tuples are actual causes for $\mathcal{Q}$ (cf. Proposition 5.4.1). From Proposition 5.4.2, $\rho_{_D}(S(a_3)) = \rho_{_D}(S(a_4)) = 1$. □

The solutions to the diagnosis problem can be used for computing repairs.

Proposition 5.4.3 Consider a database instance D with only endogenous tuples, a set of DCs of the form $\kappa\colon\ \forall \bar{x}\neg(P_1(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m)$, and their associated "abnormality-aware" integrity constraints[10] in (5.6) (in this case we do not need End_P atoms).

Each S-minimal diagnosis Δ gives rise to an S-repair of D, namely $D_\Delta = D \smallsetminus \{P(\bar{c}) \in D \mid Ab_P(\bar{c}) \in \Delta\}$; and every S-repair can be obtained in this way. Similarly, for C-repairs using C-minimal diagnoses. □

Example 5.4.5 (ex. 5.4.3 cont.) The instance $D = \{S(a_3),\ S(a_4),\ R(a_4, a_3)\}$ has three (both S- and C-) repairs with respect to the DC $\kappa\colon\ \forall xy\neg(S(x) \wedge R(x, y) \wedge S(y))$, namely $D_1 = \{S(a_3)\}$, $D_2 = \{S(a_4)\}$, and $D_3 = \{R(a_4, a_3)\}$. They can be obtained as $D_{\Delta_1}, D_{\Delta_2},\ D_{\Delta_3}$ from the only (S- and C-) diagnoses, $\Delta_1 = \{S(a_3)\}$, $\Delta_4 = \{S(a_4)\}$, $\Delta_3 = \{R(a_4, a_3)\}$, resp. □

The kind of diagnosis problem we introduced above can be formulated as a *preferred-repair problem* [Bertossi, 2011, sec. 2.5]. For this, it is good enough to materialize tables for the auxiliary predicates Ab_P and End_P, and consider the DCs of the form (5.6) (with the End_P atoms if not all tuples are endogenous), plus the DCs (5.7). The initial extensions for the Ab_P predicates are empty.

If D is inconsistent with respect to this set of DCs, the S-repairs that are obtained by only *inserting* endogenous tuples into the extensions of the Ab_P predicates correspond to subset-minimal diagnosis, and each subset-minimal diagnosis can be obtained in this way.

[10]Notice that these are not denial constraints.

5.5 Complexity Results

5.5.1 Complexity of causality and responsibility

In this section we assume $D = D^n \cup D^x$, without any assumption on D^n, D^x (other than disjointness). We start by analysing a more basic decision problem.

Definition 5.5.1 For a BCQ $\mathcal{Q}$, the *minimal contingency set decision problem* (MCSDP) is about of checking if a set of tuples Γ is an S-minimal contingency set associated to a cause τ (cf. (2.1)):

$\mathcal{MCSDP}(\mathcal{Q}) := \{(D, \tau, \Gamma) \mid \Gamma \in \mathit{Cont}(D, \mathcal{Q}, \tau)\}$. □

Due to the results in Sections 5.3.1 and 5.3.2, it clear that there is a close connection between MCSDP and the *S-repair checking* problem [Bertossi, 2011, chap. 5], about deciding if instance D' is an S-repair of instance D with respect to a set of integrity constraints. Actually, the following result is obtained from the *PTIME* solvability of the S-repair checking problem for DCs [Chomicki & Marcinkowski, 2005] (see also [Afrati & Kolaitis, 2009]).

Proposition 5.5.1 For a BCQ $\mathcal{Q}$, $\mathcal{MCSDP}(\mathcal{Q}) \in \mathit{PTIME}$.

Proof: To decide if $(D, \tau, \Gamma) \in \mathcal{MCSDP}(\mathcal{Q})$, it is good enough to observe, from Proposition 5.3.1, that $(D, \tau, \Gamma) \in \mathcal{MCSDP}(\mathcal{Q})$ iff $D \smallsetminus (\Gamma \cup \{\tau\})$ is an S-repair for D with respect to $\kappa(\mathcal{Q})$. S-repair checking can be done in *PTIME* in data [Chomicki & Marcinkowski, 2005]. □

We could also consider the decision problem defined as in Definition 5.5.1, but with C-minimal Γ. We will not use results about this problem in the following. Furthermore, its connection with the C-repair checking problem is less direct. As one can see from Section 5.3.1, C-minimal contingency sets correspond to a repair semantics somewhere between the S-minimal and C-minimal repair semantics (a subclass of *Srep*, but a superclass of *Crep*): It is about an S-minimal repair with minimum-cardinality that does not contain a particular tuple.

Now we establish that RDP is *NP*-complete for CQs in general. The *NP*-hardness is shown in [Meliou et al., 2010c]. Membership of *NP* is obtained using Proposition 5.5.1.

Theorem 5.5.1 (a) For every BCQ $\mathcal{Q}$, $\mathcal{RDP}(\mathcal{Q}) \in NP$.

(b) [Meliou et al., 2010c] There are CQs $\mathcal{Q}$ for which $\mathcal{RDP}(\mathcal{Q})$ is *NP*-hard.

Proof: (a) We give a non-deterministic *PTIME* algorithm to solve RDP. Non-deterministically guess a subset $\Gamma \subseteq D^n$, return *yes* if $|\Gamma| < \frac{1}{v}$ and $(D, \tau, \Gamma) \in \mathcal{MCSDP}$; otherwise return *no*. According to Proposition 5.5.1 this can be done in *PTIME* in data complexity. $\Box$

In order to better understand the complexity of RP, the responsibility computation problem, we will investigate the *functional*, non-decision version of RDP.

The main source of complexity when computing responsibilities is related to the hitting-set problem associated to $\mathfrak{H}^n(D) = \langle D^n, \mathfrak{S}^n(D)\rangle$ in Remark 5.3.2 (cf. (5.5)). In this case, it is about computing the cardinality of a minimum hitting set that contains a given vertex (tuple) τ. That this is a kind of *d-hitting-set problem* [Niedermeier, 2003] will be useful in Section 5.5.2.

Our responsibility problem can also be seen as a *vertex cover problem* on the *hypergraph*

$$\mathfrak{G}^n(D) = \langle D^n, \mathfrak{S}^n(D)\rangle \tag{5.17}$$

associated to $\mathfrak{H}^n(D) = \langle D^n, \mathfrak{S}^n(D)\rangle$. (That is, the HS framework can be seen as a hypergraph.) In it, the hyperedges are the members of $\mathfrak{S}^n(D)$. Determining the responsibility of a tuple τ becomes the problem on hypergraphs of determining the size of a minimum vertex cover (VC)[11] that contains vertex τ (among all VCs that contain the vertex). Again, in this problem the hyperedges are bounded in size by $|\mathcal{Q}|$.[12]

Example 5.5.1 For $\mathcal{Q}\colon \exists xy(P(x) \wedge R(x,y) \wedge P(y))$, and $D = D^n = \{P(a), P(c), R(a,c), R(a,a)\}$, $\mathfrak{S}(D) = \mathfrak{S}^n(D) = \{\{P(a), R(a,a)\}, \{P(a), P(c), R(a,c)\}\}$.

D is the set of vertices of hypergraph $\mathfrak{G}^n(D)$, whose hyperedges are $\{P(a),\ R(a,a)\}$ and $\{P(a), P(c), R(a,c)\}$. Its minimal VCs are: $vc_1 = \{P(a)\}$, $vc_2 = \{P(c),\ R(a,a)\}$,

[11] A set of vertices is a VC for a hypergraph if it intersects every hyperedge. Obviously, when we talk of *minimum* VC, we are referring to minimal in cardinality.

[12] We recall that repairs of databases with respect to DCs can be characterized as maximal independent sets of *conflict hypergraphs* (conflict graphs in the case of FDs) whose vertices are the database tuples, and hyperedges connect tuples that together violate a DC [Arenas et al., 2003b, Chomicki & Marcinkowski, 2005].

$vc_3 = \{R(a,a), R(a,c)\}$. Then, $P(a)$ is an actual cause with responsibility 1. The other tuples are actual causes with responsibility $\frac{1}{2}$. $\square$

Remark 5.5.1 To simplify the presentation, we will formulate and address our computational problems as problems for graphs (instead of hypergraphs). However, our results still hold for hypergraphs [Lopatenko & Bertossi, 2007]. Actually, the following *representation lemma* holds. $\square$

Lemma 5.5.1 There is a fixed database schema $\mathcal{S}$ and a BCQ $\mathcal{Q} \in L(\mathcal{S})$, without built-ins, such that, for every graph $G = (V, E)$ and $v \in V$, there is an instance D for $\mathcal{S}$ and a tuple $\tau \in D$, such that the size of a minimum VC of G containing v equals the responsibility of τ as an actual cause for $\mathcal{Q}$.

Proof: Consider a graph $G = (V, E)$, and assume the vertices of G are uniquely labeled.

Consider the database schema with relations $\mathit{Ver}(v_0)$ and $\mathit{Edges}(v_1, v_2, e)$, and the conjunctive query $\mathcal{Q}\colon \exists v_1 v_2 e(\mathit{Ver}(v_1) \wedge \mathit{Ver}(v_2) \wedge \mathit{Edges}(v_1, v_2, e))$. Ver stores the vertices of G, and Edges, the labeled edges. For each edge $(v_1, v_2) \in G$, Edges contains n tuples of the form (v_1, v_2, i), where n is the number of vertices in G. All the values in the third attribute of Edges are different, say from 1 to $n \times |E|$. The size of the database instance obtained through this padding of G is still polynomial in size. It is clear that $D \models \mathcal{Q}$.

Assume VC is the minimum vertex cover of G that contains the vertex v. Consider the set of tuples $\Lambda = \{\mathit{Ver}(x) \mid x \in VC\}$. Since $v \in VC$, $\Lambda = \Lambda' \cup \{\mathit{Ver}(v)\}$. Then, $D \smallsetminus (\Lambda' \cup \mathit{Ver}(v)) \not\models Q$. This is because for every tuple $\mathit{Edge}(v_i, v_j, k)$ in the instance, either v_i or v_j belongs to VC. Due to the minimality of VC, $D \smallsetminus \Lambda' \models \mathcal{Q}$.

Therefore, tuple $\mathit{Ver}(v)$ is an actual cause for $\mathcal{Q}$. Suppose Γ is a C-minimal contingency set associated to $\mathit{Ver}(v)$. Due to the C-minimality of Γ, it entirely consists of tuples in Ver. It holds that $D \smallsetminus (\Gamma \cup \{\mathit{Ver}(v')\}) \not\models \mathcal{Q}$ and $D \smallsetminus \Gamma \models \mathcal{Q}$. Consider the set $VC' = \{x \mid \mathit{Ver}(x) \in \Gamma\} \cup \{v'\}$. Since $D \smallsetminus (\Gamma \cup \{\mathit{Ver}(v')\}) \not\models \mathcal{Q}$, for every tuple $\mathit{Edge}(v_i, v_j, k)$ in D, either $v_i \in VC'$ or $v_j \in VC'$. Therefore, VC' is a minimum vertex cover of G that contains v. It holds that $\rho_{D}(\mathit{Ver}(v)) = \frac{1}{1+|\Gamma|}$. So, the size of a minimum vertex cover of G that contains v can be obtained from $\rho_{D}(\mathit{Ver}(v))$. $\square$

Having represented our responsibility problem as a graph-theoretic problem, we first consider a functional computational problems in graphs.

Definition 5.5.2 The *minimal VC membership problem* (MVCMP) consists in, given a graph $G = (V, E)$, an a vertex $v \in V$ as inputs, compute the size of a minimum vertex cover of G that contains v. $\square$

Lemma 5.5.2 Given a graph G and a vertex v in it, there is a graph G' extending G that can be constructed in polynomial time in $|G|$, such that the size of a minimum VC for G that contains v and the size of a minimum VC for G' coincide.

Proof: The size of $VC_G(v)$, the minimum vertex cover of G that contains the vertex v, can be computed from the size of I_G, the maximum independent set of G, that does not contain v. In fact,

$$|VC_G(v)| = |G| - |I_G|. \tag{5.18}$$

Since I is a maximum independent set that does not contain v, it must contain one of the adjacent vertices of v (otherwise, I is not maximum, and v can be added to I). Therefore, $|VC_G(v)|$ can be computed from the size of a maximum independent set I that contains v', one of the adjacent vertices of v.

Given a graph G and a vertex v' in it, a graph G' that extends G can be constructed in polynomial time in the size of G, such that there is a maximum independent set I of G containing v' iff v' belongs to every maximum independent set of G' iff the sizes of maximum independent sets for G and G' differ by one [Lopatenko & Bertossi, 2007, lemma 1]. Actually, the graph G' in this lemma can be obtained by adding a new vertex v'' that is connected only to the neighbors of v'. Its holds:

$$\begin{aligned} |I_G| &= |I'_G| - 1, && (5.19)\\ |I'_G| &= |G'| - |VC_{G'}|, && (5.20) \end{aligned}$$

where $VC_{G'}$ is a minimum vertex cover of G'. From (5.18), (5.19) and (5.20), we obtain: $|VC_G(v)| = |VC_{G'}|$. $\square$

From Lemma 5.5.2 and the $FP^{NP(log(n))}$-completeness of determining the size of a maximum clique in a graph [Krentel, 1988], we obtain:

Proposition 5.5.2 The MVCMP problem for graphs is $FP^{NP(log(n))}$-complete.

Proof: We prove membership by describing an algorithm in $FP^{NP(log(n))}$ for computing the size of the minimum vertex cover of a graph $G = (V, E)$ that contains a vertex $v \in V$. We use Lemma 5.5.2, and build the extended graph G'.

The size of a minimum VC for G' gives the size of the minimum VC of G that contains v. Since computing the maximum cardinality of a clique can be done in time $FP^{NP(log(n))}$ [Krentel, 1988], computing a minimum vertex cover can be done in the same time (just consider the complement graph). Therefore, MVCMP belong to $FP^{NP(log(n))}$.

Hardness can be obtained by a reduction from computing minimum vertex covers in graphs to MVCMP. Given a graph G construct the graph G' as follows: Add a vertex v to G and connect it to all vertices of G. It is easy to see that v belongs to all minimum vertex covers of G'. Furthermore, the sizes of minimum vertex covers for G and G' differ by one. Consequently, the size of a minimum vertex cover of G can be obtained from the size of a minimum vertex cover of G' that contains v. Computing the minimum vertex cover is $FP^{NP(log(n))}$-complete. This follows from the $FP^{NP(log(n))}$-completeness of computing the maximum cardinality of a clique in a graph [Krentel, 1988]. $\square$

From Lemma 5.5.1 and Proposition 5.5.2 we obtain the complexity result for RP. Membership can also be obtained from Theorem 5.5.1.

Theorem 5.5.2 (a) For every BCQ without built-ins, $\mathcal{Q}$, computing the responsibility of a tuple as a cause for $\mathcal{Q}$ is in $FP^{NP(log(n))}$.

(b) There is a database schema and a BCQ $\mathcal{Q}$, without built-ins, such that computing the responsibility of a tuple as a cause for $\mathcal{Q}$ is $FP^{NP(log(n))}$-complete. $\square$

Now we address the most responsible causes problem, MRCDP (cf. Definition 2.2.3). We use the connection with consistent query answering of Section 5.3.2.3, namely Corollary 5.3.4, and the $P^{NP(log(n))}$-completeness of consistent query answering under the C-repair semantics for queries that are conjunctions of ground atoms and a particular DC [Lopatenko & Bertossi, 2007, theo. 4].

Theorem 5.5.3 (a) For every BCQ without built-ins, $\mathcal{MRCDP}(\mathcal{Q}) \in P^{NP(log(n))}$.

(b) There is a database schema and a BCQ $\mathcal{Q}$, without built-ins, for which $\mathcal{MRCDP}(\mathcal{Q})$ is $P^{NP(log(n))}$-complete.

Proof: (a) To show that $\mathcal{MRCDP}(\mathcal{Q})$ belongs to $P^{NP(log(n))}$, consider first the hitting set framework $\mathfrak{H}^n(D) = \langle D^n, \mathfrak{S}^n(D)\rangle$ (cf. Definition 5.3.3 and 5.5) and its associated hypergraph $\mathfrak{G}^n(D)$ (cf. (5.17)).

It holds that τ is a most responsible cause for $\mathcal{Q}$ iff $\mathfrak{H}^n(D)$ has a C-minimal hitting set that contains τ (cf. Proposition 5.3.6). Therefore, τ is a most responsible cause for $\mathcal{Q}$ iff τ belongs to some minimum vertex cover of $\mathfrak{G}^n(D)$.

It is easy to see that $\mathfrak{G}^n(D)$ has a minimum vertex cover that contains τ iff $\mathfrak{G}^n(D)$ has a maximum independent set that does not contains τ. Checking if τ belongs to all maximum independent set of $\mathfrak{G}^n(D)$ can be done in $P^{NP(log(n))}$ [Lopatenko & Bertossi, 2007, lemma 2].

If τ belongs to all independent sets of $\mathfrak{G}^n(D)$, then $(D,\tau) \notin \mathcal{MRCDP}(\mathcal{Q})$; otherwise $(D,\tau) \in \mathcal{MRCDP}(\mathcal{Q})$. As a consequence, the decision can be made in time $P^{NP(log(n))}$.

(b) The proof is by a reduction, via Corollary 5.3.4, from consistent query answering under the C-repair semantics for queries that are conjunctions of ground atoms, which was proved to be $P^{NP(log(n))}$-complete in [Lopatenko & Bertossi, 2007, theo. 4]. Actually, that proof (of hardness) uses a particular database schema $\mathcal{S}$ and a DC κ. In our case, we can use the same schema $\mathcal{S}$ and the violation query V^κ associated to κ (cf. Section 5.3.2).$\square$

From Proposition 5.3.6 and the $FP^{NP(log(n))}$-completeness of determining the size of C-repairs for DCs [Lopatenko & Bertossi, 2007, theo. 3], we obtain the following for the computation of the highest responsibility value.

Proposition 5.5.3 (a) For every BCQ without built-ins, computing the responsibility of the most responsible causes is in $FP^{NP(log(n))}$.

(b) There is a database schema and a BCQ $\mathcal{Q}$, without built-ins, for which computing the responsibility of the most responsible causes is $FP^{NP(log(n))}$-complete.

Proof: (a) To show the membership of $FP^{NP(log(n))}$, consider the hypergraph $\mathfrak{G}^n(D)$ as obtained in Theorem 5.5.3. The responsibility of most responsible causes for $\mathcal{Q}$ can be obtained from the size of the minimum vertex cover of $\mathfrak{G}^n(D)$ (cf. Proposition 5.3.6). The size of the minimum vertex cover in a graph can be computed in $FP^{NP(log(n))}$, which is obtained from the membership of $FP^{NP(log(n))}$ of computing the maximum cardinality of a clique in graph [Krentel, 1988].

It is easy to verify that minimum vertex covers in hyprgraphs can be computed in the same time.

(b) This is by a reduction from the problem of determining the size of C-repairs for DCs shown to be $FP^{NP(log(n))}$-complete in [Lopatenko & Bertossi, 2007, theo. 3]. Actually, that proof (of hardness) uses a particular database schema $\mathcal{S}$ and a DC κ. In our case, we may consider the same schema $\mathcal{S}$ and the violation query V^κ associated to κ (cf. Section 5.3.2).

The size of C-repairs for an inconsistent instance D of the schema $\mathcal{S}$ with respect to κ can be obtained from the responsibility of most responsible causes for V^κ (cf. Corollary 5.3.2). □

5.5.2 FPT of responsibility

We need to cope with the intractability of computing most responsible causes. The area of *fixed parameter tractability* (FPT) [Flum, 2006] provides tools to attack this problem. In this regard, we recall that a decision problem with inputs of the form (I, p), where p is a distinguished parameter of the input, is fixed parameter tractable (or belongs to the class FPT), if it can be solved in time $O(f(|p|) \cdot |I|^c)$, where c and the hidden constant do not depend on $|p|$ or $|I|$, and f does not depend on $|I|$.

In our case, the *parameterized version of the decision problem* $\mathcal{RDP}(\mathcal{Q})$ (cf. Definition 2.2.2) is denoted with $\mathcal{RDP}^p(\mathcal{Q})$, and the distinguished parameter is k, such that $v = \frac{1}{k}$.

That $\mathcal{RDP}^p(\mathcal{Q})$ belongs to FPT can be obtained from its formulation as a d-hitting-set problem (d being the fixed upper bound on the size of the sets in the set class); in this case about deciding if there is a HS that contains the given tuple τ that has cardinality smaller that k. This problem belongs to FPT.

Theorem 5.5.4 For every BCQ $\mathcal{Q}$, $\mathcal{RDP}^p(\mathcal{Q})$ belongs to FPT, where the parameter is the inverse of the responsibility bound.

Proof: First, there is a *PTIME* parameterized algorithm for the d-hitting-set problem about deciding if there is a HS of size at most k that runs in time $O(e^k + n)$, with n the size of the underlying set and $e = d - 1 + o(d^{-1})$ [Niedermeier, 2003]. In our case, $n = |D|$, and $d = |\mathcal{Q}|$ (cf. also [Fernau et al., 2010]).

Now, to decide if the responsibility of a given tuple τ is greater than $v = \frac{1}{k}$, we consider the associated hypergraph $\mathfrak{G}^n(D)$, and we decide if it has a VC that contains τ and whose size is less than k. In order to answer this, we use Lemma 5.5.2, and build the extended hypergraph $\mathfrak{G}'$.

The size of a minimum VC for $\mathfrak{G}'$ gives the size of the minimum VC of $\mathfrak{G}^n(D)$ that contains τ. If $\mathfrak{G}^n(D)$ has a VC that contains τ of size less than k, then $\mathfrak{G}'$ has a VC of size less than k. If $\mathfrak{G}'$ has a VC of size less than k, its minimum size for a VC is less than k. Since this minimum is the same as the size of a minimum VC for $\mathfrak{G}^n(D)$ that contains τ, $\mathfrak{G}^n(D)$ has a VC of size less than k that contains τ. As a consequence, it is good enough to decide if $\mathfrak{G}'$ has a VC of size less than k. For this, we use the HS formulation of this hypergraph problem, and the already mentioned FPT algorithm. $\square$

This result and the corresponding algorithm sketched in its proof show that the higher the required responsibility degree, the lower the computational effort needed to compute the actual causes with at least that level of responsibility. In other terms, parameterized algorithms are effective for computing actual causes with high responsibility or most responsible causes. In general, parameterized algorithms are very effective when the parameter is relatively small [Flum, 2006].

