Computing the Shapley Value of Tuples in Conjunctive Queries with Negation

Alon Reshef
Computing the Shapley Value of Tuples in Conjunctive Queries with Negation

Research Thesis

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Alon Reshef

Submitted to the Senate of the Technion — Israel Institute of Technology
Tishre 5781 Haifa September 2020
This research was carried out under the supervision of Associate Prof. Benny Kimelfeld, in the Faculty of Computer Science.

Some results in this thesis have been published as articles by the author and research collaborators in conferences and journals during the course of the author’s masters research period, the most up-to-date versions of which being:


Acknowledgements

I would like to thank my advisor, Prof. Benny Kimelfeld, for his guidance and support throughout this work. I would also like to thank Ester Livshits for all her professional help and contribution to this work, to my partner Naama and to my faculty friends who walked me through the process.

The generous financial help of the Technion is gratefully acknowledged.
## Contents

List of Figures  

Abstract  \hspace{2cm} 1  

1 Introduction  \hspace{2cm} 3  

2 Preliminaries  \hspace{2cm} 7  

3 Exact Evaluation for CQs with Negation  \hspace{2cm} 13  
   3.1 Dichotomy for Self-Join-Free Queries  \hspace{2cm} 13  
   3.2 Queries with Self-Joins  \hspace{2cm} 22  
   3.3 Summation over CQs with Negation  \hspace{2cm} 29  

4 Accounting for Exogenous Relations  \hspace{2cm} 31  
   4.1 Generalized Dichotomy  \hspace{2cm} 32  
   4.2 Algorithm for the Tractable Cases  \hspace{2cm} 36  
   4.3 Application to Probabilistic Databases  \hspace{2cm} 44  
   4.4 A Note about Endogenous Relations  \hspace{2cm} 44  

5 Approximation and Relevance  \hspace{2cm} 49  
   5.1 Additive vs. Multiplicative Approximation  \hspace{2cm} 49  
   5.2 Hardness of Multiplicative Approximation  \hspace{2cm} 54  
   5.3 More on the Complexity of Relevance  \hspace{2cm} 58  
   5.4 Results Summary  \hspace{2cm} 64  
   5.5 An Insight into UCQs with Negation  \hspace{2cm} 64  

6 Conclusions  \hspace{2cm} 67  

Hebrew Abstract  \hspace{2cm} i
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>The database of the running example</td>
<td>8</td>
</tr>
<tr>
<td>4.1</td>
<td>The Gaifman graphs of the queries $q$ (left) and $q'$ (right) of Example 4.2.</td>
<td>33</td>
</tr>
<tr>
<td>4.2</td>
<td>Illustration of the execution of the algorithm ExoShap on the query $q'$ from Example 4.2.</td>
<td>38</td>
</tr>
<tr>
<td>5.1</td>
<td>$q() := R(x, y), S(z, w), \neg R(x, w), T(z)$ - A database that violates the gap property for $q$</td>
<td>51</td>
</tr>
<tr>
<td>5.2</td>
<td>The database constructed in the proof of Proposition 5.6 for $(x_1 \lor x_2) \land (\neg x_1 \lor \neg x_3) \land (x_3 \lor x_4 \lor \neg x_1 \lor \neg x_2)$</td>
<td>55</td>
</tr>
<tr>
<td>5.3</td>
<td>Relevance and multiplicative approximation - CQs with negation</td>
<td>64</td>
</tr>
<tr>
<td>5.4</td>
<td>Relevance and multiplicative approximation - UCQs with negation</td>
<td>66</td>
</tr>
</tbody>
</table>
Abstract

The Shapley value is a conventional and well-studied function for determining the contribution of a player to the coalition in a cooperative game. Among its applications in a plethora of domains, it has recently been proposed to use the Shapley value for quantifying the contribution of a tuple to the result of a database query. The database consists of two types of tuples: exogenous facts and endogenous facts. We reason about the contribution of the endogenous facts, while the exogenous facts are considered background knowledge fixed in the database. Previous work has established a thorough understanding of the tractability frontier for the class of Conjunctive Queries (CQs) and aggregate functions over CQs. It has also been established that a tractable (randomized) multiplicative approximation exists for every union of CQs. Nevertheless, all of these results are based on the monotonicity of CQs. In this work, we investigate the implication of negation on the complexity of Shapley computation, in both the exact and approximate senses. We generalize a known dichotomy to account for negated atoms. It may happen that a query that is considered intractable by previous work is actually tractable, since some of the relations consist of only exogenous tuples. We further generalize the previous dichotomy and establish tractability for additional queries. We also show that negation fundamentally changes the complexity of approximation. We do so by drawing a connection to the problem of deciding whether a tuple is “relevant” to a query, and by analyzing its complexity.
Chapter 1

Introduction

Various formal measures have been proposed for quantifying the contribution of a fact \( f \) to a query answer. Meliou et al. [MGMS10] adopted the quantity of \textit{responsibility} that is inversely proportional to the minimal number of endogenous facts that should be removed to make \( f \) counterfactual (i.e., removing \( f \) transitions the answer from true to false). Following earlier notions of formal causality by Halpern and Pearl [HP01], Salimi et al. [SBSdB16] proposed the \textit{causal effect}: assuming endogenous facts are randomly removed independently and uniformly, what is the difference in the expected query answer between assuming the presence and the absence of \( f \)? A recent framework has proposed to adopt the \textit{Shapley value} to the task [LBKS20].

The Shapley value [Sha53] is a formula for wealth distribution in a cooperative game, and it has been applied in a plethora of domains that require to attribute a share of an outcome among a group of entities [AM03, AdK14]. The use cases include bargaining foundations in economics [Gul89], takeover corporate rights in law [Nen03], pollution responsibility in environmental management [PZ03, LZS15], influence measurement in social network analysis [NN11], the utilization of Internet Service Providers (ISPs) in networks [MCL+10], and advertisement effectiveness on the Web [BDG+19]. In data management, the Shapley value has been used for assigning a level of inconsistency to facts in inconsistent knowledge bases [HK10, YVCB18, GH06], and to determine the relative contribution of features in machine-learning predictions [LF18, LL17].

In the framework of Livshits et al. [LBKS20], query answering is viewed as a cooperative game where the players are the database facts and the utility function is the query answer, in the case of aggregate queries, or 0/1 in the case of Boolean queries.\(^1\) They showed that the (precise or approximate) evaluation of the Shapley value on common aggregate queries amounts to the evaluation of the Shapley value for a Boolean query, which we focus on in this work. To illustrate the Shapley value, consider the following Boolean query asking whether there is a farmer \( m \) who exports a product \( p \)

\(^1\)As discussed by Livshits et al. [LBKS20], the definition can be easily extended to non-Boolean queries by considering the contribution to each answer separately. A similar extension applies to aggregate queries with grouping.
to a country \( c \) where \( p \) does not grow.

\[ q() : \text{Farmer}(m), \text{Export}(m, p, c), \neg \text{Grows}(c, p) \]  \hspace{1cm} (1.1)

Different Farmer facts may have different Shapley values, depending on how crucial they are to the query—which products they export, whether they grow in the destination countries, and whether alternative Farmer facts export the same products. Similarly, each Grows\((c, p)\) fact has its Shapley value. However, while the Shapley value of Farmer facts can be either positive or zero (since they can only help in satisfying the query), the Shapley value of Grows facts can be either negative or zero (since they can only help in violating the query). As explained by Livshits et al. [LBKS20], understanding the complexity of the Shapley value for Boolean queries such as (1.1) is also necessary and sufficient for understanding the complexity of the Shapley value for aggregate queries such as

\[ \text{Count}\{c \mid \text{Farmer}(m), \text{Export}(m, p, c), \neg \text{Grows}(c, p)\} \]

that counts the countries that import one or more products that they do not grow.

As in previous work on quantification of contribution of facts [MGMS10, SBSdB16, LBKS20], we view the database as consisting of two types of facts: endogenous facts and exogenous facts. Exogenous facts are taken as given (e.g., inherited from external sources) without questioning, and are beyond experimentation with hypothetical or counterfactual scenarios. On the other hand, we may have control over the endogenous facts, and these are the facts for which we reason about existence and marginal contribution. The exogenous and endogenous facts are analogous to the observations and hypotheses in the study of abductive diagnosis [EGL97, EG95] that we refer to later on. In our context, the Shapley value considers only the endogenous facts as players in the cooperative game.

Livshits et al. [LBKS20] have studied the complexity of computing the Shapley value in the case where the query is a Conjunctive Query (CQ) or a Union of Conjunctive Queries (UCQ). Their work is restricted to positive CQs and UCQs (and aggregate functions thereof). In this paper, we study the impact of negation on this complexity. Negation transforms the query into a non-monotonic query and, as the reader might expect, the impact is fundamental. As a first step, we generalize their dichotomy in the complexity for CQs without self-joins into the class of CQs with negation and without self-joins (Theorem 3.1). Furthermore, we present hardness results for a certain class of CQs with negation and with self-joins, and we demonstrate a possible approach to handle with the Shapley value computation when self-joins are involved.

The dichotomy of Livshits et al. [LBKS20] classifies the CQs precisely as in CQ inference in probabilistic tuple-independent databases [DS04]: if the CQ is hierarchical, then the problem is solvable in polynomial time, and otherwise, it is FP\(\#\)P-complete.
(i.e., complete for the intractable class of polynomial-time algorithms with an oracle to, e.g., a counter of the satisfying assignments of a propositional formula). For illustration, the CQ of (1.1) falls on the hardness side. However, that classification does not take into account the assumption that some relations may contain only exogenous data. For example, in (1.1) we might consider the GROWS relation as consisting of only exogenous information. This assumption is very significant, as it makes our example CQ a tractable one for the Shapley value, in contrast to the dichotomy. In this paper, we establish a dichotomy that accounts for both negation and exogenous relations (Theorem 4.3).

An approximation of the Shapley value of a database fact \( f \) to a Boolean query can be computed via a straightforward Monte-Carlo (average-over-samples) estimation of the expectation that Shapley defines. This estimation guarantees an additive (or absolute) approximation. However, our interest is in a multiplicative (or relative) approximation, for two main reasons. First, we seek to understand the contribution of \( f \) relative to other facts, even if it is the case that the Shapley value is small. Second, in order to get an approximation of the contribution of a fact to an aggregate query, a multiplicative approximation is required [LBKS20].

In the case of a UCQ, a multiplicative approximation of the Shapley value is tractable, that is, there is a multiplicative Fully Polynomial-Time Approximation Scheme (FPRAS). This holds true for a simple reason: an additive FPRAS is also a multiplicative FPRAS, due to the following gap property: if the Shapley value is nonzero, then it must be “large”—at least the reciprocal of a polynomial. Nevertheless, once the CQ includes negated atoms, the gap property is no longer true. In fact, we show in Theorem 5.2 that every natural CQ with negation violates the gap property, since the Shapley value can be exponentially small. This phenomenon explains why negated atoms make the Shapley value fundamentally more challenging to approximate.

In itself, the violation of the gap property shows that the approach of an additive FPRAS fails to provide a multiplicative FPRAS. Yet, it does not show that multiplicative FPRAS is computationally hard, since there might be an alternative way of obtaining a multiplicative FPRAS in polynomial time. In order to prove hardness of approximation, we investigate the problem of determining whether a fact \( f \) is relevant to a query in the following sense: in the presence of all exogenous facts and some subset of the endogenous facts, adding \( f \) can change the query answer (from false to true or from true to false). In the case of a positive CQ, being relevant to the query coincides with being an “actual cause” in the framework of causal responsibility [MGMS10]. It is also similar to being a relevant hypothesis in the context of abductive diagnosis [CT91, EGL97, EG95]. We refer the reader to Bertossi and Salimi [BS17] who have established the connection between causal responsibility and abductive diagnosis.

The connection between the relevance to the query and the Shapley value is direct: if a fact \( f \) is polarity consistent in the sense that it occurs in a relation of only positive or only negative atoms, then \( f \) is relevant if and only if its Shapley value is nonzero (i.e.,
strictly positive or strictly negative). Therefore, a multiplicative FPRAS can decide on the relevance with high probability. In the contrapositive, if we prove that the relevance to the query is an intractable decision problem, then we also establish the intractability of an FPRAS approximation. Yet, the relevance is tractable for positive CQs, and hardness results are known only for Datalog programs with recursion [BS17, EGL97]. We prove here the existence of a CQ and a polarity-consistent fact $f$ such that the decision of relevance to the query (and, hence, the multiplicative approximation of the Shapley value) is intractable.

Nevertheless, the above approach for proving hardness of the multiplicative FPRAS of the Shapley value fails if we assume that the CQ itself is polarity consistent, that is, every relation symbol (and not just the one of $f$) occurs either only positively or only negatively. We prove that the relevance problem is solvable in polynomial time for a class of CQs with negation that includes every polarity-consistent CQ. The question of whether the Shapley value has a multiplicative FPRAS for polarity-consistent CQs (and in particular CQs without self-joins) remains an open problem for future investigation.

We also consider the relevance problem for UCQs with negation. We prove that the tractability of the relevance problem generalizes to polarity-consistent UCQs. Nevertheless, the tractability does not generalize to unions of polarity-consistent CQs—we show the existence of such a UCQ where the relevance problem is intractable, and so is the Shapley zeroness (and multiplicative approximation). In other words, if every relation symbol occurs either only positively or only negatively in a UCQ, then the relevance problem is solvable in polynomial time. Yet, the assumption that this consistency holds just in every individual disjunct is (provably) not enough.

The rest of the thesis is organized as follows. In the next chapter we introduce some basic terminology that will be used throughout the thesis. In Chapter 3, we study the complexity of computing the exact Shapley value for CQs with negation, and in Chapter 4 we explore the impact of exogenous relations on this complexity. We consider the approximate computation of the value and the relevance problem in Chapter 5. We summarize our results and discuss directions for future work in Chapter 6.
Chapter 2

Preliminaries

We first define the main concepts that we use throughout the paper.

Databases and Queries

A relational schema \( S \) is a finite collection of relation symbols \( R(A_1, \ldots, A_k) \), where each \( A_i \) is an attribute of \( R \), and \( k \) is the arity of \( R \), denoted by \( \text{arity}(R) \). We assume a countably infinite set \( \text{Const} \) of constants that are used as database values. A database \( D \) (over a schema \( S \)) associates with each relation symbol \( R \) in \( S \) a finite relation \( R^D \subseteq \text{Const}^{\text{arity}(R)} \). If \( (c_1, \ldots, c_k) \) is a tuple in \( R^D \), then we refer to \( R(c_1, \ldots, c_k) \) as a fact of \( D \). We then identify a database \( D \) by the set of its facts. We assume that a database \( D \) consists of two disjoint subsets of facts: the set \( D_x \) of exogenous facts and the set \( D_n \) endogenous facts. Hence, we have \( D = D_x \cup D_n \).

Example 2.1. The database of our running example is depicted in Figure 2.1. The relations STUD and TA store the names of graduate students and teaching assistants in the university, respectively. The relation COURSE contains information about courses given in different faculties of the university. The relation REG associates graduate students with the courses they take, and the relation ADV associates students with their academic advisor. For example, Adam is a student and a teaching assistant in the university. He is registered to two courses—OS is given in the Electrical Engineering faculty and AI in the Computer Science faculty. Michael is the academic advisor of Adam. ■

A Boolean conjunctive query over a schema \( S \) is an expression of the form:

\[
q() : R_1(\tilde{t}_1), \ldots, R_n(\tilde{t}_n)
\]

where each \( R_i \) is a relation symbol of \( S \) and each \( \tilde{t}_i \) is a tuple of variables and constants (where the arity of \( \tilde{t}_i \) matches that of \( R_i \)). We refer to a Boolean conjunctive query simply as a CQ. We refer to each \( R_i(\tilde{t}_i) \) as an atom of \( q \). We denote by \( R_\alpha \) the relation
corresponding to the atom $\alpha$ of $q$. A self-join in a CQ $q$ is a pair of distinct atoms of $q$ over the same relation symbol. If $q$ does not contain any self-joins, then we say that $q$ is self-join-free. A homomorphism from $q$ to $D$ is a mapping of the variables in $q$ to the constants of $D$ such that every atom in $q$ is mapped to a fact of $D$. We denote by $D \models q$ the fact that $D$ satisfies $q$ (i.e., there is a homomorphism from $q$ to $D$) and by $D \not\models q$ the fact that $D$ violates $q$ (i.e., there is no such homomorphism).

Let $q$ be a CQ. For every variable $x$ of $q$, we denote by $A_x$ the set of all atoms $R_i(\vec{t}_i)$ of $q$ such that $x$ occurs in $\vec{t}_i$. We say that $q$ is hierarchical [DRS09] if at least one of the following holds for all variables $x$ and $y$ of $q$: (1) $A_x \subseteq A_y$, (2) $A_y \subseteq A_x$, or (3) $A_x \cap A_y = \emptyset$. It is known [DS04] that if $q$ is not hierarchical, then there exist three atoms $\alpha_x, \alpha_y,$ and $\alpha_{x,y}$ in $q$ such that the variable $x$ occurs in $\alpha_x$ but not in $\alpha_y$, the variable $y$ occurs in $\alpha_y$ but not in $\alpha_x$ and both variables occur in $\alpha_{x,y}$. We refer to each such triplet of atoms as a non-hierarchical triplet of $q$.