Now, in order to compute most responsible causes, we could apply, for each actual cause τ, the just presented FPT algorithm on the hypergraph $\mathfrak{G}^n(D)$, starting with $k = 1$, i.e. asking if there is VC of size less than 1 that contains τ. If the algorithm returns a positive result, then τ is a counterfactual cause, and has responsibility 1. Otherwise, the algorithm will be launched with $k = 2, 3, \ldots, |D^n|$, until a positive result is returned. (The procedure can be improved through binary search on $k = 1, 2, 3, \ldots, m$, with m possibly much smaller than $|D|$.)

The complexity results and algorithms provided in this section can be extend to UBCQs. This is due to Remark 5.3.1 and the construction of $\mathfrak{G}^n(D)$, which the results in this section build upon.

For the d-hitting-set problem there are also efficient parameterized approximation algorithms [Brankovic & Fernau, 2012]. They could be used to approximate the responsibility problem. Furthermore, approximation algorithms developed for the minimum VC problem on bounded hypergraphs [Halperin, 2002, Okun, 2005] should be applicable to approximate most responsible causes for query answers. Via the causality/repair connection (cf. Section 5.3.2.3), it should be possible to develop approximation algorithms to compute S-repairs of particular sizes, C-repairs, and consistent query answers with respect to DCs.

5.5.3 Some complexity results for model-based diagnosis

It is known that consistency-based diagnosis decision problems can be unsolvable [Reiter, 1987]. However, there are decidable classes of FO diagnosis specifications, and those classes are amenable to complexity analysis. However, there is little research on the complexity analysis of solvable classes of consistency-based diagnosis problems. The connection we established in the previous sections between causality, repairs and consistency-based diagnosis can be used to obtain new algorithmic and complexity results for the latter. Without trying to be exhaustive about this, which is beyond the scope of this work, we give an example of the kind of results that can be obtained.

Considering the diagnosis problem we obtained in Section 5.4, we can define a class of diagnosis problems. Cf. Example 5.4.1, in particular (5.12), for motivation.

Definition 5.5.3 A *disjunctive positive* (DP) diagnosis specification Σ is a consistent FO logical theory, such that:

(a) Σ has a signature (schema) consisting of a finite set of constants, a set of predicates $\mathcal{S}$, a set $\mathcal{S}^{ab}$ of predicates of the form Ab_R,[13] with $R \in \mathcal{S}$, and Ab_R with the same arity of R. $\mathcal{S}$ and $\mathcal{S}^{ab}$ are mutually disjoint.

(b) Σ is inconsistent with $AB^{\mathcal{S}} := \{\forall \bar{x}(Ab_R(\bar{x}) \rightarrow \textbf{false}) \mid R \in \mathcal{S}\}$.

(c) Consists of:

[13] Or any other "abducible" predicates that are different from those in $\mathcal{S}$.

(c1) Sentences of the form $\forall \bar{x}(C(\bar{x}) \longrightarrow \bigvee_i Ab_{R_i}(\bar{x}_i))$, with $\bar{x}_i \subseteq \bar{x}$, and $C(\bar{x})$ a conjunction of atoms that does not include Ab-atoms of any kind.

(c2) Sentences of the forms $\forall \bar{x}(Ab_R(\bar{x}) \longrightarrow (R(\bar{x}) \wedge S(\bar{x})))$, with $S \in \mathcal{S}$.

(c3) A finite background universal theory $\mathcal{T}$ expressed in terms of predicates in $\mathcal{S}$ (and constants) that has a unique Herbrand model.[14] □

As above, a diagnosis is a set of Ab_R-atoms that, when assumed to be true, restores the consistency of the correspondingly modified $\Sigma \cup AB^{\mathcal{S}}$.

There are at least two important computational tasks that emerge, namely, given a *disjunctive positive* (DP) diagnosis specification Σ together with $AB^{\mathcal{S}}$:

1. The *minimum-cardinality diagnosis* (MCD) problem, about computing minimum-cardinality diagnoses.

2. The *minimal membership diagnosis*, (MMD) about computing minimum-cardinality diagnoses that contain a given *Ab*-atom.

It is not difficult to see that these problems are computable (or solvable in their decision versions). Now we can obtain complexity lower bounds for them. Actually, in Section 5.4, the *responsibility* and *most responsible causes problem* were reduced to diagnosis problems for specifications that turned out to be disjunctive positive (see (5.12)).

More specifically, Proposition 5.4.2 reduces computing responsibility of a tuple to computing the size of a minimum-cardinality diagnosis that contains the tuple. Furthermore, as a simple corollary of Proposition 5.4.2, we obtain the computation of minimum-cardinality diagnoses allows us to compute most responsible causes. Now, combining all this with Proposition 5.5.3 and Theorem 5.5.2, we obtain the following lower bounds for our diagnosis problems.

Theorem 5.5.5 For disjunctive positive diagnosis specifications, the MCD and MMD problems are $FP^{NP(log(n))}$-hard in the size of their underlying Herbrand structure. □

[14]This condition is clearly satisfied by the logical reconstruction of a relational database, but can be relaxed in several ways.

Chapter 6

Preferred Causes from Preferred Repair

In QA-causality, the exogenous/endogenous partition could capture simple forms of preferences that can be expressed in terms of restricting causes to be among certain tuples or tables. In this direction, a tuple is either exogenous or endogenous. However, some problems would benefit from more complex forms of preferences. We can think of a priority relation among tuples, in such a way that, for example, we prioritize -as causes- tuples in a given relation R, and we are not interested in tuples in another relation S. So, the user can specify a priority relation between the two relations, or different *scores* for these relations [Meliou et al., 2011b].

In Section 5.3.1 we characterized causes and most responsible causes in terms of S-repairs and C-repairs, resp. We could generalize the notion of a cause and/or its responsibility by using, in principle, any *repair semantics* rSem. The latter is represented by a class of repairs $Rep^{\mathsf{rSem}}(D, \Sigma)$, of D with respect to a set of denial constraints. This class contains consistent instances over the same schema as D, and satisfy additional conditions. Actually, a repair semantics is based on two elements that determine $Rep^{\mathsf{rSem}}(D, \Sigma)$: (a) the class of admissible "repair actions" (updates to restore consistency), and (b) a form of minimality condition that forces (minimal) repairs to stay as close as possible to the original instance D [Bertossi, 2011, sec. 2.5].

When dealing with (sets of) denial constraints, the repair actions can only be of certain kinds. Usually tuple deletions have been considered. This is the case of the S- and C-repairs we have considered in this work so far. We could go beyond and consider the notion of *prioritized repair* [Staworko et al., 2012]. Also changes of attribute values can be the chosen repair actions, including the use of *null values*, to "destroy" joins (again, with different semantics, e.g. with nulls *à la* SQL [Bravo et al., 2006, Bertossi & Li, 2013]).

In this chapter we explore the possibility of introducing a notion of *preferred cause* that is based on a given repair semantics. This idea is inspired by (and generalizes) the characterization of causes in terms of repairs that we obtained before, namely (5.1), (5.2), Proposition 5.3.1, and Corollary 5.3.1.

If we define causes and their (minimal) contingency sets on the basis of a given repair semantics, the minimality condition involved in the latter will have an impact on the notion of minimal (or preferred) contingency set, and indirectly, on the notions of responsibility and most responsible cause. We could say that the efforts in [Halpern, 2014, Halpern, 2015a] to modify the original definition of HP-causality are about considering more appropriate restrictions on contingencies. Since in some cases the original HP-model does not provide intuitive results regarding causality, the modifications avoid this by recognizing some contingencies as "unreasonable" or "farfetched".

6.1 Prioritized Repairs

The prioritized repairs in [Staworko et al., 2012] are based on a *priority relation*, $\succ$, on the set of database tuples. In the case of a pair of (mutually) *conflicting tuples*, i.e. that simultaneously violate a constraint in a given set set of DCs (possibly in company of other tuples), the repair process reflects the user preference -as captured by the priority relation- on the tuples that are privileged to be kept in the database, i.e. in the intended repairs.

Given such a priority relation, in [Staworko et al., 2012] different classes of prioritized repairs are introduced, namely the class of *globally optimal repairs*, that of *Pareto-optimal repairs*, and that of *completion-optimal repairs*. Intuitively, each class relies on a different *optimality criterion* that is used to extend the priority relation $\succ$ on pairs of conflicting facts to a priority relation on the set of S-repairs. As a consequence, each of these three classes is contained in that of the S-repairs. In particular, all these repairs are based on tuple deletions.

Let us denote with $\mathit{Rep}^{\succ,\mathcal{X}}(D,\Sigma)$ the class of all prioritized repairs based on $\succ$ and the optimality criterion $\mathcal{X}$. Its elements are called $(\succ,\mathcal{X})$-*prioritized repairs* of D with respect to a the set Σ of denial constraints. It holds $\mathit{Rep}^{\succ,\mathcal{X}}(D,\Sigma) \subseteq \mathit{Srep}(D,\Sigma)$, and then, all the elements of $\mathit{Rep}^{\succ,\mathcal{X}}(D,\Sigma)$ are subsets of D.

In order to show a concrete class $\mathit{Rep}^{\succ,\mathcal{X}}(D,\Sigma)$, we first recall the definitions of priority relation and *global-optimal repair* from [Staworko et al., 2012].

Definition 6.1.1 Given an instance D and a set of denial constraints Σ , a binary relation $\succ$ on D is a *priority relation* with respect to Σ if: (a) $\succ$ is acyclic, and (b) for every $\tau, \tau' \in D$, if $\tau \succ \tau'$, then τ and τ' are mutually conflicting.[1] $\square$

Definition 6.1.2 Let D be an instance, Σ a set of DCs, and $\succ$ a corresponding priority relation. Let D' and D'' be two consistent sub-instances of D. D' is a *global improvement* of D'' if $D' \neq D''$, and for every tuple $\tau' \in D'' \smallsetminus D'$, there exists a tuple $\tau \in D' \smallsetminus D''$ such that $\tau \succ \tau'$. D' is a *global-optimal repair* of D, if D' is an S-repair and does not have a global improvement. $\square$

In this definition, the optimality criterion, a possible $\mathcal{X}$ above, is that of global-optimal repair, or $(\succ, go)$-repair, which leads to a class $\mathit{Rep}^{\succ,go}(D,\Sigma)$. We consider this repair semantics just for illustration purposes.

Example 6.1.1 (ex. 1.1.1 cont.) Assume the instance is subject to the following denial constraint:

$$\begin{aligned}\kappa\colon \quad & \forall\, JName,\ \forall\, \#Paper\ \ \neg(Author(AuName, JName) \wedge \\ & \qquad Journal(JName, Topic, \#Paper) \wedge AuName = \mathsf{John} \wedge Topic = \mathsf{XML}),\end{aligned} \tag{6.1}$$

capturing the condition that "John does not have a journal paper on XML".

D is inconsistent with respect to κ, and contains the following sets of conflicting tuples:

$$\begin{aligned} C_1 &= \{Author(John,\ TKDE), Journal(TKDE, XML, 30)\},\\ C_2 &= \{Author(John,\ TODS), Journal(TODS, XML, 32)\}.\end{aligned}$$

D has the following S-repairs, each obtained by deleting one tuple from each of C_1 and C_2,

[1]We should say $\{\tau, \tau'\}$ is a *conflict*, i.e. the two tuples jointly participate in the violation of one of the DCs in Σ.

to resolve the conflicts:

$$\begin{aligned}
D_1 &= \{\mathit{Author}(\mathit{TOM}, \mathit{XML}), \mathit{Journal}(\mathit{TKDE}, \mathit{CUBE}, \mathit{31}), \mathit{Author}(\mathit{JOHN}, \mathit{TODS}), \\
&\quad \mathit{Journal}(\mathit{TKDE}, \mathit{XML}, \mathit{30})\} \\
D_2 &= \{\mathit{Author}(\mathit{TOM}, \mathit{XML}), \mathit{Journal}(\mathit{TKDE}, \mathit{CUBE}, \mathit{31}), \mathit{Journal}(\mathit{TKDE}, \mathit{XML}, \mathit{30}), \\
&\quad \mathit{Journal}(\mathit{TODS}, \mathit{XML}, \mathit{32})\} \\
D_3 &= \{\mathit{Author}(\mathit{TOM}, \mathit{XML}), \mathit{Journal}(\mathit{TKDE}, \mathit{CUBE}, \mathit{31}), \mathit{Author}(\mathit{JOHN}, \mathit{TKDE}), \\
&\quad \mathit{Journal}(\mathit{TODS}, \mathit{XML}, \mathit{32})\} \\
D_4 &= \{\mathit{Author}(\mathit{TOM}, \mathit{XML}), \mathit{Journal}(\mathit{TKDE}, \mathit{CUBE}, \mathit{31}), \mathit{Author}(\mathit{JOHN}, \mathit{TKDE}), \\
&\quad \mathit{Author}(\mathit{JOHN}, \mathit{TODS})\}
\end{aligned}$$

(a) Now, assume a user prefers to resolve a conflict by removing tuples from the *Author* table rather than the *Journal* table, maybe because he considers the latter more reliable than the former. This can be expressed with the following priority relationships on conflicting tuples: *Journal(TKDE, XML, 30)* $\succ$ *Author(John, TKDE)* and *Journal(TODS, XML, 32)* $\succ$ *Author(John, TODS)*.

In this case only D_2 is a global-optimal repair. Actually, D_2 is a global improvement over each of D_1, D_3 and D_4. For example, if we consider D_1, then $D_2 \setminus D_1 = \{$ *Author(JOHN, TODS), Journal(TKDE, XML, 30)*$\}$ and $D_1 \setminus D_2 = \{$ *Author(JOHN, TODS), Journal(TKDE, XML, 30)*$\}$. We can see that, for each tuple in $D_2 \setminus D_1$, there is a tuple in $D_1 \setminus D_2$ that has a higher priority. Therefore, D_2 is a global improvement on D_1. So, in this case $\mathit{Rep}^{\succ,go}(D,\kappa) = \{D_2\}$

In this case, the uniqueness of the global-optimal repair is quite natural as the preference relation among conflicting tuples is a total relation. So, we know how to resolve every conflict according to the user preferences.

(b) For a more subtle situation, assume the user has the priorities as before, but in addition he tends to believe that John has a paper in *TODS*. In this case we have only the relationship *Journal(TKDE, XML, 30)* $\succ'$ *Author(John, TKDE)*, and no preference for resolving the second conflict. Now both D_1 and D_2 are global-optimal repairs. That is, now $\mathit{Rep}^{\succ',go}(D,\kappa) = \{D_1, D_2\}$. □

6.2 Preferred Causes from Prioritized Repairs

According to the motivation provided at the beginning of this chapter, we now define *preferred causes* on the basis of a class of prioritized repairs. (Compare (6.2) below with (5.1) and (5.2).) To keep things simple, we concentrate on single BCQs, $\mathcal{Q}$, whose associated denial constraints are denoted by $\kappa(\mathcal{Q})$.

In Section 5.3.2.2 actual causes and their minimal contingency sets for a UBCQ were characterized as the minimal hitting sets of the collection $\mathcal{C}$ of minimal subsets of a database that entail the query. Those minimal hitting sets are obtained by removing at least one tuple from each of the elements of $\mathcal{C}$ (cf. Proposition 5.3.6). At this point, user preferences, or priorities, could be applied to tuples that belong to a same set $\mathcal{C}$.

Definition 6.2.1 Given an instance D and a BCQ $\mathcal{Q}$, tuples τ and τ' are *jointly-contributing* if $\tau \neq \tau'$, and there exists an S-minimal $\Lambda \subseteq D$ such that $\Lambda \models \mathcal{Q}$ and $\tau, \tau' \in \Lambda$. □

Now we define priority relations on jointly-contributing tuples.

Definition 6.2.2 Given an instance D and a BCQ $\mathcal{Q}$, a binary relation $\succ_c$ on D is a *causal priority relation* with respect to $\mathcal{Q}$ if: (a) $\succ_c$ is acyclic, and (b) for every $\tau, \tau' \in D$, if $\tau \succ_c \tau'$, then τ and τ' are jointly-contributing tuples. □

This definition introduces a natural notion of preference on causality. Actually, this way of approaching priorities on causes is in (inverse) correspondence with preference on repairs as based on priority relations on conflicting tuples. To see this, first observe that for a given instance D and BCQ $\mathcal{Q}$: τ and τ' are jointly-contributing tuples for $\mathcal{Q}$ iff τ and τ' are mutually conflicting tuples for $\kappa(\mathcal{Q})$.

Next, in the context of prioritized repairs, a priority relation reflects a user preference on tuples that are preferred to be kept in the database. This is the inverse of causality, where a causal priority relation, as we defined it, reflects the tuples that are preferred to be (hypothetically or counterfactually) removed from database, to make them preferred causes.

In the following assume $\succ_c^r$ is the inverse of a causal priority relation $\succ_c$. That is, $\tau \succ_c^r \tau'$ iff $\tau' \succ_c \tau$. Clearly, $\succ_c^r$ is acyclic, and can be imposed, with the expected result, on pairs of conflicting tuples. As a consequence, $\succ_c^r$ can be used to define prioritized repairs.

Definition 6.2.3 Let D be an instance, $\mathcal{Q}$ a BCQ, τ a tuple in D, $\succ_c$ a causal priority relation on D's tuples.

(a) $\mathit{Diff}^{\succ_c^r,\mathcal{X}}(D,\kappa(\mathcal{Q}),\tau) := \{D \smallsetminus D' \mid D' \in \mathit{Rep}^{\succ_c^r,\mathcal{X}}(D,\kappa(\mathcal{Q})), \text{and } \tau \in D \smallsetminus D'\}$. (6.2)

(b) $\tau \in D$ is a $(\succ_c, \mathcal{X})$-*preferred cause* for $\mathcal{Q}$ iff $\mathit{Diff}^{\succ_c^r,\mathcal{X}}(D,\kappa(\mathcal{Q}),\tau) \neq \emptyset$. □

Notice that every $(\succ_c, \mathcal{X})$-preferred cause is also an actual cause. This follows from Proposition 5.3.1 and the fact that prioritized repairs are also S-repairs.

Similarly to Proposition 5.3.2, for each $\Lambda \in \mathit{Diff}^{\succ_c^r,\mathcal{X}}(D,\kappa(\mathcal{Q}),\tau)$, it holds that $\tau \in \Lambda$, τ is a $(\succ_c, \mathcal{X})$-preferred cause, and also an actual cause for $\mathcal{Q}$ with S-minimal contingency set $\Lambda \smallsetminus \{\tau\}$. In particular, τ's responsibility can be defined and computed as before, but now restricting its contingency sets to those of the form $\Lambda \smallsetminus \{\tau\}$, with $\Lambda \in \mathit{Diff}^{\succ_c^r,\mathcal{X}}(D,\kappa(\mathcal{Q}),\tau)$. In this way, a causal priority relation may affect the responsibility of a cause (with respect to the non-prioritized case).

Notation: $\mathit{Cont}^{\succ_c,\mathcal{X}}(D,\mathcal{Q},\tau) := \{\Lambda \smallsetminus \{\tau\} \mid \Lambda \in \mathit{Diff}^{\succ_c^r,\mathcal{X}}(D,\kappa(\mathcal{Q}),\tau)\}$ is the class of all S-minimal contingency sets for a $(\succ_c, \mathcal{X})$-preferred cause τ.

Example 6.2.1 (ex. 6.1.1 cont.) The following BCQ query $\mathcal{Q}$ is true in D:

$$\exists\, \mathit{Jname}, \exists\, \#\mathit{Paper}\ \ (\mathit{Author}(\mathsf{John}, \mathit{Jname}) \wedge \mathit{Journal}(\mathit{Jname}, \mathsf{XML}, \#\mathit{Paper}));$$

and its associated DC $\kappa(\mathcal{Q})$ is κ in (6.1).

We want to obtain the preferred causes for $\mathcal{Q}$ being, possibly unexpectedly, true in D, with the following preferences: (a) We prefer those among the *Author* tuples. (b) It is likely that John does have a paper in TODS. So, we prefer *Author(John, TODS)* not to be the cause.

These causal priorities are in inverse correspondence with those in the second case of Example 6.1.1(b) about priorities for repairs. That is, for our causal priority relation $\succ_c$ here, its inverse $\succ_c^r$ is $\succ'$ in Example 6.1.1(b). There we had $\mathit{Rep}^{\succ',go}(D,\kappa(Q)) = \{D_1, D_2\}$, which we can use to apply Definition 6.2.3.

We obtain as the globally-optimal causes, i.e. as $(\succ_c, go)$-causes: *Author(John, TKDE)*, *Author(TODS, XML, 32)* and *Author(John, TODS)*, all with the same responsibility, $\frac{1}{2}$. □

Notice that Definition 6.2.3 can be easily extended to UBCQs. This is done, as earlier In this work, by considering the set Σ of denial constraints associated to a UBCQ. In the other direction, we recall that if we start with a set of DCs Σ, the corresponding UBCQ is denoted with V^{Σ}.

As we did in the previous sections of this work, we could take advantage of algorithmic and complexity results about prioritized repairs [Staworko et al., 2012, Fagin et al., 2015], to obtain complexity results for preferred causes problems. As an example, we establish the complexity of the minimal contingency set decision problem for $(\succ_c, go)$-*preferred causes*. More precisely, for an instance D and a UBCQ $\mathcal{Q}$, the *minimal preference-contingency set* (decision) problem is about deciding if a set of tuples Γ is an S-minimal contingency set associated to a $(\succ_c, go)$-preferred cause τ.

Definition 6.2.4 For a UBCQ $\mathcal{Q}$, the *minimal preference-contingency set* decision problem is about membership of:

$$\mathcal{MPCDP}(\mathcal{Q}) := \{(D, \succ_c, \tau, \Gamma) \mid \tau \in D, \Gamma \subseteq D, \text{ and } \Gamma \in \mathit{Cont}^{\succ_c, go}(D, \mathcal{Q}, \tau)\}. \qquad \square$$

From Definition 6.2.3, there is a close connection between $\mathcal{MPCDP}$ and the *global-optimal repair checking problem*, i.e. about deciding if an instance D' is a $(\succ, go)$-repair of D with respect to a set of denial constraints. If we accept functional dependencies (FDs) among our denial constraints (and then, UBCQs that involve inequalities), the following result can be obtained from the *NP*-completeness of globally-optimal repair checking [Staworko et al., 2012] for FDs.