A CQ with safe negation, or CQ$^-$ for short, has the form:

$$q() \triangleq R_1(\vec{t}_1), \ldots, R_n(\vec{t}_n), \neg R_1'(\vec{t}_1'), \ldots, \neg R_m'(\vec{t}_m')$$

where every variable that occurs in a negated atom also occurs in an atom without negation. We refer to the atoms of $q$ appearing without negation as the positive atoms of $q$ and to the atoms that appear with negation as the negative atoms of $q$. We denote by $\text{Pos}(q)$ and $\text{Neg}(q)$ the sets of positive and negative atoms of $q$, respectively. For a CQ$^-$, we denote by $D \models q$ the fact that there is a homomorphism mapping the variables of $q$ to constants of $D$ such that every positive atom and none of the negative atoms of $q$ is mapped to a fact of $D$. The extension of the definition hierarchical from CQs to CQ$^-$’s is straightforward (i.e., we do not distinguish between positive and negative atoms in that definition). We call a relation symbol polarity consistent if it appears in $q$ only in positive atoms or only in negative atoms.
Example 2.2. We use the following queries in our examples:

\[ q_1() : \text{Stud}(x), \neg \text{TA}(x), \text{REG}(x, y) \]
\[ q_2() : \text{Stud}(x), \neg \text{TA}(x), \text{REG}(x, y), \neg \text{COURSE}(y, \text{CS}) \]
\[ q_3() : \text{ADV}(x, y), \text{ADV}(x, z), \neg \text{TA}(y), \neg \text{TA}(z), \text{REG}(y, \text{IC}), \text{REG}(z, \text{DB}) \]
\[ q_4() : \text{ADV}(x, y), \text{ADV}(x, z), \text{TA}(y), \neg \text{TA}(z), \text{REG}(z, w), \neg \text{REG}(y, w) \]

Each of these queries is a CQ \(^{-}\). The queries \( q_1 \) and \( q_2 \) are self-join-free, while the queries \( q_3 \) and \( q_4 \) have self-joins (e.g., the relation Adv occurs twice). The query \( q_1 \) is hierarchical since \( A_y \subseteq A_x \), but the others are not, since each of them contains a non-hierarchical triplet (e.g., \( \text{ADV}(x, y), \text{ADV}(x, z), \neg \text{TA}(z) \)). ■

A Union of Conjunctive Queries (UCQ) is an expression of the form:

\[ q() := q_1() \lor \cdots \lor q_n() \]

where each \( q_i \) is a CQ, and it is satisfied by a database \( D \) if \( D \models q_i \) for at least one \( i \in \{1, \ldots, n\} \). A union of CQ\(^{-}\)s is called a UCQ\(^{-}\) for short.

The Shapley Value

Given a set \( A \) of players, a cooperative game is a function \( v : \mathcal{P}(A) \rightarrow \mathbb{R} \) that maps every subset \( B \) of \( A \) to a number \( v(B) \), such that \( v(\emptyset) = 0 \). The value \( v(B) \) represents a value jointly obtained by the players of \( B \) when they cooperate. The Shapley value [Sha53] measures the share of each player \( a \in A \) in the value \( v(A) \) jointly obtained by all players. Intuitively, the Shapley value is the expected contribution of \( a \) in a random permutation of the players, where the contribution of \( a \) is the change of \( v \) due to the addition of \( a \). More formally, the Shapley value is defined as:

\[
\text{Shapley}(A, v, a) := \frac{1}{|A|!} \sum_{\sigma \in \Pi_A} (v(\sigma_a \cup \{a\}) - v(\sigma_a))
\]

where \( \Pi_A \) is the set of all possible permutations over the players in \( A \), and for each permutation \( \sigma \), we denote by \( \sigma_a \) the set of players that appear before \( a \) in the permutation.

Let \( S \) be a schema, \( D \) a database over \( S \), \( q \) a CQ or CQ\(^{-}\), and \( f \) an endogenous fact of \( D \). Following Livshits et al. [LBKS20], the Shapley value of \( f \) w.r.t. \( q \), denoted \( \text{Shapley}(D, q, f) \), is the value of \( \text{Shapley}(A, v, a) \) where:

- \( A = D_n \).
- \( v(E) = q(E \cup D_a) - q(D_a) \) for all \( E \subseteq D_n \).
- \( a = f \).
That is, we consider a cooperative game where the endogenous facts are the players and the wealth function $v(E)$ measures the change to the result of the query due to the addition of the facts of $E$ to the exogenous facts. Here, we view a Boolean CQ $q$ as a numerical query such that $q(D) = 1$ if $D \models q$ and $q(D) = 0$ otherwise.

**Example 2.3.** Consider again the database of our running example. We assume that all the facts in STUD, COURSE and ADV are exogenous, while the facts in TA and REG are endogenous. Consider the query $q_1$ asking if there is a student who is not a TA and is registered to at least one course. Note that facts from REG can only have a positive impact on the query result (i.e., they can only change it from false to true), while the facts of TA can only have a negative impact on the result (i.e., they can only change it from true to false). Clearly, it holds that $D_x \not \models q$, as no fact of REG appears in $D_x$. The answer to $q_1$ on $D_x \cup E$ for some $E \subseteq D_n$ is true if at least one of the following holds: (1) $f_1^3$ or $f_2^3$ appear in $E$, (2) $f_1^1$ or $f_2^1$ appear in $E$, but $f_1^1$ does not, or (3) $f_3^3$ appears in $E$, but $f_2^3$ does not.

We can immediately see that $f_3^3$ can never affect the query result, since David does not appear in REG; hence, we have that Shapley$(D, q_1, f_3^3) = 0$. Adding the fact $f_1^1$ in a permutation would change the query result from true to false if $f_3^3$ has been added before, and none of conditions (1) or (2) holds. Thus, the following subsets of facts may appear before $f_3^3$ in a permutation where it changes the query result: $\{ f_3^3, f_1^1 \}$, $\{ f_3^3, f_2^1 \}$, $\{ f_3^3, f_2^2 \}$, $\{ f_3^3, f_1^1, f_1^1 \}$, and $\{ f_3^3, f_2^1, f_1^1 \}$. Note that we can add $f_3^3$ to each of these subsets; thus, we have the following:

$$\text{Shapley}(D, q_1, f_3^3) = - \frac{1! \cdot 6! + 2 \cdot 2! \cdot 5! + 3 \cdot (3! \cdot 4! + 4! \cdot 3!) + 5! \cdot 2!}{8!}$$

and we conclude that Shapley$(D, q_1, f_3^3) = - \frac{2}{35}$. Similarly, the fact $f_1^1$ changes the query result from true to false when at least one of $f_1^1$ or $f_2^1$ appears earlier in the permutation, and none of conditions (1) or (2) holds. That is, the fact $f_1^1$ should appear after one of the following subsets of facts: $\{ f_1^1 \}$, $\{ f_1^1, f_2^1 \}$, $\{ f_1^1, f_2^2 \}$, $\{ f_1^1, f_3^3 \}$, $\{ f_1^1, f_2^3 \}$, $\{ f_2^1, f_3^3 \}$, $\{ f_2^1, f_2^3, f_2^3 \}$. We can again add $f_3^3$ to each of the subsets. Thus, the following holds:

$$\text{Shapley}(D, q_1, f_1^1) = - \frac{2 \cdot 1! \cdot 6! + 5 \cdot 2! \cdot 5! + 6 \cdot 3! \cdot 4! + 4 \cdot 4! \cdot 3! + 5! \cdot 2!}{8!}$$

and Shapley$(D, q_1, f_1^1) = - \frac{3}{25}$. As it holds that $|\text{Shapley}(D, q_1, f_1^1)| > |\text{Shapley}(D, q_1, f_3^3)|$, we deduce that the fact that Adam is a TA has a greater negative impact on $q_1$ than the fact that Ben is a TA. This is expected, since Adam is registered to more courses.

Next, we compute the Shapley value of the facts in the relation REG. If the fact $f_3^3$ appears in a permutation before $f_2^3$ and conditions (1) and (2) do not hold, then $f_3^3$ changes the query result from false to true. Thus, there are five possible subsets of the endogenous facts that can appear before $f_3^3$ in a permutation where $f_3^3$ affects the query result: $\emptyset$, $\{ f_1^1 \}$, $\{ f_1^1, f_1^1 \}$, $\{ f_3^3, f_3^3 \}$, $\{ f_1^1, f_2^3, f_1^1 \}$. To each one of those subsets we can add the fact $f_3^3$; hence, overall, we have ten possible subsets and we conclude that:
As for the fact \( f_1 \), it changes the query result from false to true if both \( f_2 \) and \( f_3 \) appear later in the permutation, and none of the conditions (1) or (3) holds. Hence, the subsets of the endogenous facts that can appear before \( f_1 \) in a permutation are: \( \emptyset, \{ f_3 \}, \{ f_2, f_3 \}, \{ f_3, f_2, f_1 \} \), and we have that:

\[
\text{Shapley}(D, q_1, f_1^t) = \frac{7! + 2 \cdot 1! \cdot 6! + 3 \cdot 2! \cdot 5! + 3 \cdot 3! \cdot 4! + 4! \cdot 3!}{8!} = \frac{27}{140}
\]

The same calculations hold for the fact \( f_2 \).

Finally, adding \( f_4 \) to a permutation before \( f_5 \) would change the query result from false to true, unless conditions (2) or (3) hold. In this case, there is a much larger number of subsets of facts that can appear before \( f_4 \) in a permutation where it changes the query result. We divide these subsets to four groups:

- Subsets without any fact from Reg, that is, all subsets of \( \{ f_1^t, f_2^t, f_3^t \} \).
- Subsets where the only possible facts from Reg are \( f_1 \), \( f_2 \). In this group we have: \( \{ f_1, f_1^t \}, \{ f_2, f_1^t \}, \{ f_1, f_2, f_1^t \} \). To each of these subsets we can add a subset of the facts \( \{ f_3^t, f_3^t \} \).
- Subsets that include only the fact \( f_3 \) from Reg. Here we have the subset \( \{ f_3^t, f_3^t \} \), and we can add to it every subset of \( \{ f_1^t, f_2^t \} \).
- Subsets that contain the fact \( f_3^t \) and at least one of the facts \( f_1 \), \( f_2 \), that is, \( \{ f_3^t, f_2, f_1, f_1^t \}, \{ f_3^t, f_2, f_3, f_1^t \}, \{ f_3^t, f_3, f_2, f_1^t \} \). To each one of these subsets we can add the fact \( f_3^t \).

Overall we have thirty possible subsets, and we conclude that:

\[
\text{Shapley}(D, q_1, f_4^t) = \frac{7! + 3 \cdot 1! \cdot 6! + 6 \cdot 2! \cdot 5! + 8 \cdot 3! \cdot 4! + 7 \cdot 4! \cdot 3! + 4 \cdot 5! \cdot 2! + 6! \cdot 1!}{8!} = \frac{13}{42}
\]

The same calculations hold for \( \text{Shapley}(D, q_1, f_5^t) \). Note that the sum over the Shapley values of all the endogenous facts is 1. This is expected, due to the fundamental “efficiency” property of the Shapley value which is: \( \sum_{a \in A} \text{Shapley}(A, v, a) = v(A) \) (i.e., the sum over the Shapley values of all players in a cooperative game equals to the value obtained by all players). In this case, \( v(A) = q_1(D) - q_1(D_x) = 1 \), as the the answer to the query is true over the whole database, while the answer is false when none of the endogenous facts are considered in the database.
Chapter 3

Exact Evaluation for CQs with Negation

In this chapter, we investigate the complexity of computing the Shapley value for CQ’s. First, we focus on CQ’s without self-joins and establish a dichotomy in the data complexity of the problem. Next, we provide an insight into the complexity of the problem for CQ’s with self-joins.

3.1 Dichotomy for Self-Join-Free Queries

We study the problem of computing the Shapley value for self-join-free CQ’s and establish the following dichotomy in the data complexity of the problem.

Theorem 3.1. Let q be a CQ without self-joins. If q is hierarchical, then Shapley(D, q, f) can be computed in polynomial time, given D and f. Otherwise, its computation is FP\#P-complete.\(^1\)

For illustration, the theorem states that the Shapley value can be computed in polynomial time for the query \(q_1\) of Example 2.2, but computing it for the query \(q_2\) is FP\#P-complete. Interestingly, the classification criteria is the same as the one for self-join-free CQs without negation [LBKS20]. Hence, the added negation does not change the complexity picture for the exact computation of the Shapley value. (However, as we will show later, the addition of negation has a significant impact on the approximate computation of the value.) Next, we discuss the proof of Theorem 3.1.

Tractability Side

Livshits et al. [LBKS20] introduced an algorithm for computing the Shapley value for hierarchical self-join-free CQs. This algorithm relies on a reduction from the problem

\(^1\)Recall that FP\#P is the class of problems that can be solved in polynomial time with an oracle to a \#P-complete problem.
of computing the Shapley value to that of computing the number of subsets of size \( k \) of \( D_n \) that, along with \( D_x \), satisfy \( q \). We denote this problem as \(|\text{Sat}(D, q, k)|\).

As the reduction does not assume anything about \( q \) other than the fact that it is a Boolean query, the same reduction applies to CQ\(^\neg\)'s. Hence, it is only left to show that \(|\text{Sat}(D, q, k)|\) can be computed in polynomial time for a hierarchical CQ\(^\neg\).

**Lemma 3.1.1.** Let \( q \) be a hierarchical CQ\(^\neg\) without self-joins. There is a polynomial-time algorithm for computing the number \(|\text{Sat}(D, q, k)|\) of \( k \)-subsets \( E \) of \( D_n \), such that \((D_x \cup E) \models q\). We denote this problem as \(|\text{Sat}(D, q, k)|\).

**Proof.** The algorithm \( \text{CntSat} \) of Livshits et al. [LBKS20] for computing \(|\text{Sat}(D, q, k)|\) is a recursive algorithm that reduces the number of variables in the query with each recursive call. If there is a variable \( x \) that occurs in every atom of \( q \) (i.e., a root variable), then the problem is solved using dynamic programming, by considering every possible value of \( x \). If no variable occurs in all atoms, then the query can be split into two disjoint sub-queries, in which case the problem is solved separately for each one of them. The treatment of these two cases applies to any hierarchical CQ\(^\neg\) as it only relies on the hierarchical structure of the query; however, the treatment of the base case, when no variables occur in \( q \), does not apply to queries with negation, and we now explain how it should be modified.

If at least one atom of \( q \) does not correspond to any fact of \( D \), then \( \text{CntSat} \) will return 0, as \( D \not\models q \). This will also be the case if \( k < |A| \) or \( k > |D_n| \), where \( A = \text{Atoms}(q) \cap D_n \). In any other case, the algorithm will return \( \binom{|D_n|-|A|}{k-|A|} \) which is the number of possibilities to select \( k - |A| \) facts among those in \( D_n \setminus A \) (as every fact of \( A \) should be selected to satisfy \( q \)). By modifying the base case in the following way, we ensure that the algorithm returns \(|\text{Sat}(D, q, k)|\) for a CQ\(^\neg\). The algorithm will return 0 in one of the following cases: (a) at least one of the positive atoms of \( q \) does not appear as a fact of \( D \), (b) at least one of the negative atoms of \( q \) appears as a fact of \( D \), or (c) \( k < |A^+| \) or \( k > |D_n| \) where \( A^+ = \text{Pos}(q) \cap D_n \). In any other case, the result will be \( \binom{|D_n|-|A^+|}{k-|A^+|} \). It is rather straightforward that the modified algorithm will indeed return \(|\text{Sat}(D, q, k)|\) in polynomial time, based on the correctness and efficiency of \( \text{CntSat} \). \( \blacksquare \)

**Hardness for Basic Queries**

In the remainder of this section, we focus on the proof of the negative side of the theorem. We start by proving hardness for the four simplest non-hierarchical CQ\(^\neg\)'s:

\[
\begin{align*}
q_{\text{RST}}() & := R(x), S(x, y), T(y) \\
q_{\text{R}-\text{S}-\text{T}}() & := \neg R(x), S(x, y), \neg T(y) \\
q_{\text{R}-\text{S}-\text{T}}() & := R(x), \neg S(x, y), T(y) \\
q_{\text{R,S}-\text{T}}() & := R(x), S(x, y), \neg T(y)
\end{align*}
\]
The proof for \( q_{RST} \) is given in [LBKS20]; hence, we prove that each one of the remaining three queries above is \( \text{FP}^\text{P} \)-complete separately. We start with the query \( q_{-RS-T} \).

**Lemma 3.1.2.** Computing Shapley\((D, q_{-RS-T}, f)\) is \( \text{FP}^\text{P} \)-complete.

**Proof.** We construct a reduction from the problem of computing Shapley\((D, q_{RST}, f)\) to that of computing Shapley\((D, q_{-RS-T}, f)\). We make the following assumptions on the input database \( D \) to the first problem: (a) every fact in \( S \) is exogenous, and (b) for every \( S(a, b) \) in \( D \), it holds that both \( R(a) \) and \( T(b) \) are in \( D \) as well. The database used in the proof of hardness for \( q_{RST} \) [LBKS20] satisfies these properties; hence, computing Shapley\((D, q_{RST}, f)\) for such an input is \( \text{FP}^\text{P} \)-complete.

Let \( D \) be such database, and let \( f \in D_n \). Assume, without loss of generality, that \( f = R(0) \) (the proof for a fact \( f \) in \( T \) is symmetric). Let:

\[
P_1 = \{ \sigma \mid \sigma \in \Pi_{D_n}, (\sigma_f \cup D_x) \not\models q_{RST}, (\sigma_f \cup D_x \cup \{ f \}) \models q_{RST} \}
\]

\[
P_2 = \{ \sigma \mid \sigma \in \Pi_{D_n}, (\sigma_f \cup D_x) \models q_{-RS-T}, (\sigma_f \cup D_x \cup \{ f \}) \not\models q_{-RS-T} \}
\]

Recall that \( \Pi_{D_n} \) is the set of all possible permutations of the endogenous facts, and \( \sigma_f \) is the set of facts that appear before \( f \) in the permutation \( \sigma \). That is, \( P_1 \) and \( P_2 \) are the sets of all permutations where \( f \) changes the query result from false to true w.r.t. \( q_{RST} \) and from true to false w.r.t. \( q_{-RS-T} \), respectively. We claim that \( |P_1| = |P_2| \) due to a bijection that exists between the two sets.

Let \( g \) be a function defined as follows:

\[
g : P_1 \rightarrow P_2, \quad g(\sigma) = \sigma^R
\]

where \( \sigma^R \) is the permutation \( \sigma \) in reversed order (i.e., if \( \sigma = (f_1, \ldots, f_n) \), then \( \sigma^R = (f_n, \ldots, f_1) \), where \( n = |D_n| \)). First, we prove that if \( \sigma \in P_1 \) then \( g(\sigma) \in P_2 \). If adding \( f \) to the database after all the facts in \( D_x \cup \sigma_f \) changes the query result from false to true, then for every \( S(a, b) \in D_x \), at least one of the facts \( R(a) \), \( T(b) \) is not in \( D_x \cup \sigma_f \), and there is at least one fact \( T(c) \) in \( D_x \cup \sigma_f \) such that \( S(0, c) \in D_x \). According to our assumption, for every \( S(a, b) \in D_x \) both \( R(a) \) and \( T(b) \) exist in \( D \); hence, at least one of those is in \( \sigma^R \). Moreover, there is at least one fact \( T(c) \), which is not in \( D_x \cup \sigma^R \), such that \( S(0, c) \in D_x \). We conclude that \( (D_x \cup \sigma_f) \models q_{-RS-T} \) because \( S(0, c) \) satisfies \( q_{-RS-T} \) and \( T(c) \not\in (D_x \cup \sigma_f) \). Moreover, since for every fact \( S(a, b) \) for \( a \neq 0 \) at least one of \( R(a) \) or \( T(b) \) is in \( D_x \cup \sigma_f \), we have that \( (D_x \cup \sigma_f \cup \{ f \}) \not\models q_{-RS-T} \), and \( g(\sigma) \in P_2 \).

Next, we prove that the function is injective and surjective.

- **Injectivity:** Let \( \sigma_1, \sigma_2 \in P_1 \) such that \( \sigma_1 \neq \sigma_2 \). It follows directly that \( \sigma_1^R \neq \sigma_2^R \).

- **Surjectivity:** Let \( \sigma \in P_2 \). Since \( (D_x \cup \sigma_f) \models q_{-RS-T} \), it holds that for every \( S(a, b) \in D_x \) such that \( a \neq 0 \), at least one of \( R(a) \), \( T(b) \) is in \( D_x \cup \sigma_f \). In addition, since \( (D_x \cup \sigma_f \cup \{ f \}) \not\models q_{-RS-T} \), there is a fact \( S(0, c) \in D_x \) such that \( T(c) \) is
not in $D_x \cup \sigma_f$. Observe that $\sigma_R \in P_1$, as for every $S(a,b) \in D_x$ such that $a \neq 0$, at least one of $R(a), T(b)$ is not in $D_x \cup \sigma_R$, and there is a fact $S(0,c) \in D_x$ such that $T(c)$ is in $D_x \cup \sigma_R$. Thus $f$ changes the query result from false to true w.r.t. $q_{RST}$ in $\sigma_R$. Since $\sigma_R = \sigma$, we get that $g(\sigma_R) = \sigma$.

Thus, by the definition of the Shapley value we obtain that:

$$\text{Shapley}(D, q_{RST}, f) = \frac{|P_1|}{n!} = \frac{|P_2|}{n!} = \text{Shapley}(D, q_{RST}, f)$$

and that concludes our proof. ■

Next, we give the proof of hardness for $q_{R-ST}$.

**Lemma 3.1.3.** Computing $\text{Shapley}(D, q_{R-ST}, f)$ is FP$\P$-complete.

**Proof.** We show a reduction from the problem of computing $\text{Shapley}(D, q_{RST}, f)$ to that of computing $\text{Shapley}(D, q_{R-ST}, f)$. As in the proof of the previous lemma, we make the assumption that every fact in $S$ is exogenous, while preserving the hardness of the original problem. Let $D$ be a database, and let $f \in D_n$. Assume, without loss of generality, that $f = R(0)$. Let $D'$ be a database over the same schema as $D$, where the relations $R$ and $T$ of $D'$ consist of the exact same set of facts as the relations $R$ and $T$ in $D$. As for the relation $S$ in $D'$, it will contain the following set of facts:

$$S^{D'} = \{S(a,b) \mid R(a), T(b) \in D, S(a,b) \notin D\}$$

That is, $S^{D'}$ is the “complement” of the relation $S^D$: we add a fact $f$ over the domain of $D$ to $S^{D'}$ if and only if this fact is not in $S^D$. Observe that $D'_n = D_n$. Next, we define two sets of permutations:

$$P_1 := \{\sigma \in \Pi_{D_n} \mid (D_x \cup \sigma_f) \not\models q_{RST}, (D_x \cup \sigma_f \cup \{f\}) \models q_{RST}\}$$

$$P_2 := \{\sigma \in \Pi_{D_n} \mid (D'_x \cup \sigma_f) \not\models q_{R-ST}, (D'_x \cup \sigma_f \cup \{f\}) \models q_{R-ST}\}$$

We prove that $P_1 = P_2$ by showing a mutual inclusion between the sets:

**$P_1 \subseteq P_2$:** Let $\sigma \in P_1$. Since $(D_x \cup \sigma_f) \not\models q_{RST}$, for every pair of facts $R(a), T(b)$ in $\sigma_f$, we have that $S(a,b) \notin D_x$. Moreover, since $(D_x \cup \sigma_f \cup \{f\}) \models q_{RST}$, there is a fact $S(0,c) \in D_x$ such that $T(c) \in \sigma_f$. By the definition of $S^{D'}$, we have that $S(a,b) \in D'_x$ for every pair of facts $R(a), T(b)$ in $\sigma_f$; hence, $(D'_x \cup \sigma_f) \not\models q_{R-ST}$. We also have that $S(0,c) \notin D'_x$ (while $T(c) \in \sigma_f$), and we conclude that $(D'_x \cup \sigma_f \cup \{f\}) \models q_{R-ST}$.

**$P_2 \subseteq P_1$:** Let $\sigma \in P_2$. For every $R(a), T(b) \in \sigma_f$, there is a fact $S(a,b) \in D'_x$, and there is also a fact $T(c) \in \sigma_f$ such that $S(0,c) \notin D'_x$. Thus, it holds that for every pair of facts $R(a), T(b) \in \sigma_f$, the fact $S(a,b)$ does not belong to $D_x$ by the definition of $S^{D'}$, so $R(a), T(b)$ cannot be a part of any homomorphism from $q_{RST}$ to $D$, and overall $(D_x \cup \sigma_f) \not\models q_{RST}$. Moreover, $S(0,c) \in D_x$, so the set of facts $\{R(0), S(0,c), T(c)\}$ satisfies $q_{RST}$, and we conclude that $(D'_x \cup \sigma_f \cup \{f\}) \models q_{RST}$.  

16
Finally, we deduce that:

$$\text{Shapley}(D, q_{RST}, f) = \frac{|P_1|}{n!} = \frac{|P_2|}{n!} = \text{Shapley}(D', q_{R-ST}, f)$$

where $n = |D_n| = |D'_n|$.

Most intricate is the proof of hardness for the query $q_{RS-T}$. This is due to its non-symmetrical structure that prevents us from constructing a direct reduction from the problem of computing Shapley($D, q_{RST}, f$). As usually done, we show a Turing reduction from a #P-complete problem to our problem, which implies that every problem in FP#P can be reduced to our problem.

**Lemma 3.1.4.** Computing Shapley($D, q_{RS-T}, f$) is FP#P-complete.

**Proof.** Similarly to the proof of hardness for $q_{RST}$ [LBKS20], we construct a reduction from the known #P-complete problem of computing $|S(g)|$—the number of independent sets in a bipartite graph $g$. Given an input graph $g = (A \cup B, E)$, where $A$ and $B$ are the disjoint sets of vertices in $g$, we define the following set:

$$S(g) := \{A' \cup B'|A' \subseteq A, B' \subseteq B, \forall (a,b) \in E(a \in A' \Rightarrow b \in B')\}$$

That is, $S(g)$ contains all subsets of the vertices in $g$, such that if a vertex from $A$ is in the subset, all of its neighbours from $B$ are in the subset as well. Note that a subset in $S(g)$ may include additional vertices from $B$ that are not connected to any vertex from $A$ in the subset. We denote by $S(g, k)$ the set of all $F \in S(g)$ such that $|F| = k$.

Given a bipartite graph $g = (A \cup B, E)$ where $|A| = m, |B| = n$, and $N = m + n$ such that none of the vertices in $g$ is isolated, we build a database $D^0$ which consists of the following facts: an endogenous fact $R(a)$ for every vertex $a \in A$, an endogenous fact $T(b)$ for every vertex $b \in B$, an exogenous fact $S(a, b)$ for every edge $(a, b) \in E$, and another endogenous fact $T(0)$ for a fresh constant 0. In addition, for every $a \in A$, $D^0$ will contain the exogenous fact $S(a, 0)$. We will compute the Shapley value for the fact $f = T(0)$. (In fact, we will compute $1 - \text{Shapley}(D_0, q_{RS-T}, f)$.)

There are two types of permutations $\sigma \in \Pi_{D^0_n}$ for which it holds that $q_{RS-T}(D^0_\chi \cup \sigma_f) = q_{RS-T}(D^0_\chi \cup \sigma_f \cup \{f\})$ (i.e., permutations where adding $f$ to the database after the facts in $D^0_\chi \cup \sigma_f$ does not change the result of $q_{RS-T}$):

1. $(D^0_\chi \cup \sigma_f) \not\models q_{RS-T}$ and $(D^0_\chi \cup \sigma_f \cup \{f\}) \not\models q_{RS-T}$. In this case, no fact from $R$ is in $\sigma_f$. Otherwise, there is $R(a) \in \sigma_f$ such that $\{R(a), S(a, 0)\}$ satisfy $q_{RS-T}$, based on the construction of $D^0$, in which case $(D^0_\chi \cup \sigma_f) \models q_{RS-T}$. Adding $f$ cannot turn the query result to true. The number of permutations that satisfy this property is $P_{0-0} = \frac{(N + 1)!}{m + 1}$, since each of the $m + 1$ facts in $R^D \cup \{f\}$ has an equal chance to be the first (among these facts) to appear in a permutation, and we are interested in the permutations where $f$ appears before any fact in $R^D$.  

17
2. \((D^0_x \cup \sigma_f) \models q_{RS-T}\) and \((D^0_x \cup \sigma_f \cup \{f\}) \models q_{RS-T}\). Here, we observe that \(F = \{a \mid R(a) \in \sigma_f\} \cup \{b \mid T(b) \in \sigma_f\}\) is a subset of vertices in \(g\) such that \(F \notin S(g)\). Otherwise, we get that for every \(S(a, b) \in D^0_n\) if \(R(a) \in \sigma_f\) then \(T(b) \in \sigma_f\), by the definition of \(S(g)\). Therefore, none of the pairs \(\{R(a), S(a, b)\}\) \((D_x \cup \sigma_f)\) satisfy \(q_{RS-T}\), since \(T(b) \in \sigma_f\) as well. Hence, every pair of facts from \(R\) and \(S\) that satisfies \(q_{RS-T}\) is of the form \(R(a), S(a, 0)\), but when we add \(f\) to the database in this permutation, none of these pairs satisfy the query anymore, in contradiction to the fact that \((D^0_x \cup \sigma_f \cup \{f\}) \models q_{RS-T}\). We denote the number of permutations that satisfy this property as \(P_{1 \rightarrow 1}\).

Next, we denote by \(P_{0 \rightarrow 0}\) the number of permutations in \(\Pi_{D^0_n}\) where adding \(f\) changes the result of \(q_{RS-T}\) from true to false. We have \(N + 1\) endogenous facts in \(D^0_n\), so we conclude that:

\[
P_{0 \rightarrow 0} + P_{1 \rightarrow 1} + P_{1 \rightarrow 0} = (N + 1)!
\]

Therefore, it holds that:

\[
P_{1 \rightarrow 0} = (N + 1)! - (P_{0 \rightarrow 0} + P_{1 \rightarrow 1})
\]

By the definition of the Shapley value for database facts (Definition 2) we obtain that:

\[
\text{Shapley}(D^0, q_{RS-T}, f) = \frac{P_{0 \rightarrow 0} \cdot 0 + P_{1 \rightarrow 1} \cdot 0 + P_{1 \rightarrow 0} \cdot (-1)}{(N + 1)!}
\]

\[
= \frac{[(N + 1)! - (P_{0 \rightarrow 0} + P_{1 \rightarrow 1})] \cdot (-1)}{(N + 1)!} = \frac{P_{0 \rightarrow 0} + P_{1 \rightarrow 1}}{(N + 1)!} - 1
\]

Hence, we are able to compute \(P_{1 \rightarrow 1}\) from the Shapley value of \(f\) in the following way:

\[
P_{1 \rightarrow 1} = (\text{Shapley}(D^0, q_{RS-T}, f) + 1) \cdot (N + 1)! - \frac{(N + 1)!}{m + 1}
\]

In the next step, we build \(N + 1\) instances \((D^r, f)\) for every \(r \in \{1, \ldots, N + 1\}\) as follows: the database \(D^r\) will contain an endogenous fact \(R(a)\) for every vertex \(a \in A\), an endogenous fact \(T(b)\) for every vertex \(b \in B\), an exogenous fact \(S(a, b)\) for every edge \((a, b) \in E\), and the endogenous fact \(T(0)\). In addition, for every \(i \in \{1, \ldots, r\}\), the database \(D^r\) will contain an endogenous fact \(R(0_1)\) and the exogenous fact \(S(0_1, 0)\).

Once again, we consider the two types of permutations \(\sigma \in \Pi_{D^r_n}\), where adding \(f\) does not change the result of \(q_{RS-T}\) from true to false:

1. \((D^r_x \cup \sigma_f) \models q_{RS-T}\) and \((D^r_x \cup \sigma_f \cup \{f\}) \models q_{RS-T}\). In this case, the set \(F = \{a \mid R(a) \in \sigma_f, a \neq 0_1\} \cup \{b \mid T(b) \in \sigma_f\}\) of vertices in \(g\) is such that \(F \notin S(g)\). Otherwise, we have that the only pairs of facts from \(R\) and \(S\) satisfying \(q_{RS-T}\) are of the form \(R(0_1), S(0_1, 0)\), which is a contradiction to the fact that \((D^r_x \cup \sigma_f \cup \{f\}) \models q_{RS-T}\), since \(f\) is violating each one of these homomorphisms (i.e., \(\{R(0_1), S(0_1, 0), T(0)\} \neq q_{RS-T}\)). The number of permutations satisfying
this property is: \( P^r_{1 \to 1} = P_{1 \to 1} \cdot m_r \), where \( m_r = \binom{N+r+1}{r} \cdot r! \), as the \( r \) endogenous facts of the form \( R(0_i) \) can be added to every such permutation in every possible position without affecting the result of the query.

2. \( (D'_x \cup \sigma_f) \not\models q_{RS-T} \) and \( (D'_x \cup \sigma_f \cup \{ f \}) \not\models q_{RS-T} \). To ensure that \( (D'_x \cup \sigma_f) \not\models q_{RS-T} \), none of the facts \( R(0_i) \) can appear in \( \sigma_f \) (or otherwise, the pair \( \{ R(0_i), S(0_i, 0) \} \) would satisfy the query). Furthermore, The set \( F = \{ a \mid R(a) \in \sigma_f, a \neq 0_i \} \cup \{ b \mid T(b) \in \sigma_f \} \) must be such that \( F \subseteq S(g) \), or otherwise, there is \( S(a, b) \in D'_x \) such that \( R(a) \in \sigma_f \) and \( T(b) \notin \sigma_f \), which implies that \( (D_x \cup \sigma_f) \models q_{RS-T} \), in contradiction to our assumption. The number of such permutations is: \( P_{0 \to 0}^r = \sum_{k=0}^{N} |S(g, k)| \cdot k! \cdot (N - k + r)! \).