Proposition 6.2.1 For a UBCQ $\mathcal{Q}$ with inequalities, $\mathcal{MPCDP}(\mathcal{Q})$ is *NP*-hard.

Proof: It is good enough to reduce globally-optimal repair checking to our contingency checking problem. So, consider an inconsistent instance D with respect to a set of denial constraint Σ, a priority relation for repairs $\succ$, and $D' \subseteq D$. To check if $D' \in \mathit{Rep}^{\succ, go}(D, \Sigma)$ we can check, for an arbitrary element $\tau \in D \smallsetminus D'$, if $(D, \succ^r, \tau, D \smallsetminus (D' \cup \{\tau\}) \in \mathcal{MPCDP}(V^{\Sigma})$. $\square$

It is worth contrasting this result with the tractability result in Proposition 5.5.1 for the *minimal contingency set decision problem* (MCSDP) for actual causes. Notice that Proposition 5.5.1 still holds for UBCQs with inequality.

Notice that we could generalize the notion of preferred cause by appealing to any notion of repair. More precisely, if we have a *repair semantics* rSem (based on tuple deletions for DCs), we could replace $Rep^{\succ,\mathcal{X}}(D,\kappa(\mathcal{Q}))$ in (6.2) by $Rep^{\mathsf{rSem}}(D,\kappa(\mathcal{Q}))$. However, to obtain the intended results for causes, we have to be careful, as above, about a possible inverse relationship between preference on repairs and preference on causes.

6.3 Endogenous Repairs

The partition of a database into endogenous and exogenous tuples that is used in the causality setting may also be of interest in the context of repairs. Considering that we should have more control on endogenous tuples than on exogenous ones, which may come from external sources, it makes sense to consider *endogenous repairs*, which would be obtained by updates (of any kind) on endogenous tuples only. (Of course, a symmetric treatment of "exogenous" repairs is also possible; what is relevant here is the partition.)

For example, in the case of DCs, endogenous repairs would be obtained by deleting endogenous tuples only. More formally, given $D = D^n \cup D^x$, possibly inconsistent with a set of DCs Σ, an *endogenous repair* D' of D is a maximally consistent sub-instance of D with $D \smallsetminus D' \subseteq D^n$, i.e. D' keeps all the exogenous tuples of D. If endogenous repairs form the class $Srep^n(D,\Sigma)$, it holds $Srep^n(D,\Sigma) \subseteq Srep(D,\Sigma)$.

Example 6.3.1 Consider $D = D^n \cup D^x$, with $D^n = \{R(a_2,a_1), R(a_4,a_3), S(a_3), S(a_4)\}$ and $D^x = \{R(a_3,a_3), S(a_2)\}$, and the DC $\kappa\colon\ \neg\exists xy(S(x) \wedge R(x,y) \wedge S(y))$.

Here, $Srep(D,\kappa(\mathcal{Q})) = \{D_1,\ D_2,\ D_3\}$, with $D_1 = \{R(a_2,a_1), R(a_4,a_3),\ R(a_3,a_3), S(a_4), S(a_2)\}$, $D_2 = \{R(a_2,a_1), S(a_3), S(a_4), S(a_2)\}$, and $D_3 = \{R(a_2,a_1), R(a_4,a_3), S(a_3), S(a_2)\}$. The only endogenous S-repair is D_1. $\square$

In this section, without trying to be exhaustive or detailed, we consider the possibility of defining endogenous repairs on the basis of a suitable priority relation $\succ$ on tuples,[2] while at the same time taking advantage of the *op* optimality condition considered in Section 6.1.[3]

First, if we assume that relation $\succ'$, the extension of $\succ$, is such, that $\tau \succ' \tau'$ when $\tau \in D^x$ and $\tau' \in D^n$ ($\succ'$ is $\succ$ if the latter already has this property), then it is easy to verify that every endogenous S-repair globally improves any non-endogenous S-repair. As a consequence, if there is an endogenous S-repair, then all the $(\succ', go)$-repairs are endogenous.

[2]Pairs of conflicting tuples would inherit the priority relationships from the general priority relation.

[3]Of course, we could use other optimality criteria at this points, but considering all possibilities is beyond the scope of this work.

Notice that the extension $\succ'$ may destroy the acyclicity assumption on the priority relation, because we are starting from a given (acyclic) relation $\succ$, which we are now extending.

It might be the case that there is no endogenous S-repair, in which case non-endogenous S-repairs would not be improved by an endogenous one. So, if we want only endogenous repairs, we can add an extra, dummy predicate $D(\cdot)$ to the schema, and the endogenous tuple $D(d)$ to D. We modify every DC, say $\kappa: \leftarrow\ C(\bar{x})$, by adding an extra, dummy condition: $\kappa^d: \leftarrow\ D(d), C(\bar{x})$. In this case, the S-repairs will be: $D^d := D \smallsetminus \{D(d)\}$, which is endogenous, and also all those S-repairs of D with respect to Σ (now each including $D(d)$). If we assume that $D(d) \succ' \tau$, for every $\tau \in D^n$, then every non-endogenous S-repair will be improved by D^d, and will be discarded.

If we get rid of the original priority relationships $\tau \succ \tau'$, with $\tau \in D^n, \tau' \in D^x$, if any, then the $(\succ', go)$-repairs of $D \cup \{D(d)\}$ with respect to Σ^d will be all endogenous, namely D^d plus the $D' \cup \{D(d)\}$, where D' is an endogenous $(\succ, go)$- repair of D with respect to Σ. In particular, if the only priority relationships are $D(d) \succ \tau$, with $\tau \in D^n$, then we obtain as repairs: D^d plus all the endogenous S-repairs of D with respect to Σ (each of them now including also the tuple $D(d)$).

6.4 Null-Based Causes

Consider an instance $D = \{R(c_1, \ldots, c_n), \ldots\}$ that may be inconsistent with respect to a set of DCs. The allowed repair updates are changes of attribute values by the constant *null*. We assume that $null$ does not join with any other value, including *null* itself.

In order to keep track of changes, we may introduce numbers as first arguments in tuples, as global tuple identifiers (ids). So, D becomes $D = \{R(1; c_1, \ldots, c_n), \ldots\}$. Assume that $id(\tau)$ returns the id of the tuple $\tau \in D$. For example, $id(R(1; c_1, \ldots, c_n)) = 1$.

If, by updating D into D' in this way, the value of the ith attribute in R is changed to *null*, then the change is captured as the string $R[1; i]$. These strings are collected forming the set $\mathit{Diff}^{\,null}(D, D')$. For example, if $D = \{R(1; a, b), S(2; c, d), S(3; e, f)\}$ is changed into $D' = \{R(1; a, \mathit{null}), S(2; \mathit{null}, d),\ S(3; \mathit{null}, \mathit{null})\}$, we have $\mathit{Diff}^{\,null}(D, D') = \{R[1; 2], S[2; 1], S[3; 1], S[3; 2]\}$.

A *null*-repair of D with respect to a set of DCs Σ is a consistent instance D', such that $\mathit{Diff}^{\,null}(D, D')$ is minimal under set inclusion.[4] $\mathit{Rep}^{null}(D, \Sigma)$ denotes the class of null-based repairs of D with respect to Σ.

Example 6.4.1 (ex. 6.3.1 cont.) Consider the following inconsistent instance with respect to DC κ: $\neg\exists xy(S(x) \wedge R(x, y) \wedge S(y))$:

$$D = \{R(1; a_2, a_1), R(2; a_3, a_3), R(3; a_4, a_3), S(4; a_2), S(5; a_3), S(6; a_4)\}.$$

For simplicity, we do not make any difference between endogenous and exogenous tuples. Here, the class of *null-based repairs*, $\mathit{Rep}^{null}(D, \kappa)$, is formed by:

$$D_1 = \{R(1; a_2, a_1), R(2; a_3, a_3), R(3; a_4, a_3), S(4; a_2), S(5; \mathit{null}), S(6; a_4)\},$$
$$D_2 = \{R(1; a_2, a_1), R(2; \mathit{null}, a_3), R(3; a_4, \mathit{null}), S(4; a_2), S(5; a_3), S(6; a_4)\},$$
$$D_3 = \{R(1; a_2, a_1), R(2; \mathit{null}, a_3), R(3; a_4, a_3), S(4; a_2), S(5; a_3), S(6; \mathit{null})\},$$
$$D_4 = \{R(1; a_2, a_1), R(2; a_3, \mathit{null}), R(3; a_4, \mathit{null}), S(4; a_2), S(5; a_3), S(6; a_4)\},$$
$$D_5 = \{R(1; a_2, a_1), R(2; a_3, \mathit{null}), R(3; \mathit{null}, a_3), S(4; a_2), S(5; a_3), S(6; a_4)\},$$
$$D_6 = \{R(1; a_2, a_1), R(2; a_3, \mathit{null}), R(3; a_4, a_3), S(4; a_2), S(5; a_3), S(6; \mathit{null})\}.$$

Here, $\mathit{Diff}^{\,null}(D, D_1) = \{S[5; 1]\}$, and $\mathit{Diff}^{\,null}(D, D_6) = \{R[2; 2], S[6; 1]\}$. $\square$

According to the motivation provided at the beginning of this section, we can now define causes appealing to the class of null-based repairs of D. Since repair actions in this case, are attribute-value changes, causes can be defined at both the tuple and attribute levels. The same applies to the definition of responsibility (in this case generalizing Proposition 5.3.2).

Definition 6.4.1 For D an instance and $\mathcal{Q}$ a BCQ, and $\tau \in D$ be a tuple of the form $R(i; c_1, \ldots, c_n)$.

(a) $R[i; c_j]$ is a *null-based attribute-value cause* for $\mathcal{Q}$ if there is $D' \in \mathit{Rep}^{null}(D, \kappa(\mathcal{Q}))$ with $R[i; j] \in \mathit{Diff}^{\,null}(D, D')$.

(That is, the value c_j for attribute A_j in the tuple is a cause if it is changed into a null in some repair.)

(b) τ is a *null-based tuple cause* for $\mathcal{Q}$ if some $R[i; c_j]$ is a *null-based attribute-value cause* for $\mathcal{Q}$.

[4] An alternative, but equivalent formulation can be found in [Bravo et al., 2006, Bertossi & Li, 2013].

(That is, the whole tuple is a cause if at least one of its attribute values is changed into a null in some repair.)

(c) The responsibility, $\rho^{\textit{t-null}}(\tau)$, of τ, a *null-based tuple cause* for $\mathcal{Q}$, is the inverse of $min\{|\mathit{Diff}^{\,null}(D,D')| \ : \ R[i;j] \in \mathit{Diff}^{\,null}(D,D')$, for some j, and $D' \in \mathit{Rep}^{null}(D,\kappa(\mathcal{Q}))\}$.

(d) The responsibility, $\rho^{\textit{a-null}}(R[i;c_j])$, of $R[i;c_j]$, a *null-based attribute-value cause* for $\mathcal{Q}$, is the inverse of $min\{|\mathit{Diff}^{\,null}(D,D')| \ : \ R[i;j] \in \mathit{Diff}^{\,null}(D,D')$, and $D' \in \mathit{Rep}^{null}(D,\kappa(\mathcal{Q}))\}$. $\square$

In cases (c) and (d) we minimize over the number of changes in a repair that are made together with that of the candidate tuple/attribute-value to be a cause. In the case of a tuple cause, any change made in one of its attributes is considered in the minimization. For this reason, the minimum may be smaller than the one for a fixed attribute value change; and so the responsibility at the tuple level may be greater than that at the tuple level. More precisely, if $\tau = R(i;c_1,\ldots,c_n) \in D$, and $R[i;c_j])$ is a *null-based attribute-value cause*, then it holds $\rho^{\textit{a-null}}(R[i;c_j]) \leq \rho^{\textit{t-null}}(\tau)$.

Example 6.4.2 (ex. 6.4.1 cont.) Consider $R(2;a_3,a_3) \in D$. Its projection on its first (non-id) attribute, $R[2;a_3]$, is an attribute-level cause since $R[2;1] \in \mathit{Diff}^{\,null}(D,D_2)$. Also $R[2;1] \in \mathit{Diff}^{\,null}(D,D_3)$.

Since $|\mathit{Diff}^{\,null}(D,D_2)| = |\mathit{Diff}^{\,null}(D,D_3)| = 2$, it holds $\rho^{\textit{a-null}}(R[2;1]) = \frac{1}{2}$.

Clearly $R(2;a_3,a_3)$ is a *null-based tuple cause* for $\mathcal{Q}$, with $\rho^{\textit{t-null}}(\tau) = \frac{1}{2}$. $\square$

Notice that the definition of tuple-level responsibility, i.e. case (c) in Definition 6.4.1, does not take into account that a same id, i, may appear several times in a $\mathit{Diff}^{\,null}(D,D')$. In order to do so, we could redefine the size of the latter by taking into account those multiplicities. For example, if we decrease the size of the *Diff* by one with every repetition of the id, the responsibility for a cause may (only) increase, which makes sense.

Chapter 7

Causality and Abductive Diagnosis

In general logical terms, an abductive explanation for an observation is a formula that, together with a background logical theory, entails the observation. Although one could see an abductive explanation as a cause for the observation, it has been argued that causes and abductive explanations are not necessarily the same [Psillos, 1996, Denecker et al., 2002].

Under the abductive approach to diagnosis [Console et al., 1991a, Eiter et al., 1995, Poole, 1992, Poole, 1994], it is common that the system specification rather explicitly describes causal information, specially in action theories where the effects of actions are directly represented by positive definite rules. By restricting the explanation formulas to the predicates describing primitive causes (action executions), an explanation formula which entails an observation gives also a cause for the observation [Denecker et al., 2002]. In this case, and is some sense, causality information is imposed by the system specifier [Poole, 1992].

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

In this chapter we will first establish connections between abductive diagnosis and database causality. We start by making precise the kind of abduction problems we will consider.

7.1 Datalog Abductive Diagnosis

A *Datalog abduction problem* [Eiter et al., 1997] is of the form $\mathcal{AP} = \langle \Pi, E, \mathit{Hyp}, \mathit{Obs} \rangle$, where: (a) Π is a set of Datalog rules, (b) E is a set of ground atoms (the extensional database), (c) Hyp, the hypothesis, is a finite set of ground atoms, the abducible atoms in this case,[1] and (d) Obs, the observation, is a finite conjunction of ground atoms. As it is common, we will start with the assumption that $\Pi \cup E \cup Hyp \models Obs$. $\Pi \cup E$ is called the *background theory* (or specification).

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

Definition 7.1.1 Consider a *Datalog abduction problem* $\mathcal{AP} = \langle \Pi, E, Hyp, Obs \rangle$.

(a) An *abductive diagnosis* (or simply, *a solution*) for $\mathcal{AP}$ is a subset-minimal $\Delta \subseteq Hyp$, such that $\Pi \cup E \cup \Delta \models Obs$.

This requires that no proper subset of Δ has this property. $Sol(\mathcal{AP})$ denotes the set of abductive diagnoses for problem $\mathcal{AP}$.

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

(c) A hypothesis $h \in Hyp$ is *necessary* for $\mathcal{AP}$ if h contained in all diagnosis of $\mathcal{AP}$. $Ness(\mathcal{AP})$ collects all the necessary hypothesis for $\mathcal{AP}$. □

Notice that for an ADP, $\mathcal{AP}$, $Sol(\mathcal{AP})$ is never empty due to the assumption $\Pi \cup D \cup Hyp \models Obs$. In case, $\Pi \cup D \models Obs$, it holds $Sol(\mathcal{AP}) = \{\emptyset\}$.

Example 7.1.1 (ex. 5.2.1 cont.) Consider again the digital circuit in Figure 5.1 with the same inputs. Now, we formulate this diagnosis problem as a Datalog abduction where, the data domain is $\{a, b, c, d, e, and, or\}$. The underlying, extensional database is as follows: $E = \{One(a), Zero(b), One(c), And(a, b, e, and), Or(e, c, d, or\}$.

The Datalog program Π contains rules that model the normal and the faulty behavior of each gate. We show only the Datalog rules for the *And* gate. For its normal behavior, we have the following rules:

$$\begin{aligned}
One(O) &\leftarrow And(I_1, I_2, O, G), One(I_1), One(I_2)\\
Zero(O) &\leftarrow And(I_1, I_2, O, G), One(I_1), Zero(I_2)\\
Zero(O) &\leftarrow And(I_1, I_2, O, G), Zero(I_1), One(I_2)\\
Zero(O) &\leftarrow And(I_1, I_2, O, G), Zero(I_1), One(I_2).
\end{aligned}$$

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

The faulty behavior is modeled by the following rules:

$$\begin{array}{rcl}
Zero(O) & \leftarrow & And(I_1, I_2, O, G), One(I_1), One(I_2), Faulty(G) \\
One(O) & \leftarrow & And(I_1, I_2, O, G), One(I_1), Zero(I_2), Faulty(G) \\
One(O) & \leftarrow & And(I_1, I_2, O, G), Zero(I_1), One(I_2), Faulty(G) \\
One(O) & \leftarrow & And(I_1, I_2, O, G), Zero(I_1), One(I_2), Faulty(G)
\end{array}$$

Finally, we consider $Obs: Zero(d)$, and $Hyp = \{Faulty(and)), Faulty(or)\}$. The abduction problem consists in finding minimal $\Delta \subseteq Hyp$, such that $\Pi \cup E \cup \Delta \models Zero(d)$. There is one abductive diagnosis: $\Delta = \{Faulty(or)\}$. $\square$

In the context of Datalog abduction, we are interested in deciding, for a fixed Datalog program, if a hypothesis is relevant/necessary or not, with all the data as input. More precisely, we consider the following decision problems.

Definition 7.1.2 Given a Datalog program Π, the *relevance decision problem* (RLDP) *for* Π is (deciding about the membership of):

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

Definition 7.1.3 Given a Datalog program Π, the *necessity decision problem* (NDP) *for* Π is (deciding about the membership of):

$\mathcal{NDP}(\Pi) = \{(E, Hyp, Obs, h) \mid h \in Ness(\mathcal{AP}), \text{with } \mathcal{AP} = \langle \Pi, E, Hyp, Obs \rangle\}$. $\square$

As it is common, we will assume that $|Obs|$, i.e. the number of atoms in the conjunction, is bounded above by a fixed parameter p. In many cases, $p = 1$ (a single atomic observation).

The last two definitions suggest that we are interested in the *data complexity* of the relevance and necessity decision problems for Datalog abduction. That is, the Datalog program is fixed, but the data consisting of hypotheses and input structure E may change. In contrast, under *combined complexity* the program is also part of the input, and the complexity is measured also in terms of the program size.

A comprehensive complexity analysis of several reasoning tasks on abduction from propositional logic programs, in particular of the relevance and necessity problems, is provided in [Eiter et al., 1997]. Those results are all in combined complexity. In [Eiter et al., 1997], it has been shown that for abduction from function-free first-order logic programs, the data complexity of each type of reasoning problem on the first-order case coincides with the combined complexity of the same type of reasoning problem in the propositional case. In this way, the following results can be obtained for NDP and RLDP from [Eiter et al., 1997, theo. 26] and the combined complexity of these problems for *propositional Horn abduction* (PDA), established in [Friedrich et al., 1990]. We provide here direct, *ad hoc* proofs by adapting the full machinery developed in [Eiter et al., 1997] for general programs.

Proposition 7.1.1 For every Datalog program Π, $\mathcal{NDP}(\Pi)$ is in *PTIME* (in data).

Proof: Consider a DAP $\mathcal{AP} = \langle \Pi, E, \mathit{Hyp}, \mathit{Obs} \rangle$ associated to Π, and $h \in \mathit{Hyp}$. From the subset-minimality of abductive diagnosis and Definition 7.1.1 (part (c)), we obtain $h \in \mathit{Ness}(\mathcal{AP})$ iff $\mathit{Sol}(\mathcal{AP}') = \emptyset$ where, $\mathcal{AP}' = \langle \Pi, E, \mathit{Hyp} \smallsetminus \{h\}, \mathit{Obs} \rangle$. To decide whether $\mathit{Sol}(\mathcal{AP}') = \emptyset$, it is good enough to check if $\Pi \cup E \cup \mathit{Hyp} \models \mathit{Obs}$. This can be done in polynomial time since Datalog evaluation is in polynomial time in data complexity. $\square$

Proposition 7.1.2 For Datalog programs Π, $\mathcal{RLDP}(\Pi)$ is *NP*-complete (in data).[2]

Proof: *Membership:* Consider a Datalog abduction problem $\mathcal{AP}$ and a hypotheses $h \in \mathit{Hyp}$. To check whether h is relevant for $\mathcal{AP}$, non-deterministically guess a subset $\Delta \subseteq \mathit{Hyp}$, check if: (a) $h \in \Delta$, and (b) Δ is an abductive diagnosis for $\mathcal{AP}$. If h passes both tests then it is relevant, otherwise, it is irrelevant.

Clearly, test (a) can be performed in polynomial time. We only need to show that checking (b) is also polynomial time. More precisely, we need to show that $\Pi \cup E \cup \Delta \models \mathit{Obs}$ and Δ is subset-minimal. Checking whether $\Pi \cup E \cup \Delta \models \mathit{Obs}$ can be done in polynomial time, because Datalog evaluation is polynomial time. It is easy to verify that to check the minimality of Δ, it is good enough to show that for all elements $\delta \in \Delta$, $\Pi \cup E \cup \Delta \smallsetminus \{\delta\} \not\models \mathit{Obs}$. This is because positive Datalog is monotone.

[2]More precisely, this statement (and others of this kind) means: (a) For every Datalog program Π, $\mathcal{RLDP}(\Pi) \in \mathit{NP}$; and (b) there are programs Π' for which $\mathcal{RLDP}(\Pi')$ is *NP*-hard (all this in data).

Hardness: We show that the combined complexity of deciding relevance for the Propositional Horn Clause Abduction (PHCA) problem, that is *NP*-complete [Friedrich et al., 1990], is a lower bound for the data complexity of the relevance problem for Datalog abduction.