Hence, we have that:

\[
\text{Shapley}(D^r, q_{RS-T}, f) = \frac{P_{0 \to 0}^r + P_{1 \to 1}^r}{(N + r + 1)!} - 1 = \frac{P_{1 \to 1} \cdot m_r + P_{0 \to 0}^r}{(N + r + 1)!} - 1
\]

We get that:

\[
P_{0 \to 0}^r = (\text{Shapley}(D^r, q_{RS-T}, f) + 1) \cdot (N + r + 1)! - P_{1 \to 1} \cdot m_r = \sum_{k=0}^{N} |S(g, k)| \cdot k! \cdot (N - k + r)!
\]

(Recall that \( P_{1 \to 1} \) can be computed from the Shapley value of the fact \( T(0) \) in the instance \( D^0 \).) As a consequence, we get a system of \( N + 1 \) equations:

\[
\begin{pmatrix}
0!(N + 1)! & 1!N! & \ldots & N!1! \\
0!(N + 2)! & 1!(N + 1)! & \ldots & N!2! \\
\vdots & \vdots & \ddots & \vdots \\
0!(2N + 1)! & 1!(2N)! & \ldots & N!(N + 1)! \\
\end{pmatrix}
\begin{pmatrix}
|S(g, 0)| \\
|S(g, 1)| \\
\vdots \\
|S(g, N)| \\
\end{pmatrix}
= \begin{pmatrix}
(\text{Shapley}(D^1, q_{RS-T}, f) + 1) \cdot (N + 2)! - P_{1 \to 1} \cdot m_1 \\
(\text{Shapley}(D^2, q_{RS-T}, f) + 1) \cdot (N + 3)! - P_{1 \to 1} \cdot m_2 \\
\vdots \\
(\text{Shapley}(D^{N+1}, q_{RS-T}, f) + 1) \cdot (2N + 2)! - P_{1 \to 1} \cdot m_{N+1}
\end{pmatrix}
\]

This is the same system of equations that Livshits et al. obtained in the hardness proof for \( q_{RS-T} \) [LBKS20]. There, they prove that the determinant of the coefficient matrix is not zero; hence, this system is solvable in polynomial time, providing us with the value \( |S(g)| = \sum_{k=0}^{N} |S(g, k)| \).

Finally, it is left to prove that \( |S(g)| = |IS(g)| \). For that purpose, we define a bijection between the two sets, \( h : IS(g) \to S(g) \), as follows: Let \( (A' \cup B') \in IS(g) \). Then, \( h(A' \cup B') = A' \cup (B \setminus B') \). Note that for every \( (a, b) \in E \) we have that if \( a \in A' \)
then \( b \notin B' \); hence, for every \((a, b) \in E\) it holds that if \( a \in A' \) then \( b \in (B \setminus B')\). Hence, if \((A' \cup B') \in IS(g)\), then \((A' \cup (B \setminus B')) \in S(g)\).

Injectivity: Let \( I_1 = (A'_1 \cup B'_1) \) and \( I_2 = (A'_2 \cup B'_2) \) be two distinct independent sets of \( g \) (i.e., \( I_1, I_2 \in IS(g) \)). At least one of the following holds: \( A'_1 \neq A'_2 \), or \( B'_1 \neq B'_2 \). Clearly in both cases we have that \( h(I_1) \neq h(I_2) \) as well.

Surjectivity: Let \( E = (A' \cup B') \) be a subset of vertices in \( S(g) \). Consider the subset \( I = (A' \cup (B \setminus B')) \). By the definition of \( S(g) \), for every \((a, b) \in E\) we have that if \( a \in A' \) then \( b \in B' \). Therefore, for every \((a, b) \in E\) it holds that if \( a \in A' \) then \( b \notin (B \setminus B') \). Then, we conclude that \( I \in IS(g) \) by definition. It holds that \( h(I) = (A' \cup (B \setminus (B \setminus B'))) = (A' \cup B') = E \); thus, the function \( h \) is surjective.

To conclude, we constructed a reduction from the problem of computing \(|IS(g)|\) to that of computing \( \text{Shapley}(D, q_{RS-T}, f) \); hence, computing \( \text{Shapley}(D, q_{RS-T}, f) \) is \( \text{FP}^{NP} \)-complete.

**Hardness for Every Non-Hierarchical Query**

Finally, we show that for any non-hierarchical self-join-free CQ\(^{-}\) \( q \) the computation of \( \text{Shapley}(D, q, f) \) is \( \text{FP}^{NP} \)-complete. For that, we use a reduction from the problem of computing \( \text{Shapley}(D', q', f') \) where \( q' \) is one of queries: \( q_{RST}, q_{RS-T}, q_{RS-ST}, \) or \( q_{RS-T} \), depending on the polarity of the atoms in the non-hierarchical triplet in \( q \).

**Lemma 3.1.5.** If \( q \) is a non-hierarchical CQ\(^{-}\) without self-joins, then computing \( \text{Shapley}(D, q, f) \) is \( \text{FP}^{NP} \)-complete.

**Proof.** Every non-hierarchical self-join-free CQ\(^{-}\) contains three atoms \( \alpha_x, \alpha_y, \alpha_{x,y} \) where \( x, y \in \text{Vars}(q) \), such that \( \alpha_x \in A_x \setminus A_y, \alpha_y \in A_y \setminus A_x, \alpha_{x,y} \in A_x \cap A_y \). We argue that \( q \) satisfies another property: if there is a non-hierarchical triplet \( \alpha_x, \alpha_y, \alpha_{x,y} \) where \( \alpha_{x,y} \) and at least one of \( \alpha_x \) or \( \alpha_y \) are negative, then there is another non-hierarchical triplet \( \alpha'_x, \alpha'_y, \alpha'_{x,y} \) where either \( \alpha'_{x,y} \) is positive or both \( \alpha_x \) and \( \alpha_y \) are positive. Assume, without loss of generality, that \( \alpha_x \) is negative. Since \( q \) is safe, there is a positive atom \( \alpha'_x \) such that \( \alpha'_x \in A_x \). If there exists such an atom \( \alpha'_x \) such that \( \alpha'_x \in (A_x \cap A_y) \), the triplet \( \alpha_x, \alpha'_x, \alpha_y \) satisfies the property. Otherwise, if \( \alpha_y \) is positive, the triplet \( \alpha'_x, \alpha_{x,y}, \alpha_y \) satisfies the property. Finally, if every \( \alpha'_x \) is such that \( \alpha'_x \in A_x \setminus A_y \) and \( \alpha_y \) is negative, since \( q \) is safe, we have another positive atom \( \alpha'_y \in A_y \setminus A_x \) and \( \alpha'_{x,y}, \alpha'_y \) is a non-hierarchical triplet satisfying the property.

Let \( \alpha_x, \alpha_{x,y}, \alpha_y \) be a non-hierarchical triplet of \( q \) that satisfies the above property. We construct a reduction from the problem of computing \( \text{Shapley}(D', q', f') \) where \( q' \) is one of \( q_{RST}, q_{RS-T}, q_{RS-ST}, \) or \( q_{RS-T} \) to computing \( \text{Shapley}(D, q, f) \). We have already established that computing \( \text{Shapley}(D', q', f') \) for each of these queries is \( \text{FP}^{NP} \)-complete; hence, we conclude that \( \text{Shapley}(D, q, f) \) is \( \text{FP}^{NP} \)-complete for any non-hierarchical self-join-free CQ\(^{-}\). We present the four reductions simultaneously, as they all work in a very similar way.
Depending on the polarity of the atoms in the non-hierarchical triplet of \( q \) satisfying the property indicated above, we select one of the four reductions (if there are multiple triplets satisfying this property, we choose one randomly):

1. If all three atoms are positive, we reduce from computing \( \text{Shapley}(D', q_{RST}, f') \).
2. If \( \alpha_{x,y} \) is positive while \( \alpha_x, \alpha_y \) are negative, we reduce from \( \text{Shapley}(D', q_{\neg RST}, f') \).
3. If \( \alpha_{x,y} \) is negative while \( \alpha_x, \alpha_y \) are positive, we reduce from \( \text{Shapley}(D', q_{RS \neg T}, f') \).
4. If \( \alpha_{x,y} \) is positive, and one of \( \alpha_x, \alpha_y \) is negative, we reduce from \( \text{Shapley}(D', q_{RS \neg T}, f') \).

The idea is very similar to the corresponding proof in [LBKS20]. The main difference is in the construction of the database \( D \), as \( q \) may contain negative atoms. We use the atom \( \alpha_x \) to represent the atom \( R(x) \) (or \( \neg R(x) \)) in \( q' \), the atom \( \alpha_y \) to represent the atom \( T(y) \) (or \( \neg T(y) \)) in \( q' \), and the atom \( \alpha_{x,y} \) to represent the atom \( S(x, y) \) (or \( \neg S(x, y) \)) in \( q' \). For every fact \( R(a) \) in \( D' \) (which is the input to the first problem), we insert to the relation \( R_{\alpha_x} \) in \( D \) (the input to our problem) a fact obtained by mapping the variable \( x \) in \( \alpha_x \) to \( a \), the rest of the variables to a constant \( \odot \), and every constant \( c \) in \( \alpha_x \) to itself. Similarly, for every fact \( T(b) \) in \( D \), we insert to the relation \( R_{\alpha_y} \) in \( D \) a fact obtained by mapping the variable \( y \) in \( \alpha_y \) to \( b \), the rest of the variables to a constant \( \odot \), and every constant \( c \) in \( \alpha_y \) to itself. Each such fact \( f \in D \) will be endogenous if and only if its corresponding fact \( f' \) is endogenous in \( D' \). Finally, for every fact \( S(a, b) \) and a positive atom \( \alpha \) in \( q \) that is not one of \( \alpha_x, \alpha_y \), or \( \alpha_{x,y} \), we insert to the relation \( R_{\alpha} \) in \( D \) an exogenous fact obtained by mapping the variable \( x \) in \( \alpha \) to \( a \), the variable \( y \) to \( b \), the rest of the variables to \( \odot \), and every constant \( c \) in \( \alpha \) to itself. We also add to the relation \( R_{\alpha_{x,y}} \) in \( D \) (whether \( \alpha_{x,y} \) is a positive atom or not) a fact obtained by this mapping, and it will be exogenous if and only if \( S(a, b) \) is exogenous in \( D' \).

Note that \( |D_n| = |D'_n| \), since each endogenous fact in \( D' \) was mapped to a single endogenous fact in \( D \); hence, the total number of permutations of the endogenous facts is equal for both databases, and we only need to show that for every fact \( f \in D_n \), the number of permutations of the facts in \( D_n \) where adding \( f \) changes the result of \( q \) is equal to the number of permutations of the facts in \( D'_n \) where adding \( f' \) (which is the fact that \( f \) was generated from) changes the result of \( q' \). We can then conclude that \( \text{Shapley}(D, q, f) = \text{Shapley}(D', q', f') \).

Consider a permutation \( \sigma \in \Pi_{D_n} \). It is the case that \( q(D_x \cup \sigma_f) \neq q(D_x \cup \sigma_f \cup \{f\}) \) if and only if there is a homomorphism \( h \) from \( q \) to \( D_x \cup \sigma_f \cup \{f\} \) such that the atom \( \alpha \) of \( q \) for which \( f \in R_\alpha \) is mapped to \( f \), if \( \alpha \) is a positive atom, or \( h \) is a homomorphism from \( q \) to \( D_x \cup \sigma_f \) which is violated by \( f \), if \( \alpha \) is negative. By the construction of \( D \), we deduce that \( h(x) = a, h(y) = b \), and for any other variable \( w \) of \( q \) \( h(w) = \odot \), such that \( a, b \) are values in the domain of \( D' \). Consider the permutation \( \sigma' \in \Pi_{D'_n} \), such that \( \sigma_f \) is the set of facts obtained by the facts in \( \sigma'_f \). The homomorphism \( h \) as described
above exists if and only if there exists a homomorphism \( h' \) form \( q' \) to \( D'_x \cup \sigma'_j \cup \{ f' \} \) where \( h'(x) = a, h'(y) = b, \) and \( \alpha' \) (the atom in \( D' \) that \( \alpha \) represents) is mapped to \( f' \), if \( \alpha' \) is positive, or \( h' \) is a homomorphism from \( q' \) to \( D'_x \cup \sigma'_j \), which is violated by \( f' \), if \( \alpha' \) is negative. This holds true since we have added to the relations in \( D \) associated with positive atoms that are not one of \( \alpha_x, \alpha_y, \) or \( \alpha_{x,y} \), all the necessary facts (as exogenous facts) so that every homomorphism from \( q' \) to \( D' \) will correspond to a homomorphism from \( q \) to \( D \). Moreover, the relations in \( D \) associated with negative atoms of \( q \) (except for \( \alpha_{x,y} \)) are empty and do not affect the query result. Finally, for the permutation \( \sigma \) it holds that \( q(D_x \cup \sigma_f) \neq q(D_x \cup \sigma_f \cup \{ f \}) \) if and only if it holds that \( q'(D'_x \cup \sigma'_f) \neq q'(D'_x \cup \sigma'_f \cup \{ f' \}) \) for the corresponding permutation \( \sigma' \), and that concludes our proof.

\[ \fbox{ } \]

### 3.2 Queries with Self-Joins

In this section we aim to extend our knowledge about the complexity of computing the Shapley value for CQ\(^{-}\)s, beyond the case of self-join-free queries.

#### Hard Queries with Self-Joins

The proof of Theorem 3.1 that was introduced in the previous section heavily relies on the assumption that the query is self-join-free. However, our hardness results for the basic non-hierarchical queries \( q_{RST}, q_{RS-T}, q_{R-ST} \) and \( q_{RS-T} \) can be generalized to certain CQ\(^{-}\)s with self-joins, by replacing the atom over the relation \( T \) with another atom over the relation \( R \) (e.g., we can prove hardness for the query \( -R(x), S(x, y), \neg R(y) \)). This can be proved using a reduction from the corresponding self-join-free query (e.g., the query \( -R(x), S(x, y), \neg T(y) \)) by assuming, without loss of generality, that the values in the domain of \( R^D \) and the values in the domain of \( T^D \) are disjoint. In fact, this result can be generalized to a larger class of CQ\(^{-}\)s with self-joins, as we show in the next theorem:

**Theorem 3.2.** Let \( q \) be a polarity-consistent CQ\(^{-}\) containing a non-hierarchical triplet \((\alpha_x, \alpha_{x,y}, \alpha_y)\) such that the relation \( R_{\alpha_{x,y}} \) occurs only once in \( q \). Then, computing Shapley(\( D, q, f \)) is \( \text{FP}^{\#P} \)-complete.

Recall that \( q \) is polarity consistent if every relation that appears in \( q \) appears only in positive atoms or only in negative atoms. To prove the theorem, we construct a reduction from the problem of computing Shapley(\( D', q', f' \)) where \( q' \) is one of \( q_{RST}, q_{RS-T} \) or \( q_{RS-T} \) (depending on the polarity of the atoms \( \alpha_x \) and \( \alpha_y \)) under the following assumptions: (1) all the facts of \( S \) are exogenous, and (2) for every fact \( S(a, 1) \) in \( D \), both facts \( R(a) \) and \( T(1) \) are in \( D \). The instances constructed in the proofs of hardness for all three queries satisfy these conditions; hence, the problems remain hard under these assumptions. We also assume for simplicity that the set of values used in the facts of \( R^D \) and the set of values used in the facts of \( T^D \) are disjoint.
The idea is very similar to the construction in the proof of Lemma 3.1.5. The main difference is that we may add facts to the same relation for different (positive) atoms, when self-joins are involved. In case that any of the relations $R_{\alpha_x}, R_{\alpha_y}$ occur in atoms of $q$ more than once, and we are needed to add the same fact to this relation for every such atom (e.g., when $\alpha_x = R(x,z)$, and $R(x,w)$ is another atom of $q$), this fact will be endogenous if and only if the corresponding fact in $D'$ is endogenous. We again use the atom $\alpha_x$ to represent the atom $(-)R(x)$ in $q'$, the atom $\alpha_y$ to represent the atom $(-)T(y)$ in $q'$, and the atom $\alpha_{x,y}$ to represent the atom $S(x,y)$ in $q'$. We use the assumption that the relation $R_{\alpha_{x,y}}$ occurs only once in $q$ to ensure that we do not create new “connections” between values of $x$ and values of $y$. If $\alpha_x$ and $\alpha_y$ are both positive, the reduction is from the problem of computing Shapley$(D', q_{\text{RS-T}}, f')$, if both atoms are negative, the reduction is from computing Shapley$(D', q_{\text{-RS-T}}, f')$, and if one atom is positive while the other is negative, the reduction is from computing Shapley$(D', q_{\text{RS-T}}, f')$.