A PHCA problem is of the form $\mathcal{P} = \langle \mathit{Var}, \mathcal{H}, \mathit{SD}, \mathcal{O}\rangle$, where Var is a finite set of propositional variables, $\mathcal{H} \subseteq \mathit{Var}$ contains hypotheses, SD is a set of definite propositional Horn clauses, and $\mathcal{O} \subseteq \mathit{Var}$ is the observation, with $\mathcal{H} \cap \mathcal{O} = \emptyset$. An abductive diagnosis for $\mathcal{P}$ is a subset-minimal $\Delta \subseteq \mathcal{H}$, such that $\Delta \cup \mathit{SD} \models \bigwedge_{o \in \mathcal{O}} o$. Deciding whether $h \in \mathcal{H}$ is relevant to $\mathcal{P}$ (i.e. it is an element of an abductive diagnosis of $\mathcal{P}$) is *NP*-complete [Friedrich et al., 1990].

Deciding relevance for PHCA remains *NP*-hard for the *3-bounded case* where: SD contains a rule "$\mathit{true} \leftarrow$", and all the other rules are of the form "$a \leftarrow b_1, b_2, b_3$".[3]

Now, we provide a polynomial-time reduction from the problem of deciding relevance for 3-bounded PHCA to our problem RLDP. To obtain data complexity for the latter, we need a fixed relational schema and a fixed Datalog program Π over it, so that inputs for relevance in 3-bounded PHCA are mapped to the extensional components of Π, where relevance is tested.

More precisely, given a 3-bounded PHCA $\mathcal{P}$, build the DAP problem $\mathcal{AP}^{\mathcal{P}} = \langle \Pi, E^{\mathcal{P}}, \mathit{Hyp}^{\mathcal{P}}, \mathit{Obs}^{\mathcal{P}}\rangle$ as follows, where Π is the following (non-propositional) Datalog program (whose underlying domain consists of the propositional variables in SD plus true):

$$\begin{aligned} T(\mathit{true}) &\leftarrow \\ T(x_0) &\leftarrow T(x_1), T(x_2), T(x_3), R(x_0, x_1, x_2, x_3). \end{aligned}$$

Furthermore, $E^{\mathcal{P}} := \{R(a, b_1, b_2, b_3) \mid a \leftarrow b_1, b_2, b_3$ appears in $\mathit{SD}\}$. Furthermore, $\mathit{Hyp} = \{T(a) \mid a \in \mathcal{H}\}$ and $\mathit{Obs} = \{T(a) \mid a \in \mathcal{O}\}$. Notice that this reduction can be done in polynomial-time in the size of $\mathcal{P}$.

It is possible to prove that: For a $\mathcal{P} = \langle \mathit{Var}, \mathcal{H}, \mathit{SD}, \mathcal{O}\rangle$ and a hypothesis $h \in \mathcal{H}$, h is relevant for $\mathcal{P}$ iff $T(h) \in \mathit{Rel}(\mathcal{AP}^{\mathcal{P}})$. $\square$

[3]Every PHCA can be transformed to an equivalent 3-bounded PHCA, because each rule $a \leftarrow b_1, b_2, \ldots, b_n$ can be equivalently replaced by two rules $a \leftarrow c, \ldots, b_n$ and $c \leftarrow b_1, b_2$. Furthermore, true can be used to augment rule bodies with less than three propositional variables.

The following example illustrates the last claim in the proof of Proposition 7.1.2.

Example 7.1.2 Consider the PHCA $\mathcal{P} = \langle\{a, b, c\}, \{c, b\}, \{a \leftarrow b, c;\ b \leftarrow c\}, \{a\}\rangle$. It is easy to verify that $\mathcal{P}$ has the single abductive diagnosis, $\{c\}$, and then a single relevant hypotheses, c.

The 3-bounded PHCA $\mathcal{P}^{3b} = \langle\{a, b, c\}, \{c, b\}, \{true \leftarrow;\ a \leftarrow b, c, true;\ b \leftarrow c, true, true\}, \{a\}\rangle$ is equivalent to $\mathcal{P}$.

$\mathcal{P}^{3b}$ is mapped to the DAP $\mathcal{AP}^{\mathcal{P}^{3b}} = \langle\Pi, \{R(a, b, c, true), R(c, b, true, true)\}, \{T(c), T(b)\}, \{T(a))\}\rangle$, which has a single abductive diagnosis, $\{T(c)\}$. $\square$

It is clear from this result that deciding relevance for Datalog abduction is also intractable in combined complexity. However, a tractable case of combined complexity is identified in [Gottlob et al., 2010], on the basis of the notions of *tree-decomposition and bounded tree-width*, which we now briefly present.

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

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

Intuitively, the tree-width of a hypergraph $\mathcal{H}$ is a measure of the "tree-likeness" of $\mathcal{H}$. A set of vertices that form a cycle in $\mathcal{H}$ are put into a same bag, which becomes (the bag of a) node in the corresponding tree-decomposition. If the tree-width of the hypergraph under consideration is bounded by a fixed constant, then many otherwise intractable problems become tractable [Gottlob et al., 2010].

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

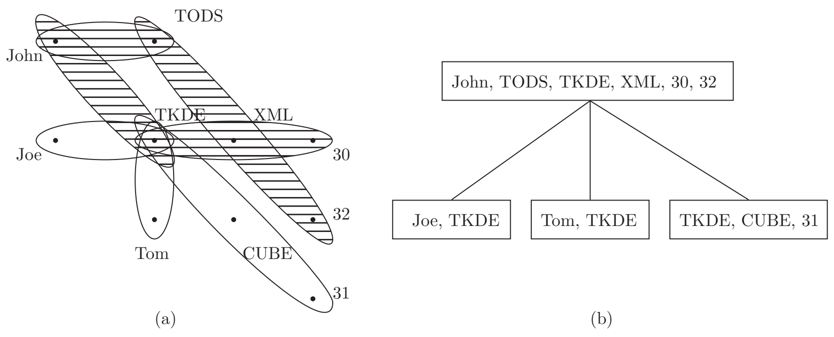

Figure 7.1: (a) $\mathcal{H}(D)$ (b) A tree decomposition of $\mathcal{H}(D)$

Example 7.1.3 Consider instance D in Example 1.1.1. The hypergraph $\mathcal{H}(D)$ associated to D is shown in Figure 7.1(a). Its vertices are the elements of $Adom(D) = \{John, Joe, Tom, TODS, TKDE, XML, Cube, 30, 31, 32\}$, the active domain of D. For example, since $Journal(TKDE, XML, 30) \in D$, $\{TKDE, XML, 30\}$ is one of the hyperedges.

The dashed ovals show four sets of vertices, i.e. hyperedges, that together form a cycle. Their elements are put into the same bag of the tree-decomposition. Figure 7.1(b) shows a possible tree-decomposition of $\mathcal{H}(D)$. In it, the maximum $|\lambda(n)|-1$ is $6-1$, corresponding to the top box bag of the tree. So, $t_w(\mathcal{H}(D)) \leq 5$. $\square$

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

Theorem 7.1.1 [Gottlob et al., 2010] Let k be an integer. For Datalog abduction problems $\mathcal{AP} = \langle \Pi, E, Hyp, Obs\rangle$ where Π is guarded, and $t_w(\mathcal{H}(E)) \leq k$, relevance can be decided in polynomial time in $|\mathcal{AP}|$.[4] More precisely, the following decision problem is tractable:

$$\mathcal{RLDP} = \{(\langle \Pi, E, Hyp, Obs\rangle, h) \mid h \in Rel(\langle \Pi, E, Hyp, Obs\rangle), h \in Hyp,$$
$$\Pi \text{ is guarded, and } t_w(\mathcal{H}(E)) \leq k\}. \quad \square$$

This is a case of tractable combined complexity with a fixed parameter that is the tree-width of the extensional database.

[4]This is Theorem 7.9 in [Gottlob et al., 2010].

7.2 Causality and Datalog Abductive Diagnosis

7.2.1 Actual causes from Datalog abductive diagnosis

Assume that Π is a boolean, possibly recursive Datalog query. Consider the relational instance $D = D^x \cup D^n$. Also assume that $\Pi \cup D \models \mathit{ans}$. Then, the decision problem in Definition 2.2.1 takes the form:

$$\mathcal{CDP}(\Pi) := \{(D, \tau) \mid \tau \in D^n, \text{and } \tau \in \mathit{Causes}(D, \Pi)\}. \tag{7.1}$$

We now show that actual causes for ans can be obtained from abductive diagnoses of the associated *causal Datalog abduction problem* (CDAP): $\mathcal{AP}^c := \langle \Pi, D^x, D^n, \mathit{ans}\rangle$, where D^x takes the role of the extensional database for Π. Accordingly, $\Pi \cup D^x$ becomes the *background theory*, D^n becomes the set of *hypothesis*, and atom ans is the observation.

Proposition 7.2.1 For an instance $D = D^x \cup D^n$ and a boolean Datalog query Π, with $\Pi \cup D \models \mathit{ans}$, and its associated CDAP $\mathcal{AP}^c$, the following hold:

(a) $\tau \in D^n$ is an counterfactual cause for ans iff $\tau \in \mathit{Ness}(\mathcal{AP}^c)$.

(b) $\tau \in D^n$ is an actual cause for ans iff $\tau \in \mathit{Rel}(\mathcal{AP}^c)$.

Proof: Part (a) is straightforward. To proof part (b), first assume τ is an actual cause for *ans*. According to the definition of an actual cause, there exists a contingency set $\Gamma \subseteq D^n$ such that $\Pi \cup D \smallsetminus \Gamma \models \mathit{ans}$ but $\Pi \cup D \smallsetminus (\Gamma \cup \{\tau\}) \not\models \mathit{ans}$. This implies that there exists a set $\Delta \subseteq D^n$ with $\tau \in \Delta$ such that $\Pi \cup \Delta \models \mathit{ans}$. It is easy to see that Δ is an abductive diagnosis for $\mathcal{AP}^c$. Therefore, $\tau \in \mathit{Rel}(\mathcal{AP}^c)$.

Second, assume $\tau \in \mathit{Rel}(\mathcal{AP}^c)$. Then there exists a set $\mathcal{S}_k \in \mathit{Sol}(\mathcal{AP}^c) = \{s_1 \ldots s_n\}$ such that $\mathcal{S}_k \models \mathit{ans}$ with $\tau \in \mathcal{S}_k$. Obviously, $\mathit{Sol}(\mathcal{AP}^c)$ is a collection of subsets of D^n. Pick a set $\Gamma \subseteq D^n$ such that for all $\mathcal{S}_i \in \mathit{Sol}(\mathcal{AP}^c)$ $i \neq k$, $\Gamma \cap \mathcal{S}_i \neq \emptyset$ and $\Gamma \cap \mathcal{S}_k = \emptyset$. It is clear that $\Pi \cup D \smallsetminus (\Gamma \cup \{t\}) \not\models \mathit{ans}$ but $\Pi \cup D \smallsetminus \Gamma \models \mathit{ans}$. Therefore, τ is an actual cause for *ans*. To complete the proof we need to show that such Γ always exists. This can be done by applying the digitalization technique to construct such Γ. Since all elements of $\mathit{Sol}(\mathcal{AP}^c)$ are subset-minimal, then, for each $\mathcal{S}_i \in \mathit{Sol}(\mathcal{AP}^c)$ with $i \neq k$, there exists a $\tau' \in \mathcal{S}_i$ such that $\tau' \notin \mathcal{S}_k$. So, Γ can be obtained from the union of differences between each $\mathcal{S}_i$ ($i \neq k$) and $\mathcal{S}_k$. □

Example 7.2.1 Consider the instance D with relations R and S as below, and the query $\Pi: \ ans \leftarrow R(x,y), S(y)$, which is true in D. Assume all tuples are endogenous.

R	A	B
	a_1	a_4
	a_2	a_1
	a_3	a_3

S	A
	a_1
	a_2
	a_3

In this case, $\mathcal{AP}^c = \langle \Pi, \emptyset, D, ans \rangle$, which has two (subset-minimal) abductive diagnoses: $\Delta_1 = \{S(a_1), R(a_2,a_1)\}$ and $\Delta_2 = \{S(a_3), R(a_3,a_3)\}$. Then, $Rel(\mathcal{AP}^c) = \{S(a_3),\ R(a_3,a_3),\ S(a_1), R(a_2,a_1)\}$. It is easy to see that the relevant hypothesis are actual causes for ans. □

7.2.2 Datalog abductive diagnosis and causal responsibility

In the previous section we showed that counterfactual and actual causes for Datalog query answers appear as necessary and relevant hypotheses in the associated Datalog abduction problem. The form causal responsibility takes in Datalog abduction is less direct. Actually, we first show that causal responsibility inspires an interesting concept for Datalog abduction, that of *degree of necessity* of a hypothesis.

Example 7.2.2 (ex. 7.2.1 cont.) Consider now $D' = \{R(a_1,a_3), R(a_2,a_3), S(a_3)\}$, and $\mathcal{AP}^c = \langle \Pi, \emptyset, D', ans \rangle$. $\mathcal{AP}^c$ has two abductive diagnosis: $\Delta_1 = \{S(a_3), R(a_1,a_3)\}$ and $\Delta_2 = \{S(a_3), R(a_2,a_3)\}$.

Here, $Ness(\mathcal{AP}^c) = \{S(a_3)\}$, i.e. only $S(a_3)$ is necessary for abductively explaining ans. However, this is not capturing the fact that $R(a_1,a_3)$ or $R(a_3,a_3)$ are also needed as a part of the explanation. □

This example suggests that necessary hypotheses might be better captured as sets of them rather than as individuals.

Definition 7.2.1 Given a DAP, $\mathcal{AP} = \langle \Pi, E, \mathit{Hyp}, \mathit{Obs} \rangle$, $N \subseteq \mathit{Hyp}$ is a *necessary-hypothesis set* if: (a) for $\mathcal{AP}_{-N} := \langle \Pi, E, \mathit{Hyp} \smallsetminus N, \mathit{Obs} \rangle$, $\mathit{Sol}(\mathcal{AP}_{-N}) = \emptyset$, and (b) N is subset-minimal, i.e. no proper subset of N has the previous property. □

It is easy to verify that a hypothesis h is necessary according to Definition 7.1.1 iff $\{h\}$ is a necessary-hypothesis set.

If we apply Definition 7.2.1 to $\mathcal{AP}^c$ in Example 7.2.2, we obtain two necessary-hypothesis sets: $N_1 = \{S(a_3)\}$ and $N_2 = \{R(a_1, a_3), R(a_2, a_3)\}$. In this case, it makes sense to claim that $S(a_3)$ is more necessary for explaining ans than the other two tuples, that need to be combined. Actually, we can think of ranking hypothesis according to the minimum-cardinality of necessary-hypothesis sets where they are included.

Definition 7.2.2 Given a DAP, $\mathcal{AP} = \langle \Pi, E, \mathit{Hyp}, \mathit{Obs}\rangle$, the *necessity-degree* of a hypothesis $h \in \mathit{Hyp}$ is $\eta_{\mathcal{AP}}(h) := \frac{1}{|N|}$ where, N is a minimum-cardinality necessary-hypothesis set with $h \in N$. If h does not belong to any necessary hypothesis set, $\eta_{\mathcal{AP}}(h) := 0$. $\square$

In Example 7.2.2, $\eta_{\mathcal{AP}^c}(S(a_3)) = 1$ and $\eta_{\mathcal{AP}^c}(R(a_2, a_3)) = \eta_{\mathcal{AP}^c}(R(a_1, a_3)) = \frac{1}{2}$. Now, if we consider the original Datalog query in the causality setting, where $\Pi \cup D' \models ans$, then $S(a_3), R(a_2, a_3), R(a_1, a_3)$ are all actual causes, with responsibilities: $\rho_{\Pi}(S(a_3)) = 1$, $\rho_{\Pi}(R(a_2, a_3)) = \rho_{\Pi}(R(a_1, a_3)) = \frac{1}{2}$. This is not a coincidence. In fact the notion of causal responsibility is in correspondence with the notion of necessity degree in the Datalog abduction setting.

Proposition 7.2.2 Let $D = D^x \cup D^n$ be an instance and Π be a boolean Datalog query with $\Pi \cup D \models ans$, and $\mathcal{AP}^c$ its associated CDAP. For $\tau \in D^n$, it holds: $\eta_{\mathcal{AP}^c}(\tau) = \rho_{\Pi}(\tau)$.

Proof: It is easy to verify that each actual cause, together with a contingency set, forms a necessary hypothesis set for the corresponding causal Datalog abduction setting (and the other way around). Then, the two values are in correspondence. $\square$

Notice that the notion of necessity-degree is interesting and applicable to general abduction from logical theories, that may not necessarily represent causal knowledge about a domain. In this case, the necessity-degree is not a causality-related notion, and merely reflects the extent by which a hypothesis is necessary for making an observation explainable within an abductive theory.

7.2.3 Datalog abductive diagnosis from actual causes

Now we show, conversely, that QA-causality can capture Datalog abduction. In particular, we show that abductive diagnoses from Datalog programs are formed essentially by actual causes for the observation. More precisely, consider a Datalog abduction problem $\mathcal{AP} = \langle \Pi, E, \mathit{Hyp}, \mathit{Obs} \rangle$, where E is the underlying extensional database, and Obs is a conjunction of ground atoms. For this we need to construct a QA-causality setting.

Proposition 7.2.3 Let $\mathcal{AP} = \langle \Pi, E, \mathit{Hyp}, \mathit{Obs} \rangle$ be a Datalog abduction problem, and $h \in \mathit{Hyp}$. It holds that h is a relevant hypothesis for $\mathcal{AP}$, i.e. $h \in \mathit{Rel}(\mathcal{AP})$, iff h is an actual cause for the associated boolean Datalog query $\Pi^c := \Pi \ \cup \ \{\mathit{ans} \leftarrow \mathit{Obs}\}$ being true in $D := D^x \cup D^n$ with $D^x := E$, and $D^n := \mathit{Hyp}$. Here, *ans* is a fresh propositional atom.

Proof: The proof is similar to that of Proposition 7.2.1. □

Example 7.2.3 Consider the DAP $\mathcal{AP}$ in Example 7.1.1. We construct a QA-causality setting as follows. Consider the instance D with relations *And*, *Or*, Faulty, *One* and *Zero*, as below, and the boolean Datalog query $\Pi^c : \ \Pi \ \cup \ \{\mathit{ans} \leftarrow \mathit{Zero}(d)\}$, where Π is the Datalog program in Example 7.1.1.

And	I_1	I_2	O	G
	a	b	e	*and*

Or	I_1	I_2	O	G
	e	c	d	*or*

Faulty	G
	and
	or

Zero	I
	b

One	I
	a
	c

It clear that $\Pi^c \cup D \models \mathit{ans}$. D is partitioned into the set of endogenous tuples $D^n := \{\mathit{Faulty}(\mathit{and}), \mathit{Faulty}(\mathit{or})\}$ and the set of exogenous tuples $D^x := D \smallsetminus D^n$.

It is easy to verify that this result has only one actual cause, namely $\mathit{Faulty}(\mathit{or})$ (with responsibility 1), confirming the correspondence with Example 7.1.1 as stated in Proposition 7.2.3. □

7.3 Complexity of Causality for Datalog Queries

Now we use the results obtained so far in this section to obtain new complexity results for Datalog QA-causality. We first consider the problem of deciding if a tuple is a *counterfactual cause* for a query answer.

A counterfactual cause is a tuple that, when removed from the database, undermines the query-answer, without having to remove other tuples, as is the case for *actual causes*. Actually, for each of the latter there may be an exponential number of contingency sets, i.e. of accompanying tuples (cf. Section 5.3.2). Notice that a counterfactual cause is an actual cause with responsibility 1.

Definition 7.3.1 For a boolean monotone query $\mathcal{Q}$, the *counterfactual causality decision problem* (CFDP) is (deciding about membership of):

$\mathcal{CFDP}(\mathcal{Q}) := \{(D, \tau) \mid \tau \in D^n \text{ and } \rho_{_\mathcal{Q}}(\tau) = 1\}$. □

The complexity of this problem can be obtained from the connection between counterfactual causation and the necessity of hypothesis in Datalog abduction via Propositions 7.1.1 and 7.2.1.

Proposition 7.3.1 For boolean Datalog queries Π, $\mathcal{CFDP}(\Pi)$ is in *PTIME* (in data).

Proof: Directly from Propositions 7.1.1 and 7.2.1. □

Now we address the complexity of the actual causality problem for Datalog queries. The following result is obtained from Propositions 7.1.2 and 7.2.3.

Proposition 7.3.2 For boolean Datalog queries Π, $\mathcal{CDP}(\Pi)$ is *NP*-complete (in data).

Proof: To show the membership of *NP*, consider an instance $D = D^n \cup D^x$ and a tuple $\tau \in D^n$. To check if $(D, \tau) \in \mathcal{CDP}(\Pi)$ (equivalently $\tau \in \mathit{Causes}(D, \Pi)$), nondeterministically guess a subset $\Gamma \subseteq D^n$, return *yes* if τ is a counterfactual cause for $\mathcal{Q}(\bar{a})$ in $D \smallsetminus \Gamma$, and *no* otherwise. By Proposition 7.3.1 this can be done in polynomial time.

The *NP*-hardness is obtained by a reduction from the relevance problem for Datalog abduction to causality problem, as given in Proposition 7.2.3. □

This result should be contrasted with the tractability of the same problem for UCQs (cf. Proposition 5.3.5).

We now introduce a fixed-parameter tractable case of the actual causality problem. Actually, we consider the "combined" version, $\mathcal{CDP}$, of the decision problem in Definition

2.2.1, where both the Datalog query and the instance are part of the input. For this, we take advantage of the tractable case of Datalog abduction presented in Section 7.1. The following is an immediate consequence of Theorem 7.1.1 and Proposition 7.2.1.

Proposition 7.3.3 For a guarded boolean Datalog query Π, an instance $D = D^x \cup D^n$, with D^x of bounded tree-width, and $\tau \in D^n$, deciding if $\tau \in \mathit{Causes}(D, \Pi)$ is fixed-parameter tractable (in combined complexity), and the parameter is the tree-width bound. $\Box$

Finally, we establish the complexity of the responsibility problem for Datalog queries.