Formally, given an input database $D'$ to the first problem, we build a database $D$ in the same way we built it in the proof of Lemma 3.1.5, except for the treatment of the atom $\alpha_{x,y}$. Since the atom $S(x,y)$ is always positive in $q'$, if $\alpha_{x,y}$ is negative, then we insert to the relation $R_{\alpha_{x,y}}$ in $D$ an exogenous fact $f$ obtained by mapping the variables $x$ and $y$ in $\alpha_{x,y}$ to some values $c_1$ and $c_2$ (from the domain of $D'$), respectively, the rest of the variables to $\odot$ and each the constant to itself if and only if $S(c_1,c_2) \notin D'$. If $\alpha_{x,y}$ is positive, then we insert $f$ to $D$ if and only if $S(c_1,c_2) \in D'$. Note that endogenous facts appear only in $R_{\alpha_x}, R_{\alpha_y}$, and that for every $f'' \in D_n$, there is a single matching fact $f \in D_n$.

We will now prove that for every endogenous fact $f''$ in $D'$ and its corresponding fact $f$ in $D$ (i.e., the fact that was generated from $f''$) it holds that Shapley$(D', q', f'') = \text{Shapley}(D, q, f)$ (recall that $q'$ is one of $q_{\text{RS-T}}, q_{\text{-RS-T}}, q_{\text{RS-T}}$). We start by proving the following.

**Lemma 3.2.1.** Let $E' \subseteq D'_n$ and let $E$ be the set of corresponding facts in $D_n$. If $(D'_x \cup E') \models q'$ then $(D_x \cup E) \models q$.

**Proof.** Since $(D'_x \cup E') \models q'$ there is a mapping $h'$ from the variables of $q'$ to the domain of $D'$ where $h'(x) = a$ for some value $a$ from the domain of $R^{D'}$ and $h'(y) = 1$ for some value 1 from the domain of $T^{D'}$ such that $h'$ maps every positive atom and none of the negative atoms of $q'$ to a fact of $D'_x \cup E'$. We claim that the mapping $h$ such that $h(x) = h'(x), h(y) = h'(y)$, and $h(w) = \odot$ for the rest of the variables, maps every positive atom and none of the negative atoms of $q$ to facts of $D_x \cup E$; hence $(D_x \cup E) \models q$.

As in the proof of Lemma 3.1.5, from the construction of $D$, we have that every positive atom of $q$ is mapped by $h$ to a fact of $D_x \cup E$. The relations associated with negative atom of $q$, except for $R_{\alpha_x}, R_{\alpha_y}$, and $R_{\alpha_{x,y}}$ are empty and do not affect the result of the query (recall that $q$ is polarity-consistent; hence, a relation that appears as a negative atom cannot appear as a positive atom as well). Moreover, the relation
\(R_{\alpha_{x,y}}\) contains the fact obtained from \(\alpha_{x,y}\) using the mapping \(h\) in \(D_x\) if and only if 
\(S(a,1) \in D'_x\).

It is only left to show that there is no negative atom of \(q\) that is mapped by \(h\) to a fact in \(D_n\). Let us assume, by way of contradiction, that a negative atom \(\beta\) of \(q\) is mapped by \(h\) to a fact \(\tilde{f}\) in \(D_n\). Observe that the relation associated with the atom \(\beta\) must be \(R_{\alpha_x}\) or \(R_{\alpha_y}\), since any other relation that is associated with negative atoms are empty. Assume, without loss of generality, that \(\tilde{f}\) belongs to the relation \(R_{\alpha_x}\) in \(D\). Since \(q\) is polarity-consistent, the atom \(\alpha_x\) is a negative atom as well. Moreover, in this case, the relation \(R\) appears as a negative atom in \(q'\). From the construction of \(D\), every endogenous fact in the relation \(R_{\alpha_x}\) in \(D\) is obtained by a mapping the variables of \(\alpha_x\) such that the variable \(x\) is mapped to a value from the domain of \(R^D\), the variable \(y\) to a value from the domain of \(T^D\), and the rest of the variables to \(\odot\). Hence, if \(h\) maps \(\beta\) to \(\tilde{f}\), then \(h\) maps the atom \(\alpha_x\) to the fact \(\tilde{f}\) as well. Then, from the construction of \(D\), we have that the corresponding fact of \(\tilde{f}\) in \(D'_n\) is \(R(a) \in E'\), which is a contradiction to the fact that \(R\) appears as a negative atom in \(q'\) and \((D'_x \cup E') \models q'\). \(\blacksquare\)

Next, we prove the following.

**Lemma 3.2.2.** Let \(E' \subseteq D'_n\) and let \(E\) be the set of corresponding facts in \(D_n\). If \((D'_x \cup E') \not\models q'\) then \((D_x \cup E) \not\models q\).

**Proof.** Let us assume, by way of contradiction, that \((D_x \cup E) \models q\). Then, there is a mapping \(h\) from the variables of \(q\) to the domain of \(D\) that maps every positive atom and none of the negative atoms of \(q\) to a fact in \(D_x \cup E\). In particular, the atom \(\alpha_{x,y}\), is mapped to a fact of \(D_x \cup E\) if and only if it is positive. From the construction of \(D\) and the uniqueness of the atom (i.e., the fact that its relation does not appear in another atom of \(q\)), we have that if \(\alpha_{x,y}\) is positive, then \(h\) maps \(\alpha_{x,y}\) to a fact of \(D_x\) if and only there exists a fact \(S(a,1)\) in \(D\) such that \(h(x) = a\) and \(h(y) = 1\). If \(\alpha_{x,y}\) is negative, then \(h\) does not map \(\alpha_{x,y}\) to a fact of \(D_x\) if and only if there exists a fact \(S(a,1)\) in \(D\) such that \(h(x) = a\) and \(h(y) = 1\).

We claim that the mapping \(h\) is such that every positive atom and none of the negative atoms of \(q'\) is mapped to a fact of \(D'_x \cup E'\), which is a contradiction to the fact that \((D'_x \cup E') \not\models q'\). We have already established that there exists an exogenous fact \(S(a,1)\) in \(D'\) assuming that \(h(x) = a\) and \(h(y) = 1\). It is only left to show that the fact \(R(a)\) belongs to \(D'_x \cup E'\) if and only if \(R\) occurs as a positive atom in \(q'\), and, similarly, the fact \(T(1)\) belongs to \(D'_x \cup E'\) if and only if \(T\) occurs as a positive atom in \(q'\).

If \(\alpha_x\) is a positive atom, then there is a fact \(\tilde{f}\) in the relation \(R_{\alpha_x}\) in \(D\) obtained from \(\alpha_x\) by mapping the variable \(x\) to the value \(a\) and the rest of the variables to the value \(\odot\), such that \(\tilde{f} \in D_x \cup E\). In this case, by the construction of \(D\), it holds that \(\tilde{f} \in D_n\) if and only if the fact \(R(a)\) corresponding to \(\tilde{f}\) is in \(D'_n\); hence, \(\tilde{f} \in E\) implies that \(R(a) \in E'\). If \(\alpha_x\) is a negative atom (in which case, the relation \(R\) occurs in \(q'\) as a negative atom), then such fact \(\tilde{f}\) does not appear in \(D_x \cup E\) (or, otherwise, \(D_x \cup E\) will
not satisfy \( q \), which implies that the fact \( R(a) \) does not appear in \( D'_x \cup E' \). We can similarly show that the fact \( T(1) \) appears in \( D'_x \cup E' \) if and only if its corresponding fact appears in \( D_x \cup E \), and that concludes our proof.

Lemmas 3.2.1 and 3.2.2 imply that an endogenous fact \( f' \) changes the result of \( q' \) in a permutation \( \sigma' \) of \( D'_n \) if and only if the fact \( f \) changes the result of \( q \) in a permutation \( \sigma \) of \( D_n \). Since the total number of permutations of the facts in \( D_n \) and \( D'_n \) is equal, we deduce that indeed \( \text{Shapley}(D, q, f) = \text{Shapley}(D', q', f') \), and that concludes the proof of hardness.

**A Tractable Fragment**

While self-joins preserve hardness in some cases where a non-hierarchical triplet of atoms occurs in the query, it is unclear whether a tractable (non trivial) query with self-joins even exists. Self-joins impose a new structure of the problem that requires finding new approaches to compute the Shapley value. We now state a positive result for a special case of \( \text{CQ}^- \)'s that consists of two atoms of the same relation, a positive atom and a negative atom. In addition, we assume that the sets of variables that occur in both atoms are equal. For example, consider the query \( q() : \preceq \text{FOLLOWS}(x, y), \neg\text{FOLLOWS}(y, x) \), asking if there exist two users such that \( x \) follows \( y \) but \( y \) does not follow \( x \). We can measure the contribution of each fact to the existence of asymmetrical relationships in \( D \), by computing \( \text{Shapley}(D, q, f) \).

**Theorem 3.3.** Let \( q \) be a \( \text{CQ}^- \) that consists of two atoms of the same relation, \( \alpha \) and \( \beta \), such that \( \alpha \) is positive, \( \beta \) is negative, and \( \text{Vars}(\alpha) = \text{Vars}(\beta) \). Then, \( \text{Shapley}(D, q, f) \) can be computed in polynomial time.

**Proof.** For simplicity, we first assume that \( q \) has no constants. Consider a query \( q() : \preceq R(x_1, \ldots, x_m), \neg R(x_1, \ldots, x_m) \) where \( \text{Vars}(\alpha) = \text{Vars}(\beta) \). That is, \( i_1, \ldots, i_m \) is a permutation of \( 1, \ldots, m \), indicating that the position of the \( j \)-th variable of \( \beta \) is the \( i_j \)-th variable of the atom \( \alpha \).

Consider an arbitrary mapping \( h(x_i) = c_i \), where \( c_i \in \text{Dom}(D) \) for every \( i \in \{1, \ldots, m\} \). Let \( f = R(c_1, \ldots, c_n) \) be a fact obtained from the atom \( \alpha \) by this mapping, and let \( f' = R(c_1, \ldots, c_n) \) be a fact obtained from \( \beta \) by this mapping. Observe that the mapping \( h \) is a homomorphism from \( q \) to \( D \) if \( f \) is a fact in \( D \), unless \( f' \) is a fact in \( D \) as well. We refer to \( f \) as a *satisfying fact*, and to \( f' \) as the *violating fact* of \( f \). Note that \( f \) may be the violating fact of another fact in the database as well, by the same arguments.

As we mentioned in the previous section, the problem of computing \( \text{Shapley}(D, q, f) \) (for any Boolean query) can be reduced to computing \( \text{Sat}(D, q, k) \). Therefore, we now show how to compute \( \text{Shapley}(D, q, k) \) for every \( k = 0, 1, \ldots, |D_n| \). We start with computing the size of the compliment set to \( \text{Sat}(D, q, f) \) (i.e., the number of \( k \)-subsets
of \(D_n\) that do not satisfy \(q\) along with \(D_x\). In the next step, we subtract this number from the total number of \(k\)-subsets of \(D_n\), which is \(\binom{|D_n|}{k}\).

For every subset \(E \subseteq D_n\), it holds that \((D_x \cup E) \not\models q\) only if for every fact \(f \in (D_x \cup E)\), its violating fact \(f'\) is in \(D_x \cup E\) as well. We define \(G_{q,R}\) to be a directed graph where every fact in \(R^D\) is a vertex, and there is an edge \((f', f)\) in \(G_{q,R}\) if \(f'\) is the violating fact of \(f\). In this graph, there is at most one edge going in to each vertex, and at most one edge going out. If \((D_x \cup E) \not\models q\), then every vertex corresponding to a fact of \(D_x \cup E\) must have an edge going in from another vertex of \(D_x \cup E\). Hence, we deduce that the sub-graph induced by \(D_x \cup E\) is a set of disjoint cycles. We now prove that the length every cycle in \(G_{q,R}\) is at most \(m\). Assume by the way of contradiction that there is a cycle of length \(l > m\). Then, there are \(l\) different facts \(f_1, \ldots, f_l\) in \(R^D\) such that \(f_i\) is violating \(f_{i+1}\) for every \(i = 1, \ldots, l-1\), and \(f_l\) is violating \(f_1\). By the structure of \(q\), the values in the positions \(1, \ldots, m\) in \(f_i\) are the values in the positions \(i_1, \ldots, i_m\) in \(f_{i+1}\) respectively. Hence, there is a cyclic transformation in the positions of these values from \(f_i\) to \(f_{i+1}\). Since \(\text{arity}(R) = m\), we deduce that after \(m\) transformations at most, the positions of the values returns to the same as it was in \(f_1\), in a contradiction to the assumption that \(l > m\).

Next, we show how to compute the number of \(k\)-subsets \(E\) for which \((D_x \cup E) \not\models q\). First, for every exogenous fact \(f\) which is on a cycle \(C\) in \(G_{q,R}\), all other facts that belong to \(C\) must be in \(D_x \cup E\) as well. Assuming otherwise would imply that there is a fact in \(D_x \cup E\) which is not violated. We refer these cycles as exogenous cycles, and denote the number of endogenous facts that appear on exogenous cycles as \(E_c\). If \(E_c > k\), then the number of subsets is clearly zero. Otherwise, \(E\) must contain an additional \(k - E_c\) facts among the endogenous facts, that jointly cannot satisfy \(q\); thus, each of these facts must be a part of a cycle in the graph, or else, its ancestor will satisfy \(q\). We denote the cycles in \(G_{q,R}\) that are not exogenous as \(C_1, \ldots, C_l\). If one of the facts that belong to \(C_i\) is in \(E\), then all other facts in \(C_i\) must be in \(E\) as well, following the same argument as aforementioned. We denote the number of cycles of length \(j\) in \(C_1, \ldots, C_l\) as \(n_j\) for each \(j \in \{1, \ldots, m\}\) (recall that every cycle length is at most \(m\), which we consider as a constant under data complexity assumption). Hence, we get the following equation:

\[
y_1 + 2 \cdot y_2 + \ldots + m \cdot y_m = k - E_c
\]

and we search for every solutions to this equation where for each \(j\), \(0 \leq y_j \leq n_j\) and \(y_j\) is a non-negative integer. Every \(y_j\) will represent the number of cycles of length \(j\), for which every fact that belong to these cycles is in \(E\). The solutions to this equation can be computed by considering all options for choosing a number in the range \(0, 1, \ldots, k - E_c\), to be assigned for every variable \(y_j\). For a certain assignment to \(y_1, \ldots, y_m\) that satisfy the equation, there are \(\binom{m}{y_1} \cdot \binom{m}{y_2} \cdot \ldots \cdot \binom{m}{y_m}\) possible \(k\)-subsets of \(D_n\) that will not satisfy the query along with \(D_x\). Finally, we sum the numbers of
subsets that were calculated for every feasible solution, and we get the compliment to the number of $\text{Sat}(D, q, k)$. 

**Allowing constants.** The idea of the proof is similar when we allow constants to appear in $q$. The main difference is that a fact in $D$ may not satisfy the query, even if its violating fact is not in the database (e.g., the fact $R(c, 2)$ cannot satisfy the query $q(\cdot) \equiv R(x, 1), \neg R(2, x)$). The vertices in $G_{q,R}$ corresponding to the non satisfying facts cannot have an edge going in, by the definition of $G_{q,R}$. Moreover, we can add the non satisfying facts to $E$, and still have that $(D \cup E) \not\models q$. We refer to a sub-path that starts with a non satisfying fact $f$, and ends with an exogenous fact as an *exogenous sub-path*. We must add to $E$ every endogenous fact along the exogenous sub-paths, otherwise, we get that $(D \cup E) \models q$; we denote the number of these endogenous facts as $E_p$. As for the rest of the endogenous facts that appear on paths that starts with a non satisfying fact $f$, we can add these facts to $E$, if every ancestor along the path is in $E$ as well. Observe that the length of every such path is at most $m$, following the same arguments proving that every cycle in $G_{q,R}$ is of length at most $m$. We denote as $l_i$ the number of sub-paths in $G_{q,R}$ starting with a non-satisfying fact such that: (1) the path is of length $i$ and has no exogenous fact on it, or (2) after removing exogenous sub-paths, the remaining path is of length $i$. Note that we can count the number of those paths easily, as all of these paths are disjoint. Next, we compute the number of $k'$-subsets of endogenous facts that do not satisfy $q$ as following. We consider another equation:

$$z_{1,1} + z_{2,1} + 2 \cdot z_{2,2} + \ldots + (m - 1) \cdot z_{m,m-1} + m \cdot z_{m,m} = k' - E_p$$

and search for every solution where it holds that $\sum_{j=1}^{i} z_{i,j} \leq l_i$ for each $i$, and $z_{i,j}$ is a non-negative integer. Every will $z_{i,j}$ represent the number of paths of length $i$, in which the first $j$ facts are in $E$. The solutions to this equation can be found in a similar way to the one used for finding the solutions to the previous equation, by considering all the possible assignments to the variables in the range $0, 1, \ldots, k' - E_p$ that satisfy the constrains (there are $O(k'^m)$ options to check). For each feasible solution to this equation, the number of $k'$-subsets is: $(\binom{l_1}{z_{1,1}} \cdot \binom{l_2}{z_{2,1}} \cdot \cdots \cdot \binom{l_m}{z_{m,1}} \cdot \cdots \cdot z_{m,m})$, by using the multinational coefficient. Next, we sum the number of subsets for every possible solution, and we get the number of $k'$-subsets of endogenous facts that do not satisfy $q$ along with $D_x$. 