Proposition 7.3.4 For boolean Datalog queries Π, $\mathcal{RDP}(\Pi)$ is *NP*-complete. $\Box$

Proof: To show membership of *NP*, consider an instance $D = D^n \cup D^x$, a tuple $\tau \in D^n$, and a responsibility bound v. To check if $\rho_{_\Pi}(\tau) > v$, non-deterministically guess a set $\Gamma \subseteq D^n$ and check if Γ is a contingency set and $\Gamma < \frac{1}{v}$. The verification can be done in polynomial time. Hardness is obtain from the *NP*-completeness of RDP for conjunctive queries established in (cf. Proposition 5.5.1) . $\Box$

Chapter 8

Causality and View-Update problem

There is a close relationship between QA-causality and the view-update problem in the form of delete-propagation. It was first suggested in [Meliou et al., 2010b, Kimelfeld, 2012b, Kimelfeld, 2012a], and here we investigate it more deeply.

8.1 View-Based Delete-Propagation

Given a monotone query $\mathcal{Q}$, we can think of it as defining a view, $\mathcal{V}$, with virtual contents $\mathcal{Q}(D)$. If $\bar{a} \in \mathcal{Q}(D)$, which may not be intended, we may try to delete some tuples from D, so that $\bar{a}$ disappears from $\mathcal{Q}(D)$. This is a particular case of database updates through views [Abiteboul et al., 1995], and may appear in different and natural formulations. The next example shows one of them.

Example 8.1.1 (ex. 2.2.3 cont.) In a particular version of the delete-propagation problem, the objective may be to delete a *minimum number* of tuples from the instance, so that an authorized access (unexpected answer to the query) is deleted from the query answers, while all other authorized accesses (other answers to the query) remain intact. □

In the following, we consider several variations of this problem, both in their functional and decision versions.

Definition 8.1.1 Let D be a database instance, and $\mathcal{Q}$ a monotone query.

(a) For $\bar{a} \in \mathcal{Q}(D)$, the *minimal-source-side-effect deletion-problem* is about computing a subset-minimal $\Lambda \subseteq D$, such that $\bar{a} \notin \mathcal{Q}(D \smallsetminus \Lambda)$.

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

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

(The superscript s stands for subset-minimal.)

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

(d) The *minimum-source-side-effect decision problem* is (deciding about the membership of):

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

(Here, c stands for cardinality.) □

Definition 8.1.2 [Buneman et al., 2002] Let D be a database instance D, and $\mathcal{Q}$ a monotone query.

(a) For $\bar{a} \in \mathcal{Q}(D)$, the *view-side-effect-free deletion-problem* is about computing a $\Lambda \subseteq D$, such that $\mathcal{Q}(D) \smallsetminus \{\bar{a}\} = \mathcal{Q}(D \smallsetminus \Lambda)$.

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

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

The decision problem in Definition 8.1.2(b) is *NP*-complete for conjunctive queries [Buneman et al., 2002, theo. 2.1]. Notice that, in contrast to Definition 8.1.1, this decision problem does not involve a candidate D', and only asks about its existence. This is because candidates always exist for Definition 8.1.1, whereas for the *view-side-effect-free deletion-problem* there may be no sub-instance that produces *exactly* the intended deletion from the view.

Example 8.1.2 Consider the instance D and the conjunctive query $\mathcal{Q}$ in Example 1.1.1. Suppose that *XML* is not among *John*'s research interests, so that tuple $\langle John, XML\rangle$ in the view $\mathcal{Q}(D)$ is unintended. We want to find tuples in D whose removal leads to the deletion of this view tuple. There are multiple ways to remove tuples from D to achieve this goal, and the decision problems described above impose different conditions on how to do this.

Notice that tuples in D related to $\langle John, XML\rangle$ are *Author(John, TKDE)*, *Journal(TODS, XML, 32)*, *Author(John, TODS)* and *Journal(TKDE, XML, 30)*.

(a) Source-side effect: The objective is to find minimal/minimum sets of tuples whose removal leads to the deletion of $\langle John, XML\rangle$. One solution is removing S_1=$\{$*Author(John, TODS), Author(John, TKDE)*$\}$ from the *Author* table. The other solution is removing S_2=$\{$*Journal(TODS, XML, 30)*, *Journal(TKDE, XML, 30)*$\}$ from the *Journal* table.

Furthermore, the removal of either S_3=$\{$*Author(John, TKDE)*, *Journal(TODS, XML, 32)*$\}$ or S_4=$\{$*Author(John, TODS)*, *Journal(TKDE, XML, 30)*$\}$ eliminate the intended view tuple. Thus, S_1, S_2, S_3 and S_4 are solutions to both the minimum- and minimal-source side-effect deletion-problems.

(b) View-side effect: Removing any of the sets S_1, S_2, S_3 or S_4, leads to the deletion of $\langle John, XML\rangle$. However, we now want those set whose elimination produce no side-effects on the view. That is, their deletion triggers the deletion of $\langle John, XML\rangle$ from the view, but not of any other tuple in it.

None of the sets S_1, S_2, S_3 and S_4 is side-effect free. For example, the deletion of S_1 also results in the deletion of $\langle John, CUBE\rangle$ from the view. □

Example 8.1.3 (ex. 2.2.5 cont.) It is easy to verify that there is no solution to the view-side-effect-free deletion-problem for answer $\langle Tom, f_3\rangle$. To eliminate this entry from the view, either $GroupUser(Tom,g_3)$ or $GroupFiles(f_3, g_3)$ must be deleted from D. Removing the former results in the additional deletion of $\langle Tom, f_1\rangle$ from the view; and eliminating the latter, results in the additional deletion of $\langle John, f_3\rangle$.

However, there is a solution to the view-side-effect-free deletion-problem for answer $\langle Joe, f_1\rangle$, by removing $GroupUser(Joe, g_1)$ from D. This deletion does not have any unintended side-effects on the view contents. □

8.2 Causality and Delete-propagation

In this section we first establish mutual reductions between the different variants of the delete-propagation problem and both QA-causality and view-conditioned causality.

8.2.1 Delete-propagation from actual causes

In this section, unless otherwise is stated, all tuples in database instances are assumed to be endogenous.

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

For a tuple $\bar{a} \in \mathcal{Q}(D)$, the delete-propagation problem, in its most general form, is about deleting a set of tuples from D, and so obtaining a subinstance D' of D, such that $\bar{a} \notin \mathcal{Q}(D')$. It is natural to expect that the deletion of $\bar{a}$ from $\mathcal{Q}(D)$ can be achieved through deletions from D of actual causes for $\bar{a}$ (to be in the view extension). However, to obtain solutions to the different variants of this problem introduced in Section 8.1, different combinations of actual causes must be considered.

First, we show that an actual cause for $\bar{a}$ forms, with any of its contingency sets, a solution to the minimal-source-side-effect deletion-problem associated to $\bar{a}$ (cf. Definition 8.1.1).

Proposition 8.2.1 For an instance D, a subinstance $D' \subseteq D$, a view $\mathcal{V}$ defined by a monotone query $\mathcal{Q}$, and $\bar{a} \in \mathcal{Q}(D)$, $(D, D', \bar{a}) \in \mathcal{MSSEP}^s(\mathcal{Q})$ iff there is a $\tau \in D \smallsetminus D'$, such that $\tau \in \mathit{Causes}(D, \mathcal{Q}(\bar{a}))$ and $D \smallsetminus (D' \cup \{\tau\}) \in \mathit{Cont}(D, \mathcal{Q}(\bar{a}), \tau)$.

Proof: Suppose first that $(D, D', \bar{a}) \in \mathcal{MSSEP}^s(\mathcal{Q})$. Then, according to Definition 8.1.1, $\bar{a} \notin \mathcal{Q}(D')$. Let $\Lambda = D \smallsetminus D'$. For an arbitrary element $\tau \in \Lambda$ (clearly, $\Lambda \neq \emptyset$), let $\Gamma := \Lambda \smallsetminus \{\tau\}$. Due to the subset-maximality of D' (then, subset-minimality of Λ), we obtain: $D \smallsetminus (\Gamma \cup \{\tau\}) \not\models \mathcal{Q}(\bar{a})$, but $D \smallsetminus \Gamma \models \mathcal{Q}(\bar{a})$. Therefore, τ is an actual cause for $\bar{a}$.

For the other direction, suppose $\tau \in \mathit{Causes}(D, \mathcal{Q}(\bar{a}))$ and $D \smallsetminus (D' \cup \{\tau\}) \in \mathit{Cont}(D, \mathcal{Q}(\bar{a}), \tau)$. Let $\Gamma := D \smallsetminus (D' \cup \{\tau\})$. From the definition of an actual cause, we obtain that $\bar{a} \notin \mathcal{Q}(D \smallsetminus (\Gamma \cup \{\tau\})$. So, $\bar{a} \notin \mathcal{Q}(D')$ (notice that $D' = D \smallsetminus (\Gamma \cup \{\tau\})$. Since Γ is a subset-minimal contingency set for τ, D' is a subset-maximal subinstance that enjoys the mentioned property. So, $(D, D', \bar{a}) \in \mathcal{MSSEP}^s(\mathcal{Q})$. □

Next we show that, in order to minimize the number of side-effects on the source (cf. Definition 8.1.1(c)), it is good enough to pick a most responsible cause for $\bar{a}$ with any of its minimum-cardinality contingency sets.

Proposition 8.2.2 For an instance D, a subinstance $D' \subseteq D$, a view $\mathcal{V}$ defined by a monotone query $\mathcal{Q}$, and $\bar{a} \in \mathcal{Q}(D)$, $(D, D', \bar{a}) \in \mathcal{MSSEP}^c(\mathcal{Q})$ iff there is a $\tau \in D \smallsetminus D'$, such that $\tau \in \mathcal{MRC}(D, \mathcal{Q}(\bar{a}))$, $\Gamma := D \smallsetminus (D' \cup \{\tau\}) \in \mathit{Cont}(D, \mathcal{Q}(\bar{a}), \tau)$, and there is no $\Gamma' \in \mathit{Cont}(D, \mathcal{Q}(\bar{a}), \tau)$ with $|\Gamma'| < |\Gamma|$.

Proof: Similar to the proof of Proposition 8.2.1. □

Now, we show that, in order to check if there exists a solution to the view-side-effect-free deletion-problem for $\bar{a} \in \mathcal{V}(D)$ (cf. Definition 8.1.2), it is good enough to check if $\bar{a}$ has a view-conditioned cause.[1]

Proposition 8.2.3 For an instance D, a view $\mathcal{V}$ defined by a monotone query $\mathcal{Q}$ with $\mathcal{Q}(D) = \{\bar{a}_1, \ldots, \bar{a}_n\}$, and $\bar{a_k} \in \mathcal{Q}(D)$, $(D, \bar{a}_k) \in \mathcal{VSEFP}(\mathcal{Q})$ iff $\mathit{vc}\text{-}\mathit{Causes}(D, \mathcal{Q}(\bar{a}_k)) \neq \emptyset$.

Proof: Assume $\bar{a}_k$ has a view-conditioned cause τ. According to Definition 2.2.4, there exists a $\Gamma \subseteq D$, such that $D \smallsetminus (\Gamma \cup \{\tau\}) \not\models \mathcal{Q}(\bar{a}_k)$, $D \smallsetminus \Gamma \models \mathcal{Q}(\bar{a}_k)$, and $D \smallsetminus (\Gamma \cup \{\tau\}) \models \mathcal{Q}(\bar{a}_i)$, for $i \in \{1, \ldots, n\} \smallsetminus \{k\}$. So, $\Gamma \cup \{\tau\}$ is a view-side-effect-free deletion-problem solution for $\bar{a}_k$; and $(D, \bar{a}_k) \in \mathcal{VSEFP}(\mathcal{Q})$. A similar argument applies for the other direction. □

Example 8.2.1 (ex. 8.1.2 cont.) We had obtained the followings solutions to the minimum- (and also minimal-) source-side-effect deletion-problem for $\langle \mathit{John}, \mathit{XML}\rangle$:

S_1=\{*Author(John, TODS), Author(John, TKDE)*\},

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

S_3=\{*Author(John, TKDE), Journal(TODS, XML, 32)*\}, and

S_4=\{*Author(John, TODS), Journal(TKDE, XML, 30)*\}.

In Example 1.1.1, we showed that *Author(John, TODS)*, *Journal(TKDE, XML, 32)*, *Author(John, TKDE)*, *Journal(TODS, XML, 32)* are actual causes for $\langle \mathit{John}, \mathit{XML}\rangle$. For

[1]Since this delete-propagation problem does not explicitly involve anything like contingency sets, the existential problem in Definition 8.1.2(b) is the right one to consider.

the cause *Author(John, TODS)*, specifically, we obtained two contingency sets: $\Gamma_1 = \{\mathit{Author}(\mathit{John}, \mathit{TKDE})\}$ and Γ_2={*Journal(TKDE, XML, 32)*}.

It is easy to verify that each actual cause for answer $\langle \mathit{John}, \mathit{XML}\rangle$, together with any of its subset-minimal (and minimum-cardinality) contingency sets, forms a solution to the minimal- (and minimum-) source-side-effect deletion-problem for $\langle \mathit{John}, \mathit{XML}\rangle$. For illustration, $\{\mathit{Author}(\mathit{John}, \mathit{TODS})\} \cup \Gamma_1$ coincides with S_1, and $\{\mathit{Author}(\mathit{John}, \mathit{TODS})\} \cup \Gamma_1$ coincides with S_4. Thus, both of them are solutions to minimal- (and minimum-) source-side-effect deletion-problem for $\langle \mathit{John}, \mathit{XML}\rangle$. This confirms Propositions 8.2.1 and 8.2.2.

Furthermore, we obtained in Example 8.1.2(b) that there is no view-side-effect-free solution to this problem, which coincides with the result obtained in Example 2.2.4, and confirms Proposition 8.2.3. □

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

Example 8.2.2 (ex. 8.2.1 cont.) Assume again that $\langle \mathit{John}, \mathit{XML}\rangle$ has to be deleted from the query answer (view extension). Assume now only the data in the *Journal* relation are reliable. Then, only deletions from the *Author* relation make sense. This can be captured by making *Journal*-tuples exogenous, and *Author*-tuples endogenous. With this partition, only $\mathit{Author}(\mathit{John}, \mathit{TODS})$ and $\mathit{Author}(\mathit{John}, \mathit{TKDE})$ are actual causes for $\langle \mathit{John}, \mathit{XML}\rangle$, with contingency sets $\Gamma = \{\mathit{Author}(\mathit{John}, \mathit{TKDE})\}$ and $\Gamma' = \{\mathit{Author}(\mathit{John}, \mathit{TODS})\}$, respectively (see Example 1.1.1).

Now, each actual cause for $\langle \mathit{John}, \mathit{XML}\rangle$, together with its one-tuple subset-minimal (and also minimum-cardinality) contingency set, leads to the same set {*Author(John,TODS), Author(John,TKDE)*}, which, according to Propositions 8.2.1 and 8.2.2, is an endogenous minimal- (and minimum-) delete-propagation solution for $\langle \mathit{John}, \mathit{XML}\rangle$. □

8.2.2 Actual causes from delete-propagation

Consider a relational instance D, and a monotone query $\mathcal{Q}$ with $\bar{a} \in \mathcal{Q}(D)$. We will show that actual causes, most responsible causes, and vc-causes for $\bar{a}$ can be obtained from different variants of the delete-propagation problem associated with $\bar{a}$.

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

Proposition 8.2.4 For an instance $D = D^n \cup D^x$ and a monotone query $\mathcal{Q}(\bar{x})$ with $\bar{a} \in \mathcal{Q}(\bar{x})$, $\tau \in D^n$ is an actual cause for $\bar{a}$ iff there is a $D' \subseteq D$ with $\tau \in (D \smallsetminus D') \subseteq D^n$ and $(D, D', \bar{a}) \in \mathcal{MSSEP}^s(\mathcal{Q})$.

Proof: Suppose $\tau \in D^n$ is an actual cause for $\bar{a}$ with a subset-minimal contingency set $\Gamma \subseteq D^n$. Let $\Lambda = \Gamma \cup \{\tau\}$ and $D' = D \smallsetminus \Lambda$. It is clear that $\bar{a} \notin \mathcal{Q}(D')$. Then, due to the subset-minimality of Λ, we obtain that $(D, D', \bar{a}) \in \mathcal{MSSEP}^s(\mathcal{Q})$. A similar argument applies to the other direction. $\square$

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

Proposition 8.2.5 For an instance $D = D^n \cup D^x$ and a monotone query $\mathcal{Q}(\bar{x})$ with $\bar{a} \in \mathcal{Q}(\bar{x})$, $\tau \in D^n$ is a most responsible actual cause for $\bar{a}$ iff there is a $D' \subseteq D$ with $t \in (D \smallsetminus D') \subseteq D^n$ and $(D, D', \bar{a}) \in \mathcal{MSSEP}^c(\mathcal{Q})$.

Proof: Similar to the proof of Proposition 8.2.4. $\square$

Finally, vc-causes for an answer can be obtained from solutions to a corresponding view-side-effect-free deletion-problem.

Proposition 8.2.6 For an instance $D = D^n \cup D^x$ and a monotone query $\mathcal{Q}(\bar{x})$ with $\mathcal{Q}(D) = \{\bar{a}_1, \ldots \bar{a}_n\}$, and $\bar{a_k} \in \mathcal{Q}(D)$, $\tau \in D^n$ is a vc-cause for $\bar{a_k}$ iff there is $D' \subseteq D$, with $\tau \in (D \smallsetminus D') \subseteq D^n$, that is a solution to the view-side-effect-free deletion-problem for $\bar{a_k}$.

Proof: Similar to the proof of Proposition 8.2.4. $\square$

Example 8.2.3 (ex. 1.1.1 and 8.2.1 cont.) Let's assume that $D = D^n$. We obtained S_1, S_2, S_3 and S_4 as solutions to the minimal- (and minimum-) source-side-effect deletion-problems for the view-element $\langle John, XML\rangle$. Let S be their union, i.e. $S = \{Author(John, TODS), Journal(TKDE, XML, 32), Author(John, TKDE), Journal(TODS, XML, 32)\}$.

It is clear that S contains actual causes for $\langle John, XML\rangle$. In this case, actual causes are also most responsible causes. This coincides with the results obtained in Example 1.1.1, and confirms Propositions 8.2.4 and 8.2.5.

Furthermore, since $\langle John, XML\rangle$ has no view-side-effect-free solution (cf. Example 8.1.2), it has no vc-cause, which confirms the results obtained in Example 2.2.4 and Proposition 8.2.6. □

8.3 Complexity of VC-Causality

We now investigate the complexity of the view-conditioned causality problem (cf. Definition 2.2.5). For this, we take advantage of the connection between vc-causality and view-side-effect-free deletion-problem.

First, the following result about the *vc-cause existence problem* (cf. Definition 2.2.6) is obtained from the *NP*-completeness of the view-side-effect-free deletion-problem decision problem for conjunctive views [Buneman et al., 2002, theo. 2.1] and Proposition 8.2.3.

Proposition 8.3.1 For CQs $\mathcal{Q}$, $\mathcal{VCEP}(\mathcal{Q})$ is *NP*-complete (in data).

Proof: For membership of *NP*, the following is a non-deterministic *PTIME* algorithm for $\mathcal{VCEP}$: Given D and answer $\bar{a}$ to $\mathcal{Q}$, guess a subset $\Gamma \subseteq D^n$ and a tuple $\tau \in D^n$, return *yes* if τ is a vc-cause for $\bar{a}$ with contingency set Γ; otherwise return *no*. This test can be performed in *PTIME* in the size of D.

Hardness is by the reduction from the (*NP*-hard) view-side-effect-free deletion-problem that is explicitly stated in Proposition 8.2.3. □

The next result is about deciding vc-causality (cf. Definition 2.2.5).

Proposition 8.3.2 For CQs $\mathcal{Q}$, $\mathcal{VCDP}(\mathcal{Q})$ is *NP*-complete (in data).

Proof: *Membership*: For an input $(D, \bar{a})$, non-deterministically guess $\tau \in D^n$ and $\Gamma \subseteq D^n$, with $\tau \notin \Gamma$. If τ is a vc-cause for $\bar{a}$ with contingency set Γ (which can be checked in polynomial time), return *yes*; otherwise return *no*.

Hardness: Given an instance D and $\bar{a} \in \mathcal{Q}(D)$, it is easy to see that: $(D, \bar{a}) \in \mathcal{VCEP}(\mathcal{Q})$ iff there is $\tau \in D^n$ with $(D, \bar{a}, \tau) \in \mathcal{VCDP}(\mathcal{Q})$. This immediately gives us a one-to-many reduction from $\mathcal{VCEP}(\mathcal{Q})$: $(D, \bar{a})$ is mapped to the polynomially-many inputs of the form $(D, \bar{a}, \tau)$ for $\mathcal{VCDP}(\mathcal{Q})$, with $\tau \in D^n$. The answer for $(D, \bar{a})$ is *yes* iff at least for one τ, $(D, \bar{a}, \tau)$ gets answer *yes*. This is a polynomial number of membership tests for $\mathcal{VCDP}(\mathcal{Q})$. □

In this result, NP-hardness is defined in terms of "Cook (or Turing) reductions" as opposed to many-one (or Karp) reductions [Garey et al., 2979, Goldreich et al., 2008]. NP-hardness under many-one reductions implies NP-hardness under Cook reductions, but the converse, although conjectured not to hold, is an open problem. However, for Cook reductions, it is still true that there is no efficient algorithm for an NP-hard problem, unless $P = NP$.

Finally, we settle the complexity of the vc-causality responsibility problem for conjunctive queries.

Proposition 8.3.3 For CQs $\mathcal{Q}$, $\mathcal{VRDP}(\mathcal{Q})$ is *NP*-complete (in data).

Proof: *Membership:* For an input $(D, \bar{a}, \tau, v)$, non-deterministically guess $\Gamma \subseteq D^n$, and return *yes* if τ is a vc-cause for $\bar{a}$ with contingency set Γ, and $|\Gamma| < \frac{1}{v}$. Otherwise, return *no*. The verification can be done in *PTIME* in data.

Hardness: By reduction from the $\mathcal{VCDP}$ problem, shown to be NP-complete in Proposition 8.3.2.