In the general case of computing the compliment of $\text{Sat}(D, q, k)$, we will go over every $k_1, k_2$ such that $k_1 + k_2 = k - E_p - E_c$. In each case, we first compute the number of $k_1$-subsets of endogenous facts in $D_n$ that appear on cycles in $G_{q,R}$ that do not satisfy $q$ (along with facts in $D_x$ that are on cycles). Then, we compute the number of $k_2$-subsets of endogenous facts in $D_n$ that appear on paths in $G_{q,R}$ starting with non satisfying facts, that do not satisfy $q$ (along with facts in $D_x$ that are on such paths). By that we cover all possible options, and we obtain $\text{Sat}(D, q, k)$. ■
Algorithm 3.1 SingleSjNeg \((D, q, f)\)

Build the graph \(G_{q,R}\)

\[ C_1, ..., C_n \leftarrow \text{cycles in } G_{q,R} \]

\[ P_1, ..., P_r \leftarrow \text{paths in } G_{q,R} \text{ starting with a non-satisfying fact} \]

\[ P'_1, ..., P'_l \leftarrow \text{paths in } G_{q,R} \text{ starting with a satisfying fact} \]

if exists \(f \in D_x\) on \(P'_1, ..., P'_l\) then

return \((|D_n|)_k\)

\[ E_c \leftarrow \text{number of endogenous facts on exogenous cycles among } C_1, ..., C_n \]

\[ E_p \leftarrow \text{number of endogenous facts on exogenous sub-paths among } P_1, ..., P_r \]

Remove exogenous sub-paths from \(P_1, ..., P_r\)

\[ n_1, ..., n_m \leftarrow \text{number of non-exogenous cycles of lengths } 1, ..., m, m = |\text{Vars}(q)| \]

\[ l_1, ..., l_m \leftarrow \text{number of paths in } P_1, ..., P_r \text{ of lengths } 1, ..., m, m = |\text{Vars}(q)| \]

let \(s_1 \leftarrow 0\)

for \(k_1, k_2\) s.t. \(k_1 + k_2 = k - E_c - E_p\) do

let \(s_1 \leftarrow 0\)

for \(y_1, ..., y_m\) s.t. \(\sum_{i=1}^m i \cdot y_i = k_1 - E_c\) do

\[ s_1 = s_1 + \Pi_{y_i=1}^{m} C_i \]

let \(s_2 \leftarrow 0\)

for \(z_{i,1}, ..., z_{m,m}\) s.t. \(\sum_{i=1}^m \sum_{j=1}^i j \cdot z_{i,j} = k_2 - E_p\) and \(\forall i, \sum_{j=1}^i z_{i,j} \leq l_i\) do

\[ s_2 = s_2 + \Pi_{z_{i,1}, ..., z_{i,i}}^{m} \]

\([s_1 \cdot s_2] \leftarrow res + s_1 \cdot s_2\)

return \((|D_n|)_k \leftarrow res\)

We summarize this result with the algorithm SingleSjNeg \((D, q, k)\) for computing \(\text{Sat}(D, q, k)\), where \(q\) is a CQ\(^-\) with one positive and one negative atom of the same relation that have the same set of variables. The correctness and complexity analysis of the algorithm follows directly from the proof of Theorem 3.3.

In conclusion, the problem of computing Shapley \((D, q, f)\) significantly changes when \(q\) has self-joins, and requires a different approach than the one used for self-join-free CQ\(^-\)s. We have seen that self-joins along with negation form a structure where facts “violate” each other, and consequently, the problem of computing the Shapley value becomes related to the problem of counting paths in a graph. Theorem 3.3 demonstrates a case where counting the number satisfying subsets is tractable, due to the special structure of the graph obtained from the query. However, even for the case of a query with one self-join where both atoms do not share the exact same set of variables, there is no guarantee that every fact has a single violating fact; hence, counting the paths and cycles in the graph becomes non-trivial, and it is not clear if this is possible in polynomial time. Finally, the complexity of computing the Shapley value for the general class of CQ\(^-\)s with self-join remains an open question (it is an open question for the case of CQs as well). The complexity is yet unknown even for simple queries, such as: \(q() : R(x, y, z), \neg R(y, x, x)\) and \(q() : R(x, y), R(y, z), \neg R(x, z)\).
3.3 Summation over CQs with Negation

We conclude this chapter with an observation regarding the Shapley value of facts w.r.t. queries of summation over (non-Boolean) CQs. We extend the definition of CQ to be an expression of the form:

\[ q(\bar{x}) : R_1(t_1), \ldots, R_n(t_n), \neg R_1'(t_1'), \ldots, \neg R_m'(t_m') \]

where \( \bar{x} \) is a tuple of variables from \( t_1, \ldots, t_n \). The answers to \( q \) on a database \( D \), denoted as \( q(D) \), are the tuples \( \bar{c} \) that are obtained by projecting to \( \bar{x} \) all homomorphisms from \( q \) to \( D \), and replacing each variable with the constant it is mapped to. Livshits et al. [LBKS20] have shown how their algorithm for computing the Shapley value for Boolean CQs can be extended to arbitrary summations over CQs, using the fundamental linearity property of the Shapley value. Our dichotomy here can be extended to such aggregate functions over CQs in a similar way.

We define the possible answers to \( q \) on \( D \) to be the set of all answers to \( q \) on any subset of \( D \). Observe that some of these answers may not be in \( q(D) \), since \( q \) is not necessarily monotone, and adding a fact to the database may violate a certain answer. Nevertheless, the set of all such possible answers is included in \( q_{\text{Pos}}(D) \), where \( q_{\text{Pos}} \) is the CQ obtained from \( q \) by removing all its negative atoms. Let \( \bar{a} \) be a possible answer to \( q \) on \( D \), and let \( q(\bar{x} \leftarrow \bar{a}) \) be the (Boolean) CQ obtained from \( q \) by replacing every variable \( x_j \) in \( \bar{x} \) with the constant \( c_j \) that \( x_j \) is mapped to in \( \bar{a} \). Due to the linearity of the Shapley value, we can compute the Shapley value of a fact \( f \) w.r.t. an aggregate function of sum over the answers to \( q \), using the Shapley values of \( q(\bar{x} \leftarrow \bar{a}) \) for each possible answer \( \bar{a} \). Hence, if \( q(\bar{x} \leftarrow \bar{a}) \) is hierarchical, then the Shapley value can be computed efficiently. Formally, let \( \varphi : \text{Const}^k \rightarrow \mathbb{R} \) be a function that maps every tuple of values in the length of \( k \) to a real number. If \( q \) is a CQ and the possible answers to \( q \) over a database \( D \) are \( \bar{a}_1, \ldots, \bar{a}_n \), then:

\[
\text{Shapley}(D, \text{sum} \varphi[q], f) = \sum_{i=1}^{n} \varphi(\bar{a}_i) \cdot \text{Shapley}(D, q(\bar{x} \leftarrow \bar{a}_i), f)
\]

where \( \text{sum} \varphi[q] \) denotes the query that sums the values of \( \varphi(\bar{a}) \) for every answer \( \bar{a} \in q(D) \).

Example 3.4. Theorem 3.1 implies that the Shapley value of a fact can be efficiently computed for the following aggregate query that sums up all the expenses \( e \) of growing products \( p \) in countries \( c \) instead of importing them:

\[
\text{Sum}_c\{q(c, p, e) : \text{GROWS}(c, p), \neg \text{IMPORT}(c, p), \text{EXPENSES}(c, p, e)\}
\]

Note that here, the Shapley value of a fact measures its contribution to the sum of expenses. Assume that every fact in the relation EXPENSES is exogenous, while the rest of the facts are endogenous. Computing the Shapley value of a fact \( f = \text{IMPORT}(c_f, p_f) \)
can be done as follows: for each possible answer to $q$, $\vec{a} = (c, p, e)$, we compute the Shapley value of $f$ w.r.t. the Boolean CQ $\neg q$ denoted as $q[(c, p, e) \leftarrow \vec{a}]$, which can be done efficiently. Observe that the Shapley value of $f$ w.r.t. some $q[(c, p, e) \leftarrow \vec{a}]$ is always negative or zero, since adding $f$ to the database can only change the result of the specific query $q[(c, p, e) \leftarrow \vec{a}]$ from true to false. That is the case only if $f$ is added to the database after both facts $\text{Grows}(c_f, p_f)$ and $\text{Expenses}(c_f, p_f, e_f)$. Finally, the Shapley value of $f$ w.r.t. the sum of expenses is a linear combination of the Shapley values of $f$ w.r.t. the Boolean CQ $\neg q$'s of the form $q[(c, p, e) \leftarrow \vec{a}]$ for every possible answers $\vec{a}$ to $q$, where the coefficients are the expenses $e$ in each such answer. In this case, $\text{Shapley}(D, \text{Sum}_e[q], f) = 0$ if $\text{Grows}(c_f, p_f)$ or $\text{Expenses}(c_f, p_f, e_f)$ do not exist in $D$, and $\text{Shapley}(D, \text{Sum}_e[q], f) = e_f \cdot 1/2$ otherwise. That is since $f$ turns the result of $q[(c, p, e) \leftarrow (c_f, p_f, e_f)]$ from true to false only in permutations where it comes after the fact $\text{Grows}(c_f, p_f)$, which are exactly half of all endogenous facts permutations.
Chapter 4

Accounting for Exogenous Relations

In the previous section, we showed that computing the Shapley value is \( \text{FP}^{\#P} \)-complete for every non-hierarchical self-join-free CQ\(^-\). Yet, this hardness result does not take into account the reasonable assumption that some of the relations in the database contain only exogenous facts. For example, Meliou et al. [MGMS10] discussed the case where all the relations in the database are exogenous, except for one (e.g., “Director” or “Movie”); this one relation may be a suspect of containing erroneous data, or the one that holds the single type of entities for which we wish to quantify the contribution. In this chapter, we show that accounting for such relations significantly changes the complexity picture and, in particular, it makes some of the intractable queries according to Theorem 3.1 tractable. In fact, we generalize Theorem 3.1 to account for exogenous relations and therefore establish the precise class of CQ’s that become tractable. Throughout this section, we underline the relations containing only exogenous facts and their associated query atoms.

Example 4.1. Livshits et al. [LBKS20] demonstrated their work on a database from the domain of academic publications. They reasoned about the contribution of researchers to the total number of citations and assumed that the information about the publications is exogenous. In particular, they considered the query:

\[
q() : \text{Author}(x, y), \quad \text{Pub}(x, z), \quad \text{Citations}(z, w)
\]

Since \( q \) is not hierarchical, their result classifies it as intractable. However, in this section, we show that there is a polynomial-time algorithm for computing the Shapley value for \( q \), under the assumption that \text{Pub} and \text{Citations} contain only exogenous facts. Furthermore, we show that even if we had that prior knowledge about the relation \text{Citations} alone, we would still able to compute the Shapley value efficiently. This is due to the fact that we can reduce the problem of computing Shapley\((D, q, f)\) to that of computing Shapley\((D, q', f)\) for the hierarchical query \(q'() : \text{Author}(x, y), \text{Pub}(x, z),\)
by removing from the relation \( \text{Pub} \) in \( D \) every fact \( \text{Pub}(a, b) \) such that there is no fact \( \text{Citations}(b, c) \) in \( D \) and then removing the relation \( \text{Citations} \) from the query.

Next, consider the database of our running example (Figure 2.1). We have assumed that the information about the students and courses in the faculty is exogenous, and our goal was to understand how much the fact that a student takes or teaches a course affects the result of different queries. For example, consider again the query \( q_2 \) from Example 2.2.

\[
q_2() : \overline{\text{Stud}(x)}, \neg\text{TA}(x), \text{REG}(x, y), \neg\text{Course}(y, \text{CS})
\]

Theorem 3.1 classifies this query as intractable for computing the Shapley value, as it is not hierarchical. Yet, again, under the assumption that every fact in \( \text{Stud} \) and \( \text{Course} \) is exogenous, the Shapley value can be computed in polynomial time. Note that when negation is added to the picture, we cannot simply remove exogenous atoms, as removing an exogenous atom may turn a query with safe negation into a query with negation that is not safe (e.g., \( q'(x) = R(x), \neg S(x, y), T(y) \)).

\[
\square
\]

## 4.1 Generalized Dichotomy

We start by formally defining the problem that we study in this section. We define an exogenous relation \( R \) to be a relation that consists only of exogenous facts. We fix a schema \( S \), a set \( X \) of exogenous relations in \( S \), and a self-join-free CQ\( ^\neg \) \( q \). We denote by \( S_X \) a schema with the set \( X \) of exogenous relations. Note that we do not assume anything about the facts in the relations outside \( X \) and they may contain both endogenous and exogenous facts. Then, our goal is to compute \( \text{Shapley}(D, q, f) \), given a database \( D \) over \( S_X \) and a fact \( f \in D_n \).

Clearly, the assumption that some of the relations of \( S \) are exogenous does not change the fact that we can compute the Shapley value in polynomial time for any hierarchical CQ\( ^\neg \). To understand the impact of this assumption on the complexity of non-hierarchical CQ\( ^\neg \)'s, consider the query \( q_{R-ST} \) defined in Chapter 3. If we assume that only \( S \) is exogenous, then the query remains hard, as \( S \) already contains only exogenous facts in the proof of hardness for \( q_{R-ST} \) (Lemma 3.1.3). We can generalize this example and show that having a non-hierarchical triplet \((\alpha_x, \alpha_x, y, \alpha_y)\) where \( R_{\alpha_x} \notin X \) and \( R_{\alpha_y} \notin X \) is a sufficient condition for FP\#P-hardness, as the hardness proofs of the previous chapter can be easily generalized to this case. Is having such a triplet a necessary condition for hardness? Next, we answer this question negatively. Consider the following queries:

\[
q() : \neg R(x, w), S(z, x), \neg P(z, w), T(y, w)
\]

\[
q'(x) : \neg R(x, w), S(z, x), \neg P(z, y), T(y, w)
\]

32
In both queries, the exogenous relations are $S$ and $P$, and they differ only in one variable that occurs in the atom of $P$. While the two queries are very similar and are both classified as intractable by Theorem 3.1, we will show that in the model considered in this section, $\text{Shapley}(D, q, f)$ can be computed in polynomial time for every endogenous fact $f$, while computing $\text{Shapley}(D, q', f)$ is $\text{FP}^{\#P}$-complete. This holds true as while in both cases the non-exogenous atoms are connected via the exogenous atoms, they are connected in different ways. While the connection in $q$ between the variable $x$ in $\neg R(x, w)$ and the variable $y$ in $T(y, w)$ goes through the variable $w$, in $q'$ the connection between $x$ and $y$ is possible through the variable $z$ as well, and we need to be able to distinguish between these two cases. In the terminology we set next, we say that $x$ and $y$ are connected via a non-hierarchical path in $q'$ (but not in $q$).

**Non-hierarchical path.** Let $S_X$ be a schema. The Gaifman graph $G(q)$ of a CQ $q$ is the graph that contains a vertex for every variable in $q$ and an edge between two vertices if the corresponding variables occur together in an atom of $q$. We say that a CQ $q$ has a non-hierarchical path if there are two atoms $\alpha_x, \alpha_y$ and two variables $x, y$ in $q$ such that: (1) $R_{\alpha_x} \not\in X$ and $R_{\alpha_y} \not\in X$, (2) the variable $x$ occurs in $\alpha_x$ but not in $\alpha_y$, while the variable $y$ occurs in $\alpha_y$ and not in $\alpha_x$, and (3) the graph obtained from $G(q)$ by removing the vertex of every variable in $\alpha_x$ and $\alpha_y$ (except for $x$ and $y$) contains a path between $x$ and $y$. In this case, we say that the non-hierarchical path of $q$ is induced by the atoms $\alpha_x$ and $\alpha_y$.

**Example 4.2.** Consider the query:

$q() : = \neg R(x), Q(x, v), S(x, z), U(z, w), \neg P(w, y), T(y, v)$

The Gaifman graph $G(q)$ is illustrated in Figure 4.1a. We claim that $q$ has a non-hierarchical path induced by the atoms $\neg R(x)$ and $T(y, v)$. Note that there is a path $x \rightarrow v \rightarrow y$ in $G(q)$ between $x$ and $y$; however, this is not enough to determine that $q$ has a non-hierarchical path, as we need to find a path that does not pass through the variables of $\neg R(x)$ and $T(y, v)$. And indeed, if we remove from $G(q)$ the variable $v$ occurring in the atom $T(y, v)$ and every edge connected to it (i.e., every dotted line in the graph of Figure 4.1a), there is a path $x \rightarrow z \rightarrow w \rightarrow y$ between the variables $x$ and $y$ in the resulting graph, and we conclude that $q$ has a non-hierarchical path.
Next, consider the query:

\[
q'() : = U(t, r), \neg T(y), Q(y, w), \\
\neg V(t), R(x, y), \neg S(x, z), O(z), P(u, y, w)
\]

The reader can easily verify, using the graph of Figure 4.1b, that \(q'\) does not have a non-hierarchical path. This is because the variables of \(U(t, r)\) and the variables of \(\neg T(y)\) or \(Q(y, w)\) are not connected in \(G(q')\). Moreover, every variable in \(\neg T(y)\) also appears in \(Q(y, w')\); hence, no non-hierarchical path can be induced by these two atoms.