Map $(D, \bar{a}, \tau)$, an input for $\mathcal{VCDP}(\mathcal{Q})$, to the input $(D, \bar{a}, \tau, k)$ for $\mathcal{VRDP}(\mathcal{Q})$, where $k = \frac{1}{|D|+1}$. Clearly, $(D, \bar{a}, \tau) \in \mathcal{VCDP}(\mathcal{Q})$ iff $(D, \tau, \bar{a}, k) \in \mathcal{VRDP}(\mathcal{Q})$. This follows from the fact that $\tau \in D^n$ is an actual cause for $\bar{a}$ iff $vc\text{-}\rho_{\mathcal{Q}(\bar{a})}(\tau) \geq \frac{1}{|D|}$. □

Notice that the previous proof uses a Karp reduction, but from a problem identified as NP-hard through the use of a Cook reduction.

The next result is obtained from the $FP^{NP(log(n))}$-completeness of computing the highest responsibility associated to a query answer (i.e. the responsibility of the most responsible causes for the answer) (cf. Proposition 5.5.3).

Proposition 8.3.4 Computing the size of a solution to a minimum-source-side-effect deletion-problem is $FP^{NP(log(n))}$-hard. $\square$

Proof: By reduction from computing responsibility of a most responsible cause on the basis of Proposition 8.2.2. $\square$

All results on vc-causality in this section also hold for UCQs.

Chapter 9

Degree of Causal Contribution

The notion of causal responsibility in [Meliou et al., 2010c] intends to provide a metric to quantify the causal contribution, as a numerical degree, of a tuple to a query answer. This responsibility-based ranking is considered as one of the most important contributions of HP-causality to data management [Meliou, et al., 2011c].

Causal responsibility can be traced back to [Chockler & Halpern, 2004], where it is suggested that, for variables A and B, the degree of responsibility of A for B should be $\frac{1}{(N+1)}$, and N is the *minimum number of changes* that have to be made to obtain a situation where B counterfactually depends directly on A. If A is not a cause for B, then the degree of responsibility of A for B is 0. It has been pointed out that, although the degree of responsibility is a number between 0 and 1, it does not act like a probability [Halpern, 2015b].

In the previous chapters, a close connection between causal responsibility and other notions in data management, such as the minimum-source-side-effect deletion-problem and cardinality-based repairs, has been unveiled (see also [Salimi & Bertossi, 2015a, Salimi & Bertossi, 2015b, Cibele et al., 2016]). The underlying reason for this is that these notions are all motivated by the need to perform a minimum number of changes in a database so that a new state of the database has a desired property. Therefore, causal responsibility is indeed an important notion that can capture and unify several problems in data management.

However, in this chapter we argue in technical terms that causal responsibility as introduced in [Meliou et al., 2010c] may only partially fulfil the original intention of providing a plausible ranking for tuples in terms of their causal contributions to a query answer. More specifically, we show that in addition to the "minimum number of changes" required to make a tuple τ a counterfactual cause, there are other factors that affect our judgment for the attribution of a degree of causal contribution to τ. Actually, we develop a metric that does take into account all those other factors, matches our intuition, and enjoys interesting properties.

9.1 Necessity and Sufficiency for QA-Causality

Definition 9.1.1 For an instance $D = D^x \cup D^n$ and a Datalog query Π with $D \cup \Pi \models Ans(\bar{a})$:[1]

(a) $S \subseteq D^n$ is a *sufficient-set* for $\bar{a}$ if $S \cup D^x \cup \Pi \models Ans(\bar{a})$.

(b) $S \subseteq D^n$ is a *minimal-sufficient-set* for $\bar{a}$, if it is a sufficient-set for $\bar{a}$, but no proper subset of S is a sufficient-set for $\bar{a}$.[2]

(c) $N \subseteq D^n$ is a *necessary-set* for $\bar{a}$ if $(D^n \smallsetminus N) \cup D^x \cup \Pi \not\models Ans(\bar{a})$.

(d) $N \subseteq D^n$ is a *minimal-necessary-set* for $\bar{a}$, if it is a necessary-set for $\bar{a}$, but no proper subset of N is a necessary-set for $\bar{a}$. □

Next, we define two indices that measure the extent by which a tuple is necessary or sufficient to support a query answer.

Definition 9.1.2 For an instance $D = D^x \cup D^n$, a Datalog query Π, with $D \cup \Pi \models Ans(\bar{a})$, and a tuple $\tau \in D^n$:

(a) The *sufficiency-degree* of τ (for $\bar{a}$), denoted by $\sigma_{\Pi(\bar{a})}(\tau)$, is $\frac{1}{|S|}$, where S is a minimum-cardinality sufficient-set for $\bar{a}$ with $\tau \in S$. If τ is not contained in any of the sufficient-sets of $\bar{a}$, then $\sigma_{\Pi(\bar{a})}(\tau) = 0$.

(b) The *necessity-degree* of τ (for $\bar{a}$), denoted by $\eta_{\Pi(\bar{a})}(\tau)$, is $\frac{1}{|N|}$, where N is a minimum-cardinality necessary-set for $\bar{a}$ with $\tau \in N$. If τ is not contained in any of the necessary-sets of $\bar{a}$, then $\eta_{\Pi(\bar{a})}(\tau) = 0$. □

Example 9.1.1 Consider the instance D with relations R and T as below, and the query Π: $Ans(x) \leftarrow R(x, y), T(y)$, which is true in D. Assume all tuples are endogenous.

R	A	B
	a_1	a_4
	a_3	a_1
	a_3	a_3

T	A
	a_1
	a_2
	a_3

[1] For a CQ $\mathcal{Q}$, this statement takes the form $D \models \mathcal{Q}(\bar{a})$.

[2] Notice that when $D = D^n$, sufficient-sets of a query answer coincide with its *minimal-witnesses* as in [Buneman et al., 2001] (cf. Chapter 1).

Here, $\langle a_3 \rangle$ is an answer to the query, and has the following minimal sufficient-sets: $S_1 = \{R(a_3, a_1), T(a_3)\}$ and $S_2 = \{R(a_3, a_3), T(a_3)\}$; and the following necessary-sets: $N_1 = \{T(a_3)\}$ and $N_2 = \{R(a_3, a_1), R(a_3, a_3)\}$.

The tuples in the minimal-sufficient (necessary) sets of a_3 are $T(a_3)$, $R(a_3, a_1)$ and $R(a_3, a_3)$, which coincide with its actual causes for $\langle a_3 \rangle$.

The sufficiency-degree of each of the actual causes is $\frac{1}{2}$. However, the necessity-degree of $T(a_3)$ is 1, whereas the necessity-degree of $R(a_3, a_1)$ and $R(a_3, a_3)$ is $\frac{1}{2}$. $\square$

For Datalog queries, the collection of the subset-minimal contingency sets associated with a cause τ, cf. Equation (5.3), takes the form:

$$\begin{aligned} \mathit{Cont}(D, \Pi(\bar{a}), \tau) \quad := \quad & \{\Gamma \subseteq D^n \mid D \smallsetminus \Gamma \cup \Pi \models \mathit{Ans}(\bar{a}),\ D \smallsetminus (\Gamma \cup \{\tau\}) \cup \Pi \not\models \mathit{Ans}(\bar{a}), \\ & \text{and } \forall \Gamma'' \subsetneqq \Gamma,\ D \smallsetminus (\Gamma'' \cup \{\tau\}) \cup \Pi \models \mathit{Ans}(\bar{a})\}. \end{aligned}$$

Proposition 9.1.1 Let $D = D^x \cup D^n$, Π be a Datalog query with $D \cup \Pi \models \mathit{Ans}(\bar{a})$, and $\tau \in D^n$.

(a) $\tau \in \mathit{Causes}(D, \Pi(\bar{a}))$, with $\Gamma \in \mathit{Cont}(D, \Pi(\bar{a}), \tau)$, iff $N = \Gamma \cup \{\tau\}$ is a minimal-necessary-set for $\bar{a}$.

(b) τ belongs to some minimal-sufficient-sets for $\bar{a}$ iff it belongs to some minimal-necessary-set for $\bar{a}$.

Proof: Part (a) is straightforward. We only prove part (b). Assume $\mathcal{SF} = \{S_1, \ldots, S_m\}$ is the set of all minimal-sufficient-sets for $\bar{a}$, and there exists $S \in \mathcal{SF}$ with $\tau \in S$. Consider an arbitrary $\Gamma \subseteq D^n$ such that, for every $S_i \in \mathcal{S}$ where $S_i \neq S$, it holds $\Gamma \cap S_i \neq \emptyset$ and $\Gamma \cap S = \emptyset$. Then, $N = \Gamma \cup \{\tau\}$ is a necessary set for $\bar{a}$.

It is good enough to prove that such a Γ always exists. In fact, since all subsets of $\mathcal{SF}$ are subset-minimal, for each $S_i \in \mathcal{SF}$ with $S_i \neq \mathcal{SF}$, it holds $S_i \cap S = \emptyset$. Therefore, we can choose $\Gamma := \bigcup_1^m \{\bar{t} \mid \bar{t} \in (S \triangle S_i)\}$. The proof in the other direction is straightforward. $\square$

Proposition 9.1.1 basically states that the tuples in minimal-sufficient or minimal-necessary-sets for a query answer are the actual causes for that answer.

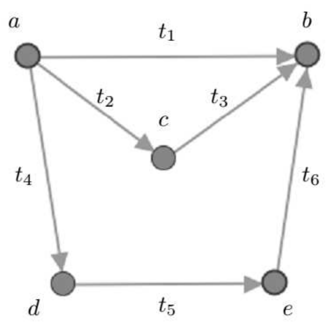

Figure 9.1: Graph G associated to instance D

Example 9.1.2 Consider the instance D with a single binary relation E as below (t_1-t_6 are tuple identifiers). Assume all tuples are endogenous.

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

Instance D can be represented as the directed graph $G(\mathcal{V}, \mathcal{E})$ in Figure 9.1, where $\mathcal{V}$ coincides with the active domain of D (i.e., the set of constants in E), and $\mathcal{E}$ contains an edge (v_1, v_2) iff $E(v_1, v_2) \in D$. The tuple identifiers are used as labels for the corresponding edges in the graph.

For simplicity, we will refer to the database tuples through their identifiers.

Consider the Datalog query Π:

$$\begin{aligned} Ans(x, y) &\leftarrow P(x, y) \\ P(x, y) &\leftarrow E(x, y) \\ P(x, y) &\leftarrow P(x, z), E(z, y). \end{aligned}$$

It collects pairs of vertices of G that are connected through a path.

It is easy to verify that $\langle a, b\rangle$ is an answer to this query, $Causes(D, \Pi(\langle s, t\rangle)) = \{t_1, t_2, t_3, t_4, t_5, t_6\}$, and the causal responsibility of each cause is $\frac{1}{3}$.

All the actual causes are equally responsible. However, it makes more sense to claim that t_1 contributes more to the answer $\langle a, b\rangle$ than the other tuples.

Answer $\langle a, b\rangle$ has the following minimal-necessary-sets: $N_1 = \{t_1, t_2, t_4\}$, $N_2 = \{t_1, t_2, t_5\}$, $N_3 = \{t_1, t_2, t_6\}$, $N_4 = \{t_1, t_3, t_4\}$, $N_5 = \{t_1, t_3, t_5\}$, $N_6 = \{t_1, t_3, t_6\}$. So,

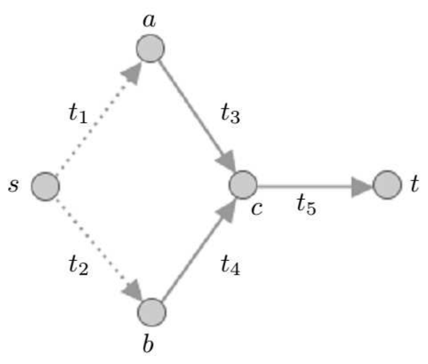

Figure 9.2: Graph G' associated to the instance D'

all the causal tuples have the same necessity-degree. The same answer has the following minimal-sufficient-sets: $S_1 = \{t_1\}$, $S_2 = \{t_2, t_3\}$, $S_3 = \{t_4, t_5, t_6\}$. So, in this case, the sufficiency-degree of t_1 is 1, higher than that of any of the other causes.

A combination of edges that forms a path between a and b forms a sufficient-set for $\langle a, b\rangle$. Moreover, a combination of edges whose removal disconnects a and b is a necessary-set for the answer. It makes sense to claim that tuples that belong to shorter paths between a and b contribute more to the answer than tuples that belong to longer paths. In other words, tuples with higher sufficiency-degree seems to contribute more to the answer. $\square$

It turns out that causal responsibility as introduced in [Meliou et al., 2010c] is a a measure of the extent by which a tuple is necessary for a query answer. A simple corollary of Proposition 9.1.1 states that causal responsibility of a tuple for a query answer coincides with its necessity-degree.

For simplicity, in the rest of this section we will concentrate mostly on boolean Datalog queries. We recall that in this case, the answer is the propositional atom ans or nothing (no answer).

Corollary 9.1.1 Let $D = D^x \cup D^n$ be an instance, and Π be a boolean Datalog query with $\Pi \cup D \models ans$. It holds: $\eta_{\Pi}(\tau) = \rho_{\Pi}(\tau)$. $\square$

Recalling the observation made in Example 9.1.2, one could propose the sufficiency-degree as the right metric to measure causal contribution. However, the next example suggests that both the sufficiency and necessity-degrees have to be taken into account.

Example 9.1.3 Consider the instance D' with a single binary relation E as below (t_1-t_5 are tuple identifiers). Assume t_1 and t_2 are exogenous.

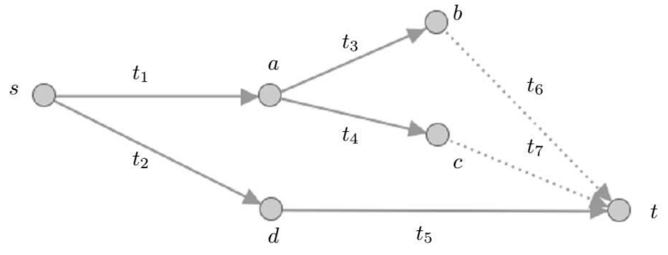

Figure 9.3: Graph G'' associated to the instance D''

E	A	B
t_1	s	a
t_2	s	b
t_3	a	c
t_4	b	c
t_5	c	t

Consider the boolean Datalog query Π:

$$\begin{array}{rcl} ans & \leftarrow & P(s,t) \\ P(x,y) & \leftarrow & E(x,y) \\ P(x,y) & \leftarrow & P(x,z), E(z,y). \end{array}$$

The query asks if there exist a path between s and t in graph G', the graph associated to instance D', shown in Figure 9.2. The dashed edges correspond to the exogenous tuples.

Clearly, $\Pi \cup D' \models ans$, and this answer (ans) has the following minimal-sufficient-sets: $S_1 = \{t_2, t_5\}$, $S_2 = \{t_4, t_5\}$, and the following minimal-necessary-sets: $N_1 = \{t_5\}$, $N_2 = \{t_2, t_4\}$.

We obtain: $\eta_\Pi(t_5) = 1$, $\eta_\Pi(t_2) = \eta_\Pi(t_4) = \frac{1}{2}$; but $\sigma_\Pi(t_2) = \sigma_\Pi(t_4) = \sigma_\Pi(t_5) = \frac{1}{2}$.

Intuitively, it makes sense to claim that t_5 contributes more to the result, because the answer is counterfactually depends on t_5 (no need for an accompanying contingency set). We can see that in this case, in contrast with Example 9.1.2, ranking causal tuples on the basis of their necessity-degrees is inline with our intuition that they contribute more to the query result. □

The next example shows that considering both sufficiency and necessity-degrees may not capture the intuition about which tuple contributes more as cause.

Example 9.1.4 (ex. 9.1.3 cont.) Consider the same query Π, but now the instance D'' as below. Assume t_6 and t_7 are exogenous tuples.

E	A	B
t_1	s	a
t_2	s	b
t_3	a	c
t_4	a	d
t_5	b	τ
t_6	c	τ
t_7	d	τ

It is clear that $\Pi \cup D'' \models ans$. The answer ans has the following minimal-necessary and sufficient-sets: $S_1 = \{t_1, t_3\}$, $S_2 = \{t_1, t_4\}$, $S_3 = \{t_2, t_5\}$, $N_1 = \{t_1, t_2\}$, $N_2 = \{t_1, t_5\}$, $N_3 = \{t_2, t_3, t_4\}$; and $N_4 = \{t_5, t_3, t_4\}$. Here, $\sigma_{\Pi}(t_1) = \sigma_{\Pi}(t_2) = \frac{1}{2}$, and $\eta_{\Pi}(t_1) = \eta_{\Pi}(t_2) = \frac{1}{2}$.

If the degree of causal contribution of the tuple to the answer can be captured by considering only the sufficiency and necessary-degrees, we would conclude that, for example, t_1 and t_2 equally contribute to the result. However, looking at the graph representation of D'' in Figure 9.3, it is reasonable to claim that t_1 contributes more to the result than t_2. This is because, t_1 contributes to two paths between s and t, whereas t_2 contributes to only one path. In other words, t_1 is contained in more sufficient-sets than t_2.

We observe that our intuition about the difference in causal contribution between t_1 and t_2 to the answer is captured by the difference between the number of necessary-sets of the answer that these tuples belong to. To see this, notice that t_1 and t_2 are contained in two minimal-necessary sets. More specifically, t_1 belongs to N_1 and N_2; and t_2 to N_1 and N_3. Now, since each superset of a minimal-necessary-set is a necessary-set, and in this case, $|N_1| < |N_3|$, it turns out that N_1 has more supersets than N_3. That is, t_1 is contained in more necessary-sets for the answer than t_2. Actually, t_1 is contained in 18 necessary-sets for the answer, whereas t_2 is contained only in 10 necessary-sets. $\square$

9.2 Degree of Causal Contribution

On the basis of the previous discussion and examples, our intention is to use the number of necessity-sets as the right candidate to measure causal contribution. Specifically, we propose the following metric to measure the causal contribution of a tuple to a query answer.

Definition 9.2.1 For an instance D, a Datalog query Π with $\Pi \cup D \models Ans(\bar{a})$, and a tuple $\tau \in D^n$, the *degree of causal contribution* of τ to answer $\bar{a}$ is:

$$\mathcal{C}_{\Pi(\bar{a})}(\tau) \;\; = \;\; \frac{\#N(\tau)}{\Sigma_{\tau' \in D} \;\; \#N(\tau')}, \tag{9.1}$$

where, $\#N(\tau)$ is the number of necessary-sets of $\bar{a}$ to which τ belongs. □

Notice that exogenous and non-cause tuples contribute with 0 to the sum in the denominator. Also notice that in this measure we are *not* restricting the necessary-sets involved in the computation of $\#N(\tau)$ to be minimal. We illustrate the usefulness of this metric by applying it to several examples.

Example 9.2.1 (ex. 9.1.2 cont.) The table below shows the degrees of causal contribution for tuples in D to the answer $\langle a, b\rangle$. In this case, $\#N(t_1) = 21$, $\#N(t_2) = \#N(t_3) = 7$, $\#N(t_4) = \#N(t_5) = \#N(t_6) = 3$.

$\mathcal{C}$	$\langle a, b\rangle$
0.48	t_1
0.16	t_2
0.16	t_3
0.07	t_4
0.07	t_5
0.07	t_6

As already discussed, a plausible ranking of tuples in terms of their causal contribution to the answer should put t_1 at the top; t_2 and t_3 in the middle; and t_4, t_5 and t_6 at the bottom. This is perfectly captured by the ranking on the left-hand-side. □

Example 9.2.2 (ex. 9.1.4 cont.) The table below shows the degrees of causal contribution of the tuples to the query answer. Here, $\#N(t_1) = 9$, $\#N(t_2) = \#N(t_5) = 5$, $\#N(t_3) = \#N(t_4) = \#N(t_6) = 3$.

$\mathcal{C}$	$\langle s, t\rangle$
0.36	t_1
0.20	t_2
0.20	t_5
0.12	t_3
0.12	t_4

Again, the degrees of causal contribution match our intuition. □

Example 9.2.3 (ex. 1.1.4 and 1.5.1 cont.) Consider again the answer *Musical* to the query $\mathcal{Q}$ in (1.2). The degrees of causal contribution of the actual causes of this answer are presented below.

$\mathcal{C}$	*Musical*
0.28	*Director(23468, Humphrey, Burton)*
0.22	*Director(23456, David, Burton)*
0.12	*Director(23488, Tim, Burton)*
0.12	*Movie(526338, "Sweeney Todd", 2007)*
0.07	*Movie(359516, "Let's Fall in Love", 1933)*
0.07	*Movie(565577, "The Melody Lingers On", 1935)*
0.04	*Movie(6539, "Candide", 1989)*
0.04	*Movie(173629, "Flight", 1999)*
0.04	*Movie(389987, "Manon Lescaut", 1997)*

As we already argued in Example 1.5.1, the tuple for "Humphrey Burton" contributes more to the genre "Musical" (the query answer) than those for "David Burton" and "Tim Burton". Our ranking shows a match with our intuition.

Furthermore, this ranking reflects the fact that these director tuples contribute more to the category "Musical" than their movie tuples. The only exception is "Tim Burton", who equally contributes to this category, with his movie "Sweeney Todd". This is because he only has one musical movie. □

The next proposition shows that the degree of causality enjoys two interesting properties.

Proposition 9.2.1 Given a database $D = D^n \cup D^x$, a Datalog query Π with $\Pi \cup D \models Ans(\bar{a})$, the degree of causal contribution has the following properties:

$\mathcal{P}_1$: For any tuple $\tau \in D^n$, $0 \leq \mathcal{C}_{\Pi(\bar{a})}(\tau) \leq 1$. Furthermore, $\mathcal{C}_{\Pi(\bar{a})}(\tau) = 0$ when τ is not an actual cause for $\bar{a}$; and $\mathcal{C}_{\Pi(\bar{a})}(\tau) = 1$ when τ is a unique actual cause for $\bar{a}$. (Actually, this happens when $\sigma_{\Pi(\bar{a})}(\tau) = \eta_{\Pi(\bar{a})}(\tau) = 1$).

$\mathcal{P}_2$: $\sum_{\tau \in D} \mathcal{C}_{\Pi(\bar{a})}(\tau) = 1$, that is the degree of causality metric assigns to each tuple $\tau \in D$ a value that describes its relative share as a cause to answer $\bar{a}$. (In particular, if a tuple has a degree of causality of 1, it is the only actual cause.)

Proof: To prove the last part of $\mathcal{P}_1$ (the first in immediate, so as that of $\mathcal{P}_2$), suppose τ is a unique actual cause for $\bar{a}$. Then, for all $\bar{t} \in D$ with $\bar{t} \neq \tau$, it holds $\#N(\tau) = 0$. Actually, this is a simple corollary of Proposition 9.1.1. Therefore, $\mathcal{C}_{\Pi(\bar{a})}(\tau) = 1$. □

The last three examples clearly illustrate these properties. Actually, properties $\mathcal{P}_1$ and $\mathcal{P}_2$ in Proposition 9.2.1 could be used as *postulates* that any sensitive enough measure of causal contribution should satisfy. As a matter of fact, causal responsibility (as defined in [Meliou et al., 2010c]) does not satisfy them, as Example 9.2.4 below shows.

In fact, a basic fact of causal responsibility is that a tuple τ makes the maximum causal contribution to an answer $\bar{a}$ when it is a counterfactual cause for $\bar{a}$. Although counterfactual dependency is considered to be the *strongest form of causal dependency* between a tuple and an answer, the following example shows that a counterfactual cause may not necessarily have 1 as degree of causal contribution.

Example 9.2.4 (ex. 9.1.3 cont.)

(a) Consider instance $I = \{E(s,t)\}$. It holds: $\Pi \cup I \models \mathit{ans}$. This answer has the counterfactual cause $E(s,t)$, with responsibility 1. Moreover, $\eta_{_\Pi}(E(s,t)) = 1 = \sigma_{_\Pi}(E(s,t)) = 1$; so that $E(s,t)$ makes the maximum causal contribution to the answer. Furthermore, $\mathcal{C}_{\Pi(\bar{a})}(E(s,a)) = 1$.

(b) Consider instance $I' = \{E(s,a),\ E(a,t)\}$. The answer of the query Π from I' is also ans. Now, both $E(s,a)$ and $E(a,t)$ are counterfactual causes for the result, i.e. $\eta_{_\Pi}(E(s,a)) = \eta_{_\Pi}(E(a,t)) = 1$. However, in this case the combination of the tuples forms the only sufficient-set for the answer, i.e. $\sigma_{_\Pi}(E(s,a)) = \sigma_{_\Pi}(E(a,t)) = \frac{1}{2}$.

We can see that the total contribution to the answer is shared between the both tuples. Our metric perfectly captures the distinction between this case and (a). This is because, we obtain $\mathcal{C}_{_\Pi}(E(s,a)) = \mathcal{C}_{_\Pi}(E(a,t)) = \frac{1}{2}$. However, causal responsibility (in the sense of [Meliou et al., 2010c]) assigns 1 to both of these tuple, because they are counterfactual causes for the answer. Also notice that the sum of these tuples' responsibilities is 2. $\square$

We should point out that the properties $\mathcal{P}_1$ and $\mathcal{P}_2$ do not also hold for the sufficiency- and necessity-degrees. For the latter this is implied by the connection between necessity-degree and causal responsibility (cf. Corollary 9.1.1). For the former, consider Example 9.1.3, where we obtained that t_1 has sufficiency-degree 1. This collides with $\mathcal{P}_2$, because t_1 is not the single actual cause for the answer. The same example shows that the sum of the sufficiency-degrees is not equal to 1.

9.3 Discussion on the Degree of Causal Contribution

A key observation, which helps us justify the number necessary-sets as the basic component in the definition of degree of causal contribution, ii that it is mathematically related to the other factors we have considered in this chapter, among them the sufficiency- and necessity-degrees, and the number of sufficient-sets that contain a cause. In fact, the necessary-sets for an answer $\bar{a}$ are supersets of its minimal-necessary-sets. Now, the minimal-necessary-sets for $\bar{a}$ are essentially the *subset-minimal hitting-sets* of the class of all minimal-sufficient-sets for $\bar{a}$. Actually, the following result can be obtained from Proposition 9.1.1 and the results in Section 5.3.2.2.

Proposition 9.3.1 For an instance $D = D^x \cup D^n$, a Datalog query Π with $D \cup \Pi \models Ans(\bar{a})$, $N \subseteq D^n$ is a minimal-necessary-set for $\bar{a}$ iff N is a subset-minimal hitting-set for $\mathcal{SF}$, the class of all the minimal-sufficient-sets for $\bar{a}$.

Proof: Suppose N is a subset-minimal hitting-set for the class $\mathcal{SF}$. Then from the definition of a hitting-set, for all $S \in \mathcal{SF}$ it holds $N \cap S \neq \emptyset$. Due to the subset-minimality of the sufficient-sets, it holds $(D^n \smallsetminus N) \cup D^x \cup \Pi \not\models Ans(\bar{a})$. Therefore, N is a minimal-necessary-set for $\bar{a}$. A similar argument applies in the other direction. $\square$

From Proposition 9.3.1, we obtain that, for a cause τ for an answer $\bar{a}$, $\#N(\tau)$ is related to the number of minimal-hitting sets for the class of minimal-sufficient-sets for $\bar{a}$ that contain τ. Counting hitting-sets is a problem that has been investigated. For example, in [Peter & Molokov, 2009, lemma 2] it is shown (in essence) that the number of different minimal-hitting sets of size k for a collection of sets that contain a fixed element e is a function of the size of the maximum-cardinality set in the collection and the size of the minimum-cardinality hitting-sets that contain e.

Recall from Proposition 9.1.1 that a cause τ for a query answer, together with any of its contingency sets, forms a necessary-set for the query answer. Therefore, $\#N(\tau)$ coincides with the number of contingency sets associated to τ. That is, the degree of causal contribution (cf. Definition 9.2.1) can be seen as the (normalized or relative) frequency with which a tuple τ is a counterfactual cause for $\bar{a}$ across all possible contingency sets. This interpretation reflects a close connection between the *degree of causal contribution* (DCC) introduced in this chapter and the *degree of causation* (DOC) introduced in [Braham & Van Hee, 2009], which is based on the concept of *Necessary Element of a Sufficient Set* (NESS) test (cf. Section 3.2). In fact, the authors argue that in order to define a degree of causation, one should focus on the relative frequency with which an action satisfies the NESS test.

Intuitively, the DOC is the normalized frequency by which an event is a "critically sufficient condition" for an observation. More specifically, the authors consider the full range of sufficient conditions for an observation, in which an event is necessary. This intuition can be adapted in the context of QA-causality as follows: the DOC of a tuple τ to a query answer $\bar{a}$ is the number of subinstances D' of the database D where $\bar{a}$ is still an answer and τ is a counterfactual cause for $\bar{a}$ in D'. It is not difficult to see that this numerical value coincides with the number of contingency sets associated to τ, that is $\#N(\tau)$.

In contrast to our DCC, the DOC proposed in [Braham & Van Hee, 2009] does not emerge in relation to the notion of causal responsibility by [Chockler & Halpern, 2004]. In fact, the authors developed their metric merely on the basis of the intuition that, in order to define a degree of causation, one should focus on the relative frequency with which an action satisfies the NESS test. Furthermore, no mathematical justification or motivating examples are provided to support their intuition.

We have argued in precise terms, and illustrating with several examples, the factors that affect our judgment of the degree of causal contribution of a tuple to a query answer. Namely, the sufficiency- and necessity-degrees and the numbers of necessary- and sufficient-sets that a tuple belongs to. Furthermore, we discussed that, among the mentioned factors, the number of necessary-sets a tuple is contained in depends on the rest of the factors, and is the right metric to measure degree of causal contribution of a tuple. Although this discussion and model is restricted to QA-causality, the underlying idea could be extended to general setting of HP-causality.

Considering that each contingency set associated to a tuple is basically a way to make it a counterfactual case, the intuition behind [Braham & Van Hee, 2009] coincides with that of [Zultan et al., 2013], according to which, in order to ascribe a degree of causation, people take into account not only the number of changes required to make A a counterfactual cause for B, but also the number of ways to reach a situation where B counterfactually depends on A.

In this chapter we have proposed the notion of degree of causal contribution, and provided a precise definition. However, there are many open problems about this new measure that have to be mathematically investigated. Among many others, we see the following as immediate challenges for future work: (a) A complexity analysis of the computational problems related to the degree of causal contribution (we conjecture that it is intractable). (b) The identification of classes of queries for which computing this metric is tractable (c) The extension of our proposed metric to the general HP-model, as an alternative to the notion of causal responsibility. (d) The investigation of the connection between this metric and other related problems, e.g. the model-counting problem. (e) The identification of a precise mathematical connection between necessary-sets and other factors we have considered in this chapter, e.g. sufficiency-and necessity-degrees, and the number of sufficient-sets that contain a cause.

Chapter 10

Discussion and Conclusions

10.1 Issues with Causality and Responsibility

10.1.1 QA-causality vs. learning causal information from data

It is important to make the distinction between learning causality from data (i.e., extracting causal information from data) and the notion of QA-causality as defined and investigated in [Meliou et al., 2010c], and used in this work.

It is well understood that to address causal questions (or to assess causal assumptions) using data, some knowledge of the *data-generating process* is required [Pearl, 2010]. More specifically, causality can only be established in a controlled physical experiment where one changes one single variable, while keeping all others unchanged (performing an intervention), and observes changes in the output (effects of the intervention). This requires knowledge and control of the data-generating process, which goes beyond the data alone.

The key building blocks of relational databases are *entities* and *relations (or predicates)*. Entities represent objects that exist in the domain of a database and relations represent properties of these objects or relationships among them. A query form a relational database instance can be seen as a Datalog program (cf. Section 2.2), provided that the program has a distinguished output predicate that collects the answers. From this perspective, a query may be seen as a causal relation between the predicates in the body of the rules and the output predicate. However, this causal relation may not represent causal information about the domain objects that are represented by these predicates (see Example 10.1.1).

Having considered a query as a causal relation, it is possible to reason about causes for the query answers. The query provides the required knowledge to perform interventions, assess the effects of the interventions and establish actual causation for the query answers. In this direction, QA-causes merely convey information about the formation of query answers as mathematical objects rather than the domain objects they represent.

Example 10.1.1 Consider the instance D with a single relation $Sells$ as below. This relation contains some information on beer menus in Brussels' bars (t_1 and t_6 are tuple identifiers).

Sells	*Bar*	*Beer*	*Price*
t_1	*Hard Rock Cafe*	*Budweiser*	€4.45
t_2	*Hard Rock Cafe*	*Delirium*	€3.58
t_3	*Hard Rock Cafe*	*Duvel*	€2.58
t_4	*Play Boy Bar*	*Corona*	€2.65
t_5	*Play Boy Bar*	*Budweiser*	€4.75
t_6	*Play Boy Bar*	*Duvel*	€2.18

Suppose a beer is known to be "cheap" in Brussels if there are at least two bars that sell it for under €3. This is captured by the CQ query *Cheap* as below:[1]

$$\begin{aligned}\mathit{Cheap}(\mathit{Beer}) :\quad & \exists \mathit{Bar}_1\, \exists \mathit{Bar}_2\, \exists \mathit{Price}_1 \exists \mathit{Price}_2 (\mathit{Sells}(\mathit{Bar}_1, \mathit{Beer}, \mathit{Price}_1) \wedge \\ & \mathit{Sells}(\mathit{Bar}_2, \mathit{Beer}, \mathit{Price}_2) \wedge\ (\mathit{Price}_1 < 3) \wedge \\ & (\mathit{Price}_2 < 3) \wedge (\mathit{Bar}_1 \neq \mathit{Bar}_2)), \end{aligned} \tag{10.1}$$

which has only one answer, *Duvel*. It is clear that t_3 and t_6 are counterfactual causes for this answer. However, t_3 and t_6 should not be interpreted as causes for "Duvel" being a cheap beer. The fact that the two bars offer this beer for less than €3, normally has noting to do with the beer price. That is, the query in (10.1) represents associational information in the domain. Nevertheless, t_3 and t_6 can be seen as causes for "Duvel" to be among the query answers (i.e., they are causes for the observed associational fact in the domain). □

Consequently, extrapolating QA-causes as causal information about the domain objects is only justified under the assumption that the query represents causal knowledge about these objects. Scenarios that we dealt with in this work e.g., Examples 1.1.1 and 1.1.4 are as such. This confusion arises frequently in QA-causality literature e.g., [Meliou et al., 2010c, Meliou et al., 2010b, Roy & Suciu, 2014, Meliou et al., 2011b].

[1]The definition of causality in [Meliou et al., 2010c] is applicable to monotone queries in general. Since CQs with built-ins preserve monotonicity, the definition of causality can be applied to them. However, built-ins are used in this example for illustration purposes and are not the focus of this work.

10.1.2 Objections to causality

Causality as introduced in [Halpern & Pearl, 2005], aka. HP-causality, is the basis for the notion of causality in [Meliou et al., 2010c]. HP-causality has been the object of some criticism [Halpern, 2014], which is justified in some (more complex, non-relational) settings, specially due to the presence of different kinds of *logical variables* (or lack thereof) and the lack of a generic and generally accepted definition for the notion of contingency.

We could say that the efforts in [Halpern, 2014, Halpern, 2015a] to modify the original HP-definition of causality are about considering more appropriate restrictions on contingencies. Since in some cases the original HP-model does not provide intuitive results regarding causality, the modifications avoid this by recognizing some contingencies as "unreasonable" or "farfetched".

In our context the objections do not apply: variables just say that a certain tuple belongs to the instance (or not); and for relational databases the closed-world assumption applies. In [Halpern, 2014, Halpern, 2015a], the definition of HP-causality is slightly modified. In our setting, this modified definition does not change contingency sets and actual causes or their properties.

10.1.3 Issues with degree of responsibility

Degree of responsibility as introduced in [Chockler & Halpern, 2004] is the basis for the notion of responsibility in [Meliou et al., 2010c]. In [Gerstenberg et al., 2010] it has been argued that people use something similar to the intuition behind the notion of degree of responsibility to ascribe responsibility. However, more recently it has been discussed that people take into account not only the number of changes required to make A a counterfactual cause for B, but also the number of ways to reach a situation where B counterfactually depends directly on A [Zultan et al., 2013].

In [Halpern, 2015b], it has been argued that while causal responsibility in [Chockler & Halpern, 2004] could capture some reasonable intuitions, alternative definitions might be more appropriate in some applications. In [Braham & Van Hee, 2009], it has been argued that the notion of responsibility as defined in [Chockler & Halpern, 2004] does not determine the share of an action in bringing about an outcome, but only "the extent to which there are other causes". They propose an alternative metric called "degree of causality", on the basis of the notion of NESS test. However, no clear distinction between the two approaches has been established.

The proposed metric of degree of causal contribution in this work is an attempt to provide a more intuitive and informative alternative for causal responsibility in the context of QA-causality in databases. In this work, we have made a clear distinction between our proposal and that of [Meliou et al., 2010c].

10.2 Conclusions and Future Work

10.2.1 Causality, database repair and consistent-based diagnosis

In Chapters 5 and 6, we have unveiled and formalized some first interesting relationships between causality in databases, database repairs, and consistency-based diagnosis. These connections allow us to apply results and techniques developed for each of them to the others. This is particularly beneficial for causality in databases, where still a limited number of results and techniques have been obtained or developed.

The connections we established here inspired complexity results for causality, e.g. Theorems 5.5.2 and 5.5.3, and were used to prove them. We appealed to several non-trivial results (and the proofs thereof) about repairs/CQA obtained in [Lopatenko & Bertossi, 2007]. It is also the case that the well-established hitting-set approach to diagnosis inspired a similar approach to causal responsibility, which in its turn allowed us to obtain results about its fixed-parameter tractability. It is also the case that diagnostic reasoning, as a form of non-monotonic reasoning, can provide a solid foundation for causality in databases and query answer explanation, in general [Cheney et al., 2009b, Cheney et al., 2011].

The characterization of causes in terms of repairs opens up the possibility of introducing various notions of causes based on different repair semantics. Following this idea, in Chapter 6, we define notions of preferred causes; in particular one based on prioritized repairs [Staworko et al., 2012]. We also propose a finer-granularity approach to causality, at the attribute level rather than at the tuple level, that is based on interventions that are repair actions that replace attribute values by null values.

10.2.1.1 ASP specification of causes

S-repairs can be specified by means of *answer set programs* (ASPs) [Arenas et al., 2003a, Barcelo et al., 2003], and C-repairs too, with the use of weak program constraints [Arenas et al., 2003a]. This should allow for the introduction of ASPs in the context of causality, for specification and reasoning. There are also ASP-based specifications of diagnosis [Eiter et al., 1999] that could be brought into a more complete picture.

10.2.1.2 Endogenous repairs

As discussed in Chapter 6, the partition of a database into endogenous and exogenous tuples may also be of interest in the context of repairs. We may prefer endogenous repairs that change (delete in this case) only endogenous tuples. However, if there are no endogenous tuples, a preference condition could be imposed on repairs, keeping those that change exogenous tuples the least. This is something to explore.

As a further extension, it could be possible to assume that combinations of (only) exogenous tuples never violate the integrity constraints, which could be checked at upload time. In this sense, there would be a part of the database that is considered to be consistent, while the other is subject to possible repairs. For somehow related research, see [Greco et al., 2014].

10.2.1.3 Causes and functional dependencies, and beyond

Functional dependencies are DCs with conjunctive violation views with inequality, and are still monotonic. There is much research on repairs and consistent query answering for functional dependencies, and more complex integrity constraints [Bertossi, 2011]. In causality, mostly CQs without built-ins have been considered. The repair connection could be exploited to obtain results for causality and CQs with inequality, and also other classes of queries.

10.2.2 Causality, abductive diagnosis and the view-update problem

In Chapters 7 and 8, we have related QA-causality to abductive diagnosis and the view-update problem, respectively. The established connections between QA-causality, Datalog abduction, and the delete-propagation problems allow us to adopt and adapt established results for some of them for/to the others. In this way, we obtain some new complexity results for QA-causality.

10.2.2.1 Causality and integrity constraints

As pointed out in [Meliou et al., 2010b], the notion of QA-causality must be revised in the presence of integrity constraints (ICs). Consider the query $\mathcal{Q}(x) : \exists z \exists u \ (R(x,y,z) \wedge S(x,y,u))$, and the referential constraint $\forall x \forall y \forall z (R(x,y,z) \rightarrow \exists u S(x,y,u))$. Under the assumption that this constraint is satisfied, $\mathcal{Q}(x)$ is equivalent to $\mathcal{Q}'(x) : \exists y \exists z R(x,y,z)$, yet computing causality and responsibility for these queries may yield different results.

The view-update problem has been studied in relation to ICs in [Kimelfeld, 2012a, Cong et al., 2006]. The specification of causality in terms the view-update problem in this work should allow for the introduction of ICs in the context of causality.

10.2.3 Database repairs, abduction and view-update problem

We point out that database repairs are related to the view-update problem. Actually, *answer set programs* (ASPs) [Brewka et al., 2011] for database repairs [Bertossi, 2011] implicity repair the database by updating conjunctive combinations of intentional, annotated predicates. Those logical combinations -views after all- capture violations of integrity constraints in the original database or along the (implicitly iterative) repair process (a reason for the use of annotations). This should allow for the introduction of ASPs in the context of causality, for specification and reasoning. There are also ASP-based specifications of diagnosis [Eiter et al., 1999] that could be brought into a more complete picture.

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

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

10.2.4 Causality and reverse data management

The results obtained in this work and previously reported in [Bertossi & Salimi, 2014, Salimi & Bertossi, 2014, Salimi & Bertossi, 2015a, Salimi & Bertossi, 2015b, Salimi & Bertossi, 2015c] suggest that causal reasoning is a central activity in the *reverse data management problems* identified in [Meliou et al., 2011a]. This category has been defined based on the authors' observation that these problems invert a data transformation process, in order to reason diagnostically back about an observation. Our work is the first to study these problems from the perspective of causality.

In fact, we believe that causality can provide a unifying foundation for reverse data management problems. Having a unifying foundation for these problems opens the playground to adopt (and possibly adapt) the established results from one area to the others. It also opens up the possibility of lifting the established concepts and results from causality to reverse data management.

10.2.5 Degree of causal contribution

In Chapter 9 we argued in technical terms that causal responsibility as introduced in [Meliou et al., 2010c] may only partially fulfil the original intention of providing a plausible ranking for tuples in terms of their causal contributions to a query answer. More specifically, we show that in addition to the "minimum number of changes" required to make a tuple τ a counterfactual cause, there are other factors that affect our judgment for the attribution of a degree of causal contribution to τ. Actually, we develop a metric that does take into account all those other factors, matches our intuition, and enjoys interesting properties.

However, there are many open problems about this new measure that have to be mathematically investigated. Among many others, we see the following as immediate challenges for future work: (a) A complexity analysis of the computational problems related to the degree of causal contribution (we conjecture that it is intractable). (b) The identification of classes of queries for which computing this metric is tractable (c) The extension of our proposed metric to the general HP-model, as an alternative to the notion of causal responsibility. (d) The investigation of the connection between this metric and other related problems, e.g. the model-counting problem. (e) The identification of a precise mathematical connection between necessary-sets and other factors we have considered in this chapter, e.g. sufficiency-and necessity-degrees, and the number of sufficient-sets that contain a cause.

We observe a close connection between the degree of causal contribution, as introduced in this work, and the indices that have been used as measures of degree of causation in [Braham & Van Hee, 2009]. However, our justification comes from a different perspective.

Bibliography

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

[Afrati & Kolaitis, 2009] Afrati, F. and Kolaitis, P. Repair Checking in Inconsistent Databases: Algorithms and Complexity. *Proc. International Conference on Database Theory (ICDT)* , 2009, pp. 31-41.

[Arenas et al., 1999] Arenas, M., Bertossi, L. and Chomicki, J. Consistent Query Answers in Inconsistent Databases. *Proc. of the ACM Symposium on Principles of Database Systems (PODS)*, 1999, pp. 68-79.

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

[Arenas et al., 2003b] Arenas, M., Bertossi, L., Chomicki, J., He, X., Raghavan, V. and Spinrad, J. Scalar Aggregation in Inconsistent Databases. *Theoretical Computer Science*, 2003, 296:405-434.