We prove the following generalization of Theorem 3.1 that account for exogenous relations.

**Theorem 4.3.** Let \(S_X\) be a schema and let \(q\) be a CQ\(^-\) without self-joins. If \(q\) has a non-hierarchical path, then computing Shapley\((D, q, f)\) is FP\(^\#\)-complete. Otherwise, Shapley\((D, q, f)\) can be computed in polynomial time, given \(D\) and \(f\).

The proof of the hardness side of Theorem 4.3 is very similar to the proof of hardness for Theorem 3.1. Let \(S_X\) be a schema and let \(q\) be a self-join-free CQ\(^-\) that contains a non-hierarchical path. Similarly to the proof of Theorem 3.2, we construct a reduction from the problem of computing Shapley\((D, q', f)\) where \(q'\) is one of \(q_{RS}, q_{RS-T}\) or \(q_{RS\rightarrow T}\) to that of computing Shapley\((D, q, f)\). We again assume that in the input to the first problem all the facts of \(S\) are exogenous, and for every fact \(S(a, 1)\) in \(D\), both facts \(R(a)\) and \(T(1)\) are in \(D\). We also assume that the set of values used in the facts of \(R^D\) and the set of values used in the facts of \(T^D\) are disjoint.

Since \(q\) has a non-hierarchical path, there exist two atoms \(\alpha_x\) and \(\alpha_y\) in \(q\) and two variables \(x, y\), such that \(R_{\alpha_x} \not\in X\) and \(R_{\alpha_y} \not\in X\), the variable \(x\) occurs in \(\alpha_x\) but not in \(\alpha_y\) and the variable \(y\) occurs in \(\alpha_y\) but not in \(\alpha_x\). Moreover, there exists a path \(x - v_1 - \cdots - v_n - y\) in the graph obtained from the Gaifman graph \(G(q)\) of \(q\) by removing every variable in \((\text{Vars}(\alpha_x) \cup \text{Vars}(\alpha_y)) \setminus \{x, y\}\). The idea is the following. We use the atoms \(\alpha_x\) and \(\alpha_y\) to represent the atoms \((-)R(x)\) and \((-)T(y)\) in \(q'\), respectively, and we use the non-hierarchical path to represent the connections between them (i.e., the atom \(S(x, y)\)). If \(\alpha_x\) and \(\alpha_y\) are both positive, the reduction is from the problem of computing Shapley\((D, q_{RS}, f)\), if both atoms are negative, the reduction is from computing Shapley\((D, q_{RS\rightarrow T}, f)\), and if one atom is positive while the other is negative, the reduction is from computing Shapley\((D, q_{RS\rightarrow T}, f)\).

Formally, given an input database \(D\) to the first problem, we build a database \(D'\) in the following way. For every fact \(f = R(a)\) we assign the value \(a\) to the variable \(x\) in \(\alpha_x\), the value \(\odot\) to the rest of the variables and every constant to itself, and we add the corresponding fact \(f'\) to the relation \(R_{\alpha_x}\) in \(D'\). The fact \(f'\) will be endogenous if and only if \(f\) is endogenous. Similarly, for every fact \(f = T(1)\) we assign the value \(1\) to the variable \(y\) in \(\alpha_y\), the value \(\odot\) to the rest of the variables and every constant to itself, and we add the corresponding fact \(f'\) to the relation \(R_{\alpha_y}\) in \(D'\). Again, the fact
Let $f'$ will be endogenous if and only if $f$ is endogenous. Next, for every fact $S(a, 1)$ in $D$ and an atom $\alpha$ in $q$ that is not one of $\alpha_x$ or $\alpha_y$, we assign the value $a$ to the variable $x$, the value 1 to the variable $y$, the value $\langle a, 1 \rangle$ to the variables $v_1, \ldots, v_n$ along the non-hierarchical path, the value $\odot$ to the rest of the variables and every constant to itself, and we add the corresponding exogenous fact to the relation $R_\alpha$ in $D'$, if we have not added this fact to $D'$ already. Note that $|D_n| = |D'_n|$.

Now, given the database $D'$ we construct a database $D''$ which will be the input to our problem in the following way. We first copy all the endogenous facts from $D'$ to $D''$. Then, for every relation $R$ in $D'$ corresponding to a positive atom of $q$, we copy every exogenous fact from $R^{D'}$ to $R^{D''}$. For every relation $R$ in $D'$ corresponding to a negative atom of $q$, we add to $R^{D''}$ every exogenous fact over the domain of $D'$ if and only if it does not occur in $R^{D'}$ (i.e., $R^{D''} = \overline{R^{D'}}$). Note that since we did not change the endogenous facts, we have that $|D_n| = |D'_n| = |D''_n|$.

We will now prove that for every endogenous fact $f_1$ in $D$ and its corresponding fact $f_2$ in $D''$ it holds that $\text{Shapley}(D, q', f_1) = \text{Shapley}(D'', q, f_2)$ (recall that $q'$ is one of $q_{\text{RST}}, q_{\text{RS,T}}, q_{\text{RS,T}}$). We start by proving the following.

Lemma 4.1.1. Let $E \subseteq D_n$ and let $E''$ be the set of corresponding facts in $D''_n$. If $(D_x \cup E) \models q'$ then $(D''_x \cup E'') \models q$.

Proof. Since $(D_x \cup E) \models q'$ there is a mapping $h$ from the variables of $q'$ to the domain of $D$ where $h(x) = a$ for some value $a$ from the domain of $R^D$ and $h(y) = 1$ for some value 1 from the domain of $T^D$ such that $h$ maps every positive atom and none of the negative atoms of $q'$ to a fact of $D_x \cup E$. We claim that the mapping $h'$ such that $h'(x) = a$, $h'(y) = 1$, $h'(z) = \langle a, 1 \rangle$ for every variable $z$ along the non-hierarchical path, and $h'(w) = \odot$ for the rest of the variables, maps every positive atom and none of the negative atoms of $q$ to facts of $D''_x \cup E''$; hence $(D''_x \cup E'') \models q$.

When considering the fact $S(a, 1)$ in the construction of $D'$, we have created a mapping $h'$ from the variables of $q$ such that $h'(x) = a$, $h'(y) = 1$, $h'(z) = \langle a, 1 \rangle$ for every variable along the non-hierarchical path, and $h'(w) = \odot$ for the rest of the variables, and we have added all the resulting facts (associated with atoms that are not one of $\alpha_x$ or $\alpha_y$) to $D'$ (as exogenous facts). When constructing $D''$ we have removed every such fact if it was generated from a negative atom of $q$. Hence, $h'$ is a mapping from the the variables of $q \setminus \{\alpha_x, \alpha_y\}$ (which is the query obtained from $q$ by removing the atoms $\alpha_x$ and $\alpha_y$) to the domain of $D''$ such that every positive atom and none of the negative atoms of $q$ appears as a fact in $D''_x$. Moreover, it holds that $R(a) \in D_n$ if and only if $h'(\alpha_x) \in D''_n$ and similarly $T(1) \in D_n$ if and only if $h'(\alpha_y) \in D''_n$. Then, $(D''_x \cup E'')$ indeed satisfies $q$ and that concludes our proof.

Next, we prove the following.

Lemma 4.1.2. Let $E \subseteq D_n$ and let $E''$ be the set of corresponding facts in $D''_n$. If $(D_x \cup E) \not\models q'$ then $(D''_x \cup E'') \not\models q$.  

35
Proof. Assume, by way of contradiction, that \( D'_n \cup E'' \) satisfies \( q \). Hence, there is a mapping \( h \) from the variables of \( q \) to the domain of \( D'' \) such that every positive atom and none of the negative atoms of \( q \) is mapped into a fact in \( D'_n \cup E'' \). We now look at the non-hierarchical path \( x - v_1 - \cdots - v_n - y \) in the Gaifman graph of \( q \). From the construction of \( D' \), every fact \( f \in D' \) in a relation corresponding to an atom \( \alpha \) that uses both \( x \) and \( v_1 \) is obtained from \( \alpha \) by mapping the variable \( x \) to some value \( c_1 \) and the variable \( v_1 \) to some value \( \langle c_1, c_2 \rangle \) such that \( S(c_1, c_2) \) is in \( D \). If \( \alpha \) is positive, then \( D'' \) also contains only such facts, and if \( \alpha \) is negative, then \( D'' \) does not contain only such facts. Using an atom \( \alpha' \) containing the variables \( y \) and \( v_n \), we can show, in a similar way, that \( h(v_n) = \langle d_1, d_2 \rangle \) for some values \( d_1, d_2 \) such that \( S(d_1, d_2) \) is in \( D \). Finally, every two consecutive variables \( v_i, v_{i+1} \) in the non-hierarchical path occur together in at least one atom \( \alpha \) of \( q \), and from the construction of \( D'' \), it holds that if \( \alpha \) is positive, then \( R_\alpha \) contains only facts where both \( v_i \) and \( v_{i+1} \) are mapped to the same value, and if \( \alpha \) is negative, then these are the only facts that are not in \( R_\alpha \); hence, we have that \( h(v_i) = h(v_{i+1}) \) and we conclude that \( c_1 = d_1 \) and \( c_2 = d_2 \), and the mapping \( h \) assigns some value \( \langle a, 1 \rangle \) to every variable along the non-hierarchical path, such that \( S(a, 1) \) is in \( D \).

When constructing the database \( D' \), we have only assigned the value \( \langle a, 1 \rangle \) to variables if there exists an exogenous fact \( S(a, 1) \) in \( D \). Hence, we have established that such a fact exists in \( D \). Moreover, it holds that \( R(a) \in E \) if and only if \( h(\alpha_x) \in E'' \) and similarly \( T(1) \in E \) if and only if \( h(\alpha_y) \in E'' \). In all cases, the restriction of \( h \) to the variables \( x \) and \( y \) maps every positive atom and none of the negative atoms of \( q' \) to \( D_x \cup E \); thus, \( (D_x \cup E) \models q' \), which is a contradiction to our assumption. \( \blacksquare \)

The remainder of the proof is rather straightforward based on these two lemmas. The total number of permutations of the facts in \( D_n \) and \( D''_n \) is equal, and the lemmas prove that the number of permutations where \( f \) changes the result of \( q' \) in \( D \) is equal to the number of permutations where it changes the result of \( q \) in \( D' \); hence, we conclude that \( \text{Shapley}(D, q', f) = \text{Shapley}(D', q, f) \).

In the following section, we discuss the proof of the positive side of Theorem 4.3.

### 4.2 Algorithm for the Tractable Cases

We will show that computing the Shapley value for a self-join-free CQ\(^-\) that does not have a non-hierarchical path can be reduced to computing the Shapley value for a hierarchical query \( q' \) without self-joins. Our reduction consists of three steps that will form the basis to our algorithm. Since the Shapley value can be computed in polynomial time for hierarchical CQ\(^-\)’s (Theorem 3.1), and the same algorithm works for the model that we consider in this chapter, we will conclude that the Shapley value can be computed in polynomial time for such queries.

For the remainder of this section, we fix a schema \( S_X \) and a self-join-free CQ\(^-\) \( q \)
that does not have a non-hierarchical path. We first introduce some definitions and notations that we will use throughout the proof. We denote by \(\text{Atoms}(q)\) and \(\text{Vars}(q)\) the sets of atoms and variables of \(q\), respectively. We say that an atom \(\alpha\) of \(q\) is an exogenous atom if \(R_\alpha \in X\). We say that a variable \(x\) of \(q\) is an exogenous variable if it occurs only in exogenous atoms of \(q\). We denote the sets of all exogenous atoms and variables of \(q\) by \(\text{Atoms}_x(q)\) and \(\text{Vars}_x(q)\), respectively. We denote by \(\text{Atoms}_{x,\alpha}(q)\) the set \((\text{Atoms}(q) \setminus \text{Atoms}_x(q))\) of non-exogenous atoms\(^1\) in \(q\) and by \(\text{Vars}_{x,\alpha}(q)\) the set \((\text{Vars}(q) \setminus \text{Vars}_x(q))\) of non-exogenous variables.

Next, we define the exogenous atom graph \(g_x(q)\) of \(q\) to be the graph that contains a vertex for every exogenous atom in \(q\) and an edge between two vertices if the corresponding two atoms share an exogenous variable. The following lemma draws a connection between the properties of \(g_x(q)\) and the existence of a non-hierarchical path in \(G(q)\). In particular, we prove that if a query \(q\) does not have a non-hierarchical path, then for every connected component \(C\) of \(g_x(q)\) there is a non-exogenous atom \(\alpha\) of \(q\) such that \(\text{Vars}_{x,\alpha}(C) \subseteq \text{Vars}(\alpha)\). This property is of high significance, as our reduction strongly relies on it.

**Lemma 4.2.1.** For every connected component \(C\) of \(g_x(q)\) there is an atom \(\alpha \in \text{Atoms}_{x,\alpha}(q)\) such that \(\text{Vars}_{x,\alpha}(C) \subseteq \text{Vars}(\alpha)\).

**Proof.** Let \(C\) be a connected component of \(g_x(q)\). Assume, by way of contradiction, that there is no \(\alpha \in \text{Atoms}_{x,\alpha}(q)\) such that \(\text{Vars}_{x,\alpha}(C) \subseteq \text{Vars}(\alpha)\), and let \(\alpha \in \text{Atoms}_{x,\alpha}(q)\) be an atom of \(q\) such that \(\text{Vars}_{x,\alpha}(C) \cap \text{Vars}(\alpha)\) is maximal among all atoms in \(\text{Atoms}_{x,\alpha}(q)\). Since \(\text{Vars}_{x,\alpha}(C) \neq \emptyset\) and every non-exogenous variable occurs in a non-exogenous atom, there exists \(x \in (\text{Vars}_{x,\alpha}(C) \cap \text{Vars}(\alpha))\). Moreover, since \(\text{Vars}_{x,\alpha}(C) \not\subseteq \text{Vars}(\alpha)\), there exists \(y \in \text{Vars}_{x,\alpha}(C)\) that does not occur in \(\alpha\). Since \(y\) is not an exogenous variable, there is another \(\alpha' \in \text{Atoms}_{x,\alpha}(q)\) such that \(y \in \text{Vars}(\alpha')\). It cannot be the case that \(x \in \text{Vars}(\alpha')\) (as otherwise, we get a contradiction to the maximality of \(\text{Vars}_{x,\alpha}(C) \cap \text{Vars}(\alpha)\)); hence, we conclude that \(x \in (\text{Vars}(\alpha) \setminus \text{Vars}(\alpha'))\), \(y \in (\text{Vars}(\alpha') \setminus \text{Vars}(\alpha))\), and \(x, y \in \text{Vars}(C)\).

We claim that \(\alpha\) and \(\alpha'\) induce a non-hierarchical path in \(G(q)\). Since \(x, y \in \text{Vars}(C)\), there exist two (not necessarily distinct) atoms \(\beta_1, \beta_2 \in C\) such that \(x \in \beta_1\) and \(y \in \beta_2\). Since \(\beta_1\) and \(\beta_2\) belong to the same connected component, there exists a path in \(g_x(q)\) between \(\beta_1\) and \(\beta_2\), such that the edges along the path correspond to exogenous variables of \(q\). Therefore, there is a path \(x - v_1 - \cdots - v_n - y\) in \(G(q)\), such that each \(v_i\) is an exogenous variable (hence, \(v_i \notin \text{Vars}(\alpha)\) and \(v_i \notin \text{Vars}(\alpha')\)). This path is a non-hierarchical path by definition. \(\blacksquare\)

**Example 4.4.** Consider the query \(q'\) of Example 4.2. We have already established that \(q'\) does not have a non-hierarchical path. Figure 4.2a illustrates both the exogenous

---

\(^1\)Note that we do not refer to non-exogenous atoms as “endogenous” atoms, since the corresponding relations may contain both endogenous and exogenous facts.
atom graph of \( q' \) and the result of Lemma 4.2.1. The atoms in the white rectangles are the exogenous atoms of \( q' \), and the atoms in the gray circles are the non-exogenous atoms. Every gray rectangle containing a set of exogenous atoms represents a connected component in \( g_\alpha(q') \). For example, the atoms \( R(x, y) \) and \( \neg S(x, z) \) share the exogenous variable \( x \) and the atoms \( \neg S(x, z) \) and \( O(z) \) share the exogenous variable \( z \). Hence, all three atoms form a connected component \( C \) in the graph. The only non-exogenous variable in \( C \) is \( y \) and, indeed, there is a non-exogenous atom \( \neg T(y) \) that uses \( y \). In fact, there are two such atoms, and in the next step we can arbitrary select one of them. The exogenous atom \( P(u, y, w) \) is a connected component on its own, as its only exogenous variable \( u \) does not occur in any other atom. And, again, there is a non-exogenous atom \( Q(y, w) \) that uses both non-exogenous variables \( y \) and \( u \) of \( P(u, y, w) \). ■

Next, we discuss the first step of our reduction. We prove that we can replace every connected component \( C \) of \( g_\alpha(q) \) with a single exogenous atom in \( q \), obtained by “joining” all the atoms of \( C \) (and the corresponding relations of \( D \)), without affecting the Shapley value. Since some of the atoms in a connected component \( C \) may be negated, and it is not clear how to combine positive and negative atoms into a single atom, we first replace them with positive atoms and compute the complement of the corresponding relations. Formally, given a negated atom \( \alpha \), we denote by \( \overline{\alpha} \) the atom obtained from \( \alpha \) by removing the negation. Then, we denote by \( R^D_\alpha \) the relation obtained from \( R^D_\alpha \) by adding every fact over the domain of \( D \) if and only if it does not appear in \( R^D_\alpha \). That is, if the arity of \( R_\alpha \) is \( k \), then we add to \( R^D_\alpha \) a fact \( R_\alpha(c_1, \ldots, c_k) \), where each \( c_i \) is a constant from the domain of \( D \), if and only if \( R_\alpha(c_1, \ldots, c_k) \notin R^D_\alpha \). Hence, we obtain a query \( q' \) by replacing every negated exogenous atom \( \alpha \) of \( q \) with the atom \( \overline{\alpha} \), and we construct a database \( D' \) by replacing every exogenous relation \( R^D_\alpha \) corresponding to a negated atom of \( q \) with the complement relation \( R^D_\alpha \). The same idea has been used in the proof of hardness for the query \( q_{R-ST} \) in the previous chapter...
3.1.3

Let \( R \) be obtained by an inner join between the relations \( h \). A homomorphism \( h \) maps \( \alpha \) to a fact of \( D \) if and only if it maps \( \beta \) to a fact of \( D' \). Again, since the rest of the atoms are unchanged in \( q' \), we conclude that \( h \) maps every positive atom and none of the negative atoms of \( q' \) to a fact of \( D' \).