[Arenas et al., 2003b] Arenas, M., Barcelo, P., Libkin, L. and Murlak, F. *Relational and XML Data Exchange*. Morgan & Claypool, Synthesis Lectures on Data Management, 2010.

[Arenas et al., 2010] Arenas, M., Prez, J., Reutter, J. and Riveros, C. Composition and Inversion of Schema Mappings. *Proc. of the ACM SIGMOD International Conference on Management of Data*, 2010, 38(3):17-28.

[Arieli et al., 2004] Arieli, O., Denecker, M., Van Nuffelen, B. and Bruynooghe, M. Coherent Integration of Databases by Abductive Logic Programming. *Journal of Artificial Intelligence Research (JAIR)*, 2004, 21:245-286.

[Barcelo et al., 2003] Barcelo, P., Bertossi, L. and Bravo, L. Characterizing and Computing Semantically Correct Answers from Databases with Annotated Logic and Answer Sets. In *Semantics of Databases*, Springer LNCS 2582, 2003, pp. 1-27.

[Bergman et al., 2015] Bergman, M., Milo, T., Novgorodov, S. and Tan, W. C. QOCO: A Query Oriented Data Cleaning System with Oracles. *Proc. of the VLDB Endowment (PVLDB)*, 2015, pp. 8(12):1900-1911.

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

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

[Bertossi & Salimi, 2014] Bertossi, L. and Salimi, B. Unifying Causality, Diagnosis, Repairs and View-Updates in Databases. *Presented at the First International Workshop on Big Uncertain Data (BUDA)*, 2014, Posted at: arXiv:1405.4228 [cs.DB].

[Brankovic & Fernau, 2012] Brankovic, L. and Fernau, H. Parameterized Approximation Algorithms for Hitting Set. *Approximation and Online Algorithms*, 2012, Springer LNCS 7164, pp. 63-76.

[Braham & Van Hee, 2009] Braham, M. and Van Hees, M. Degrees of Causation. *Erkenntnis*, 2009, 71:323-344.

[Bravo et al., 2006] Bravo, L. and Bertossi, L. Semantically Correct Query Answers in the Presence of Null Values. *Proc. EDBT Workshop on Inconsistency and Incompleteness in Databases (IIDB)*, Springer LNCS 4254, 2006, pp. 336-357.

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

[Borgida et al., 2008] Borgida, A., Calvanese, D. and Rodriguez-Muro, M. Explanation in DL-Lite. *Proc. DL Workshop*, CEUR-WS Proc. Vol-353, 2008.

[Buneman et al., 2001] Buneman, P., Khanna, S. and Tan, W. C. Why and Where: A Characterization of Data Provenance. *Proc. International Conference on Database Theory (ICDT)* , 2001, pp. 316-330.

[Buneman et al., 2002] Buneman, P., Khanna, S. and Tan, W. C. On Propagation of Deletions and Annotations Through Views. *Proc. of the ACM Symposium on Principles of Database Systems (PODS)*, 2002, pp. 150-158.

[Buneman & Tan, 2007] Buneman, P. and Tan, W. C. Provenance in Databases. *Proc. of the ACM SIGMOD International Conference on Management of Data*, 2007, pp. 1171-1173.

[ten Cate et al., 2015] ten Cate, B., Civili, C., Sherkhonov, E. and Tan, W. C. High-Level Why-Not Explanations using Ontologies. *Proc. of the ACM Symposium on Principles of Database Systems (PODS)*, 2015.

[Ceri et al., 1989] Ceri, S., Gottlob, G. and Tanca, L. *Logic Programming and Databases*. Springer, 1989.

[Calvanese et al., 2013] Calvanese, D., Ortiz, M., Simkus, M. and Stefanoni, G. Reasoning about Explanations for Negative Query Answers in DL-Lite. *Journal of Artificial Intelligence Research (JAIR)*, 2013, 48:635-669.

[Cibele et al., 2016] Cibele, F., Gatterbauer, W., Immerman, N. and Meliou A. A Characterization of the Complexity of Resilience and Responsibility for Conjunctive Queries. *Proc. of the VLDB Endowment (PVLDB)*, 2016, 9(3).

[Chapman & Jagadish, 2009] Chapman, A. and Jagadish, H. V. Why Not? *Proc. of the ACM SIGMOD International Conference on Management of Data*, 2009, pp. 523-534.

[Cheney, 2010] Cheney, J. Causality and the Semantics of Provenance. *Proc. of the Workshop on Developments in Computational Models*, 2010, pp. 63-74.

[Cheney et al., 2009a] Cheney, J., Chiticariu, L. and Tan, W. C. Provenance in Databases: Why, How, And Where. *Foundations and Trends in Databases*, 2009, 1(4):379-474.

[Cheney et al., 2009b] Cheney, J., Chong, S., Foster, N., Seltzer, M. I. and Vansummeren, S. Provenance: A Future History. *Proc. OOPSLA Companion*, 2009, pp. 957-964.

[Cheney et al., 2011] Cheney, J. Is Provenance Logical? *Proc. of the International Workshop on Logic in Databases (LID)*, 2011, pp. 2-6.

[Chomicki & Marcinkowski, 2005] Chomicki, J. and Marcinkowski, J. Minimal-Change Integrity Maintenance Using Tuple Deletions. *Information and Computation*, 2005, 197(1&2):90-121.

[Chockler & Halpern, 2004] Chockler, H. and Halpern, J. Y. Responsibility and Blame: A Structural-Model Approach. *Journal of Artificial Intelligence Research (JAIR)*, 2004, 22:93-115.

[Cong et al., 2006] Cong, G., Fan, W. and Geerts, F. Annotation Propagation Revisited for Key Preserving Views. *Proc. Information and Knowledge Management (CIKM)*, 2006, pp. 632-641.

[Console et al., 1991a] Console, L., Theseider-Dupre, D and Torasso, P. On the Relationship between Abduction and Deduction. *Journal of Logic and Computation*, 1991, 1(5):661-690.

[Console et al., 1991b] Console, L. and Torasso, P. A Spectrum of Logical Definitions of Model-Based Diagnosis. *Computational Intelligence*, 1991, 7:133-141.

[Console et al., 1995] Console, L., Sapino, M. L. and Theseider-Dupre, D. The Role of Abduction in Database View Updating. *Journal of Intelligent Information Systems*, 1995, 4(3):261-280.

[Cui et al., 2000] Cui, Y., Widom, J. and Wiener, J. L. Tracing the Lineage of View Data in a Warehousing Environment. *ACM Transactions on Database Systems (TODS)*, 2000, 25(2):179-227.

[Cui et al., 2001] Cui, Y. and Widom, J. Run-time Translation of View Tuple Deletions Using Data Lineage. Technical Report, Stanford University, 2001. http://dbpubs.stanford.edu:8090/pub/2001-24

[Das et al., 2011] Das, M., Amer-Yahia, S., Das, G. and Yu, C. Mri: Meaningful Interpretations of Collaborative Ratings. *Proc. of the VLDB Endowment (PVLDB)*, 2011, 4(11):1063-1074.

[Debosschere et al., 2015] Debosschere, M. and Geerts, F. Cell-Based Causality for Data Repairs. *Proc. Theory and Practice of Provenance (TaPP)*, 2015.

[Denecker et al., 2002] Denecker, M. and Kakas A. C. Abduction in Logic Programming. *Computational Logic: Logic Programming and Beyond*, Springer LNCS 2407, 2002, pp. 402-436.

[Eiter et al., 1999] Eiter, T., Faber, W., Leone, N. and Pfeifer, G. The Diagnosis Frontend of the DLV System. *AI Communications*, 1999, 12(1&2):99-111.

[Eiter et al., 1995] Eiter, T. and Gottlob, G. The Complexity of Logic-Based Abduction. *Journal of the ACM (JACM)*, 1995, 42(1):3-42.

[Eiter et al., 1997] Eiter, T., Gottlob, G. and Leone, N. Abduction from Logic Programs: Semantics and Complexity. *Theoretical Computer Science*, 1997, 189(1&2):129-177.

[Feldman et al., 2010] Feldman, A., Provan G. and Gemund A. V. Approximate Model-Based Diagnosis Using Greedy Stochastic Search. *Journal of Artificial Intelligence Research (JAIR)*, 2010, 87(14):3157-3174.

[Fagin, 2007] Fagin, R, Inverting Schema Mappings. *Transactions Database Systems (TODS)*, 32(4):2007.

[Fagin et al., 2015] Fagin, R, Kimelfeld, B. and Kolaitis, P. Dichotomies in the Complexity of Preferred Repairs. *Proc. of the ACM Symposium on Principles of Database Systems (PODS)*, 2015, pp. 3-15.

[Fernau et al., 2010] Fernau, H. Parameterized Algorithmics for *d*-Hitting Set. *International Journal of Computer Mathematics*, 2010, 87(14):3157-3174.

[Friedrich et al., 1990] Friedrich, G., Gottlob. G. and Nejdl, W. Hypothesis Classification, Abductive Diagnosis and Therapy. *Proc. of the International Workshop on Expert Systems in Engineering*, 1990, Springer LNCS 462, pp. 69-78.

[Feinberg, 1968] Feinberg, J. Collective Responsibility. *The Journal of Philosophy*, 1968, pp. 674-688.

[Flum, 2006] Flum, J. and Grohe, M. *Parameterized Complexity Theory*. Springer, 2006.

[Gertz, 1996] Gertz, M. Diagnosis and Repair of Constraint Violations in Database Systems. PhD Thesis, Universität Hannover, 1996.

[Gerstenberg et al., 2010] Gerstenberg, T. and Lagnado, D. Spreading the blame: The Allocation of Responsibility Amongst Multiple Agents. *Cognition*, 2010, 115(1):166-171

[Glavic & Miller, 2011] Glavic, B. and Miller., R. J. Reexamining Some Holy Grails of Data Provenance. *Proc. Theory and Practice of Provenance (TaPP)*, 2011.

[Glavic et al., 2015] Glavic, B., Kohler, S., Riddle, S. and Ludascher, B. Towards Constraint-Based Explanations for Answers and Non-Answers. *Proc. Theory and Practice of Provenance (TaPP)*, 2015.

[Glavic et al., 2007] Glavic, B., Klaus R., Dittrich, K. R. Data Provenance: A Categorization of Existing Approaches. *Proc. Datenbanksysteme für Business, Technologie und Web (BTW)*, 2007, 7(12):227-241.

[Glavic et al., 2013] Glavic, B. Siddique, J., Andritsos, P. and Miller., R. J. Provenance for Data Mining. *Proc. Theory and Practice of Provenance (TaPP)*, 2013.

[Greco et al., 2014] Greco, S., Pijcke, F. and Wijsen, J. Certain Query Answering in Partially Consistent Databases. *of the VLDB Endowment (PVLDB)*, 2014, 7(5):353-364.

[Green, 2009] Green, T. J. Containment of Conjunctive Queries on Annotated Relations. *Proc. International Conference on Database Theory (ICDT)* , 2009, PP. 296309.

[Garey et al., 2979] Garey, M. and Johnson, D. *Computers and Intractability*. W. H. Freeman and Company, 1979.

[Goldreich et al., 2008] Goldreich, O. *Computational Complexity*. Cambridge Univ. Press, 2008.

[Gottlob et al., 2010] Gottlob, G., Pichler, R. and Wei, F. Tractable Database Design and Datalog Abduction through Bounded Treewidth. *Information Systems*, 2010, 35(3):278-298.

[Halperin, 2002] Halperin, E. Improved Approximation Algorithms for the Vertex Cover Problem in Graphs and Hyper-Graphs. *SIAM Journal on Computing*, 2002, pp. 1608-1623.

[Halpern, 2014] Halpern, J. Y. Appropriate Causal Models and Stability of Causation. *Proc. International Conference on Principles of Knowledge Representation (KR)*, 2014.

[Halpern, 2008] Halpern, J. Y. Defaults and Normality in Causal Structures. *Proc. International Conference on Principles of Knowledge Representation (KR)*, 2008, pp. 198-208.

[Halpern, 2015a] Halpern, J. Y. A Modification of Halpern-Pearl Definition of Causality. *Proc. International Joint Conferences on Artificial Intelligence (IJCAI)*, 2015.

[Halpern, 2015b] Halpern, J. Y. Cause, Responsibility and Blame: A Structural-Model Approach. *Law, Probability and Risk*, 2015, 14 (2): 91-118.

[Halpern & Pearl, 2005] Halpern, J. Y. and Pearl, J. Causes and Explanations: A Structural-Model Approach: Part 1. *The British Journal for the Philosophy of Science*, 2005, 56:843-887.

[Hart & Honore, 1959] Hart, H. L. A. and Honore, A. M. *Causation in the Law.* Oxford University Press, 1959.

[Herschel et al., 2009] Herschel, M., Hernandez, M. A. and Tan, W. C. Artemis: A System for Analyzing Missing Answers. *Proc. of the VLDB Endowment (PVLDB)*, 2009, 2(2):1550-1553.

[Huang et al., 2008] Huang, J., Chen, T., Doan, A. and Naughton, J. F. On The Provenance of Non-Answers to Queries over Extracted Data. *Proc. of the VLDB Endowment (PVLDB)*, 2008, 1(1):736-747.

[Kakas & Mancarella, 1990] Kakas A. C. and Mancarella, P. Database Updates Through Abduction. *Proc. International Conference on Very Large Databases (VLDB),* , 1990, pp. 650-661.

[Kanagal et al., 2011] Kanagal, B., Li, J. and Deshpande, A. Sensitivity Analysis and Explanations for Robust Query Evaluation in Probabilistic Databases. *Proc. of the ACM SIGMOD International Conference on Management of Data*, 2011, pp. 841-852.

[Krentel, 1988] Krentel, M. The Complexity of Optimization Problems. *Journal of Computer and System Sciences*, 1988, 36:490-509.

[Karvounarakis, 2010] Karvounarakis, G. Ives, Z. G. and Tannen, V. Querying Data Provenance. *Proc. of the ACM SIGMOD International Conference on Management of Data*, 2010, pp. 951-962.

[Kimelfeld, 2012a] Kimelfeld, B. A Dichotomy in the Complexity of Deletion Propagation with Functional Dependencies. *Proc. of the ACM Symposium on Principles of Database Systems (PODS)*, 2012.

[Kimelfeld, 2012b] Kimelfeld, B., Vondrak, J. and Williams, R. Maximizing Conjunctive Views in Deletion Propagation. *Transactions Database Systems (TODS)*, 2012, 37(4):24.

[Khoussainova, 2012] Khoussainova, N., Balazinska, M. and Suciu, D. Perfxplain: Debugging Mapreduce Job Performance. *Proc. International Conference on Very Large Databases (VLDB)*, 2012, 5(7):598-609.

[Kohler et al., 2013] Kohler S., Ludascher, B. and Zinn, D. First-Order Provenance Games. In *In Search of Elegance in the Theory and Practice of Computation*, Springer LNCS 8000, 2013, pp. 382-399.

[Lewis, 1972] Lewis, D. Causation. *The Journal of Philosophy*, 1973, 70(17):556-567.

[Lopatenko & Bertossi, 2007] Lopatenko, A. and Bertossi, L. Complexity of Consistent Query Answering in Databases under Cardinality-Based and Incremental Repair Semantics. *Proc. International Conference on Database Theory (ICDT)*, 2007, pp. 179-193. Extended version posted at: arXiv:cs/0604002 [cs.DB].

[Meliou et al., 2011a] Meliou, A., Gatterbauer, W. and Suciu, D. Reverse Data Management. *Proc. of the VLDB Endowment (PVLDB)*, 2011, 4(12):1490-1493.

[Meliou et al., 2011b] Meliou, A., Gatterbauer, W. and Suciu, D. Bringing Provenance to its Full Potential Using Causal Reasoning. *Proc. Theory and Practice of Provenance (TaPP)*, 2011.

[Meliou et al., 2010a] Meliou, A., Gatterbauer, W. Moore, K. F. and Suciu, D. Why so? or Why not? Functional Causality for Explaining Query Answers. *Proc. of the International Workshop on Management of Uncertain Data*, 2010, pp. 317.

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

[Meliou et al., 2010c] Meliou, A., Gatterbauer, W. Moore, K. F. and Suciu, D. The Complexity of Causality and Responsibility for Query Answers and Non-Answers. *Proc. International Conference on Very Large Databases (VLDB)*, 2010, pp. 34-41.

[Meliou, et al., 2011c] Meliou, A., Gatterbauer, W, Nath., S. and Suciu, D. Tracing Data Errors with View-Conditioned Causality. *Proc. of the ACM SIGMOD International Conference on Management of Data* , 2011, pp. 505-516.

[Meliou et al., 2014] Meliou, A., Roy, S. and Suciu, D. Causality and Explanations in Databases. *Proc. Proceedings of the VLDB Endowment (PVLDB)*, 2014, 7(13):1715-1716.

[Moore, 1999] Moore, M. S. Causation and Responsibility. *Social Philosophy and Policy*, 1999, 16(2):1-51.

[Mozetic & Holzbaur, 1994] Mozetic, I. and Holzbaur, C. Controlling the Complexity in Model-Based Diagnosis *Annals of Mathematics and Artificial Intelligence*, 1994, 11(1-4): 297-314.

[Niedermeier, 2003] Niedermeier, R. and Rossmanith, P. An Efficient Fixed-Parameter Algorithm for 3-hitting set. *Journal of Discrete Algorithms*, 2003 1(1):89-102.

[Okun, 2005] Okun, M. On Approximation of the Vertex Cover Problem in Hypergraphs. *Discrete Optimization*, 2005, 2(1):101-111.

[Papadimitriou, 1994] Papadimitriou, Ch. *Computational Complexity*. Addison-Wesley, 1994.

[Pearl, 2009] Pearl, J. *Causality: Models, Reasoning and Inference.* Cambridge University Press, 2000/2009.

[Pearl, 2010] Pearl, J. An Introduction to Causal Inference. *The International Journal of Biostatistics*, 2010, 6(2).

[Peter & Molokov, 2009] Peter, D. and Molokov, L. The Union of Minimal Hitting Sets: Parameterized Combinatorial Bounds and Counting. *Journal of Discrete Algorithms*, 2009, 7(4):391-401.

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

[Poole, 1988] Poole, D. A Logical Framework for Default Reasoning. *Artificial Intelligence*, 1988, 36:27-47.

[Poole, 1989] Poole, D. Normality and Faults in Logic-Based Diagnosis. *Proc. International Joint Conferences on Artificial Intelligence (IJCAI)*, 1989, pp. 1304-1310.

[Poole, 1992] Poole, D. Logic Programming, Abduction and Probability. *New Generation Computing*, 1992, pp. 530-538.

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

[Reiter, 1984] Reiter, R. Towards a Logical Reconstruction of Relational Database Theory. In *On Conceptual Modelling*, Springer, 1984, pp. 191-233.

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

[Roy & Suciu, 2014] Roy, S. and Suciu, D. A Formal Approach to Finding Explanations for Database Queries. *Proc. of the ACM SIGMOD International Conference on Management of Data* , 2014, pp. 1579-1590,

[Riddle et al., 2014] Riddle, S., Kohler, S. and Ludascher, B. Towards Constraint Provenance Games. *Proc. Theory and Practice of Provenance (TaPP)*, 2014.

[Salimi & Bertossi, 2014] Salimi, B. and Bertossi, L. Causality in Databases: The Diagnosis and Repair Connections. Presented at *The 15th International Workshop on Non-Monotonic Reasoning (NMR)*, 2014. Posted at: arXiv:1404.6857 [cs.DB].

[Salimi & Bertossi, 2015a] Salimi, B. and Bertossi, L. From Causes for Database Queries to Repairs and Model-Based Diagnosis and Back. *Proc. International Conference on Database Theory (ICDT)* , 2015, pp. 342-362.

[Salimi & Bertossi, 2015b] Salimi, B. and Bertossi, L. Query-Answer Causality in Databases: Abductive Diagnosis and View-Updates. *Proc. UAI Workshop on Causal Inference*, CEUR-WS Proc. Vol-1504, 2015.

[Salimi & Bertossi, 2015c] Salimi, B. and Bertossi, L. From Causes for Database Queries to Repairs and Model-Based Diagnosis and Back. Journal Submission. Posted at: arXiv:1412.4311 [cs.DB].

[Simmhan et al., 2005] Simmhan, Y. L., Plale, B. and Gannon, D. A Survey of Data Provenance Techniques. Technical Report, Indiana University, 2005. https://www.cs.indiana.edu/ftp/techreports/TR618.pdf

[Staworko et al., 2012] Staworko, S., Chomicki, J. and Marcinkowski, J. Prioritized Repairing and Consistent Query Answering in Relational Databases. *Annals of Mathematics and Artificial Intelligence*, 2012, 64(2&3):209-246.

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

[Tran & Chan, 2010] Tran, Q. T. and Chan, C. Y. How to Conquer Why-Not Questions. *Proc. of the ACM SIGMOD International Conference on Management of Data* , 2010, pp. 15-26.

[Tannen, 2010] Tannen, V. Provenance for Database Transformations. *Proc. International Conference on Extending Database Technology (EDBT)*, 2010, page 1. Keynote Talk, Slides Posted at: http://www.cis.upenn.edu/ val/EDBTkeynoteLausanne.pdf

[Tannen, 2013] Tannen, V. Provenance Propagation in Complex Queries. In *Buneman Festschrift*, 2013, Springer LNCS 8000, pp. 483-493.

[Wu & Madden, 2013] Wu, E. and Madden, S. Scorpion: Explaining Away Outliers in Aggregate Queries. *Proc. of the VLDB Endowment (PVLDB)*, 2013, 6(8):553-564.

[Wright, 1985] Wright, R. W. Causation in Tort Law. *California Law Review*, 1985, 73:1735-1828.

[Wright, 1988] Wright, R. W. Causation, Responsibility, Risk, Probability, Naked Statistics, and Proof: Pruning the Bramble Bush by Clarifying the Concepts. *Iowa Law Review*, 1988, 73:1001-1077.

[Wright, 2001] Wright, R. W. Once More Into the Bramble Bush: Duty, Causal Contribution, and the Extent of Legal Responsibility. *Vanderbilt Law Review*, 2001, 54(3):1071-1132.

[Zultan et al., 2013] Zultan, R., Gerstenberg, T. and Lagnado, D. Finding Fault: Causality and Counterfactuals in Group Attributions. *Cognition*, 2013, 125(3):429-440.