The proof of the second direction is very similar. If \( (D_x \cup E) \models q' \), then there is a homomorphism \( h \) from the variables of \( q' \) to the constants of \( D \). Moreover, for every \( E \subseteq D_n \), it holds that \( (D_x \cup E) \models q \) if and only if \( (D'_x \cup E) \models q' \). This is rather straightforward from the construction of \( D' \). If \( (D_x \cup E) \models q \), then there is a homomorphism \( h \) from the variables of \( q \) to the constants of \( D \) that does not map \( \alpha \) to any fact of \( R^D_\alpha \) (since \( \alpha \) is a negated atom); hence, it maps \( \overline{\alpha} \) to a fact of \( \overline{R^D_\alpha} \). Every other atom \( \beta \) of \( q \) also occurs in \( q' \) and we have not changed the relations corresponding to other atoms; thus, \( h \) maps \( \beta \) to a fact of \( (D_x \cup E) \) if and only if it maps \( \beta \) to a fact of \( (D'_x \cup E) \), and we conclude that \( h \) maps every positive atom and none of the negative atoms of \( q' \) to a fact of \( (D'_x \cup E) \).

Proof. Note that the difference between \( D \) and \( D' \) is restricted to the exogenous facts; thus, we have that \( D_n = D'_n \). Moreover, for every \( E \subseteq D_n \), it holds that \( (D_x \cup E) \models q \) if and only if \( (D'_x \cup E) \models q' \). This is rather straightforward from the construction of \( D' \). If \( (D_x \cup E) \models q \), then there is a homomorphism \( h \) from the variables of \( q \) to the constants of \( D \) that does not map \( \alpha \) to any fact of \( R^D_\alpha \) (since \( \alpha \) is a negated atom); hence, it maps \( \overline{\alpha} \) to a fact of \( \overline{R^D_\alpha} \). Every other atom \( \beta \) of \( q \) also occurs in \( q' \) and we have not changed the relations corresponding to other atoms; thus, \( h \) maps \( \beta \) to a fact of \( (D_x \cup E) \) if and only if it maps \( \beta \) to a fact of \( (D'_x \cup E) \), and we conclude that \( h \) maps every positive atom and none of the negative atoms of \( q' \) to a fact of \( (D'_x \cup E) \).

The proof of the second direction is very similar. If \( (D'_x \cup E) \models q' \), then there is a homomorphism \( h \) from the variables of \( q' \) to the constants of \( D' \) that maps \( \overline{\alpha} \) to a fact of \( \overline{R^D_\alpha} \). Again, since the rest of the atoms are unchanged in \( q' \), we conclude that \( h \) maps every positive atom and none of the negative atoms of \( q \) to a fact of \( (D_x \cup E) \). We conclude that the total number of permutations is equal in both databases, and the number of permutations where \( f \) changes the query result is equal as well; hence, \( \text{Shapley}(D, q, f) = \text{Shapley}(D', q', f) \). \( \blacksquare \)

From now on, we assume that every exogenous atom of \( q \) is positive. We use that assumption to prove the following.

Lemma 4.2.3. Computing \( \text{Shapley}(D, q, f) \), given \( D \) and \( f \), can be efficiently reduced to computing \( \text{Shapley}(D', q', f) \) for a CQ without self-joins such that: (1) every exogenous variable of \( q' \) occurs in a single atom, and (2) \( q' \) does not have any non-hierarchical path.

Proof. Let \( C \) be a connected component of \( g_\alpha(q) \), and let the set \( \{\alpha_1, ..., \alpha_k\} \) be the set of (exogenous) atoms in \( C \). Let \( q' \) be the query obtained from \( q \) by replacing all the atoms of \( C \) with a single atom \( \alpha_C \), such that \( \text{Vars}(\alpha_C) = \cup_{i \in \{1, ..., k\}} \text{Vars}(\alpha_i) \). Observe that since \( C \) is a connected component of \( g_\alpha(x) \), none of the exogenous variables occurring in \( \alpha_C \) also occurs in another atom of \( q' \). Let \( D' \) be the database obtained from \( D \) by replacing the exogenous relations \( R_{\alpha_1}, ..., R_{\alpha_k} \) with a single exogenous relation \( R_{\alpha_C} \) consisting of the set of answers to the query \( q_C(\overline{x}) : \alpha_1, ..., \alpha_k \) on the database \( D \) (where every variable of \( \alpha_1, ..., \alpha_k \) occurs in \( \overline{x} \)). That is, the facts in the relation \( R_{\alpha_C}^{D'} \) are obtained by an inner join between the relations \( R_{\alpha_1}^{D}, ..., R_{\alpha_k}^{D} \), where the relations are joined according to the variables of the corresponding atoms.

Since we have only changed the exogenous relations in \( D \) to obtain \( D' \), we have that \( D'_n = D_n \). We now prove that for every \( E \subseteq D_n \) it holds that \( (D_x \cup E) \models q \) if
and only if \((D'_x \cup E) \models q'\), which implies that Shapley\((D, q, f) = \text{Shapley}(D', q', f)\) for every endogenous fact \(f\). Let \(E \subseteq D_n\). If \((D_x \cup E) \models q\), then there is a homomorphism \(h\) from \(q\) to \(D_x \cup E\). Note that the only negative atoms of \(q\) are atoms corresponding to non-exogenous relations; hence, if \(h\) does not map any negative atom of \(q\) to a fact of \(D_x \cup E\), it also does not map any negative atom of \(q'\) to a fact of \(D'_x \cup E\). As for the positive atoms, the homomorphism \(h\) maps every positive atom of \(q\), and, in particular, the atoms \(\alpha_1, \ldots, \alpha_k\) of the connected component \(C\), to facts of \(D_x \cup E\). Assume that \(h(v) = c_v\) for every variable in \(\alpha_1, \ldots, \alpha_k\). By the definition of \(q_C\), the tuple \((c_{v_1}, \ldots, c_{v_n})\) (where \(v_1, \ldots, v_n\) are the variables occurring in \(\alpha_1, \ldots, \alpha_k\)) is an answer to \(q_C\). Hence, the atom \(\alpha_C\) in \(q'\) is mapped to a fact of \(D'_x\). Every positive atom of \(q\) that is not one of \(\alpha_1, \ldots, \alpha_k\) also occurs in \(q'\) and it is mapped to a fact of \(D_x \cup E\) that also appears in \(D'_x \cup E\); hence, we conclude that \(h\) is a homomorphism from \(q'\) to \(D' \cup E\).

Similarly, if we assume that \((D'_x \cup E) \models q'\), then there is a homomorphism \(h\) from \(q'\) to \(D'_x \cup E\). Every atom \(\alpha \in \text{Atoms}(q) \setminus \{\alpha_1, \ldots, \alpha_k\}\) occurs in both \(q\) and \(q'\), and the relation \(R_{\alpha}\) is the same in \(D\) and \(D'\); thus, every such \(\alpha\) is mapped to a fact of \(D_x \cup E\) if and only if it is mapped to a fact of \(D'_x \cup E\). Since every fact in \(R_{\alpha_C}'\) is mapped by \(h\) to a fact \(R_{\alpha_C}(c_{v_1}, \ldots, c_{v_n})\) in \(D'_x\), then every atom \(\alpha_i\) in \(q\) is mapped by \(h\) to a fact \(R_{\alpha_i}(c_{v_1}, \ldots, c_{v_n})\) in \(D\) where \(\{v_1, \ldots, v_n\}\) is the set of variables occurring in \(\alpha_i\), as if such a fact did not exist, we would never obtain the tuple \((c_{v_1}, \ldots, c_{v_n})\) as an answer to \(q_C\) on \(D\). Hence, we have that \((D_x \cup E) \models q\), as evidenced by \(h\).

The above argument holds for every connected component of \(g_x(q)\); hence, we can replace every connected component with a single atom in \(q\) and change the database \(D\) accordingly. This will result in a query \(q'\) where every exogenous variable occurs exactly once and that concludes our proof. We finish this proof by showing that \(q'\) does not have a non-hierarchical path.

Let us assume, by way of contradiction, that the query \(q'\) has a non-hierarchical path induced by the atoms \(\alpha_x\) and \(\alpha_y\). Hence, in the Gaifman graph of \(q'\), there is a path \(x - v_1 - \cdots - v_n - y\) that does not pass through the variables of \(\alpha_x\) and \(\alpha_y\). We claim that there is also a non-hierarchical path induced by \(\alpha_x\) and \(\alpha_y\) in \(q\), in contradiction to the fact that \(q\) does not have a non-hierarchical path. Let \(v_i, v_{i+1}\) be two consecutive variables in the path. If \(v_i\) and \(v_{i+1}\) occur together in a non-exogenous atom of \(q'\), then they occur together in the same non-exogenous atom of \(q\), and \(v_i, v_{i+1}\) are also connected in the Gaifman graph of \(q\). Otherwise, \(v_i, v_{i+1}\) occur together in an exogenous atom of \(q'\). This exogenous atom represents a connected component \(\{\alpha_1, \ldots, \alpha_k\}\) in \(g_x(q)\). Let \(\alpha_j\) be the atom where the variable \(v_i\) occurs and let \(\alpha_r\) be the variable where the variable \(v_{i+1}\) occurs. By the definition of \(g_x(q)\), there is a path \(u_1 - \cdots - u_m\) between \(\alpha_j\) and \(\alpha_r\) such that \(u_1 \in \alpha_j, u_m \in \alpha_r\) and every \(u_i\) is an exogenous variable (hence, it does not occur in \(\alpha_x\) or \(\alpha_y\)). We conclude that the Gaifman graph of \(q\) contains the path \(v_i - u_1 - \cdots - u_m - v_{i+1}\) that does not pass through the variables of \(\alpha_x\) or \(\alpha_y\).
Therefore, there is a non-hierarchical path between \( x \) and \( y \) in \( q \), and that concludes our proof.

One may suggest that it is possible to avoid replacing the negated exogenous atoms of \( q \) with positive atoms before combining exogenous atoms, by simply constructing the relation \( R_{α_{C}} \) in \( D \) using the query \( q_C(\vec{x}) \colon α_1, \ldots, α_k \), where \( α_1, \ldots, α_k \) are the original (possibly negated) atoms in the connected component \( C \), and \( \vec{x} \) contains every variable of \( C \). The problem with this approach is that the resulting \( q_C \) may have non-safe negation, as a non-exogenous variable of \( C \) may appear only in negated atoms of \( C \) (and in a positive atom outside \( C \)). Thus, it is essential to replace the relations corresponding to negated exogenous atoms of \( q \) with their complement relations, before combining the atoms of \( C \) into a single one.

**Example 4.5.** Consider again the query \( q' \) illustrated in Figure 4.2. Since the atom \( \neg S(x, z) \) in the topmost connected component \{\( R(x, y), \neg S(x, z), O(z) \}\) is negated, we first replace it with a positive atom \( S(x, z) \). Then, we combine all three atoms into a single atom \( R(x, y, z) \), as illustrated in Figure 4.2b, and replace these atoms in the query with the new atom. The new relation in the database will contain every answer to the query \( q_C(x, y, z) \colon R(x, y), S(x, z), O(z) \) on \( D \). Note that \( \neg V(t) \) is a connected component on its own, but we still replace it with a positive atom \( V(t) \).

Next, we use the results of Lemmas 4.2.1 and 4.2.3 to reduce the computation of \( \text{Shapley}(D, q, f) \) to the computation of \( \text{Shapley}(D', q', f) \) for a query \( q' \) where every exogenous atom corresponds to a non-exogenous atom such that the two have the exact same variables.

In the next lemma we will use the following notation. For an atom \( α \in \text{Atoms}(q) \), a variable \( v \in \text{Vars}(α) \) and a fact \( f \in R_α^D \), we denote by \( f[v] \) the value of the fact \( f \) in the attribute of \( R_α^D \) corresponding to the position of \( v \) in \( α \). For example, for \( α = R(x, y, z) \), we denote by \( f[y] \) the value of the fact \( f \) in the second attribute of \( R^D \).

Recall that \( \text{Atoms}_κ(q) \) and \( \text{Atoms}_κ(x)(q) \) are the sets of exogenous and non-exogenous atom in \( q \), respectively.

**Lemma 4.2.4.** Computing \( \text{Shapley}(D, q, f) \) can be efficiently reduced to computing \( \text{Shapley}(D', q', f) \) for a CQ \( q' \) without self-joins such that: (1) for every \( α \in \text{Atoms}_κ(q') \) there exists \( β \in \text{Atoms}_κ(x)(q') \) for which \( \text{Vars}(α) = \text{Vars}(β) \), and (2) \( q' \) does not have any non-hierarchical path.

**Proof.** As shown in Lemma 4.2.3, we can reduce the problem of \( \text{Shapley}(D, q, f) \), given \( D \) and \( f \), to that of computing \( \text{Shapley}(D', q', f) \) where \( q' \) is such that every exogenous variable occurs in a single atom of \( q' \). This means that every connected component of \( g_κ(q') \) contains a single atom. Moreover, we have that \( q' \) does not have a non-hierarchical path. For convenience, from now on, we refer to the query \( q' \) simply as \( q \) and to the database \( D' \) simply as \( D \), as we do not rely on the original query and
database in our proof. We show that we can further reduce the problem of computing Shapley($D,q,f$) to that of computing Shapley($D',q',f$), where for every $\alpha \in \text{Atoms}_x(q')$ there is $\beta \in \text{Atoms}_x(q')$ such that $\text{Vars}(\alpha) = \text{Vars}(\beta)$. We will do that by first removing the exogenous variables of $q$ and then adding to each exogenous atom all the variables occurring in the non-exogenous atom that “contains” it.

Let $\alpha \in \text{Atoms}_x(q)$. Lemma 4.2.1 implies that there exists $\beta \in \text{Atoms}_x(q)$ such that $\text{Vars}_x(\alpha) \subseteq \text{Vars}(\beta)$. We generate the query $q'$ in two steps. First, we remove from $\alpha$ every exogenous variable, and obtain a new atom $\alpha' = R_{\alpha'}(x_1, \ldots, x_n)$, where $x_1, \ldots, x_n$ are the non-exogenous variables in $\alpha$. Then, we replace the relation $R_{\alpha}$ in $D$ with the relation $R_{\alpha'}$ consisting of the set of answers to the query $q(x_1, \ldots, x_n) \vdash \alpha$ on $D$. In the next step, we obtain an atom $\alpha''$ by adding to $\alpha'$ every variable in $\text{Vars}(\beta) \setminus \text{Vars}_x(\alpha)$. That is, if $\{v_1, \ldots, v_m\}$ is the set of variables occurring in $\beta$ but not in $\alpha$, then $\alpha'' = R_{\alpha''}(x_1, \ldots, x_n, v_1, \ldots, v_m)$. Then, we obtain the relation $R^{D}_{\alpha''}$ from $R^{D}_{\alpha'}$ in the following way. From every $f = R_{\alpha'}(c_1, \ldots, c_n)$ in $R^{D}_{\alpha'}$, we generate $|\text{Dom}(D)|^m$ facts of the form $f = R_{\alpha''}(c_1, \ldots, c_n, d_1, \ldots, d_m)$, where $d_1, \ldots, d_m \in \text{Dom}(D)$, and add all of them to $R^{D}_{\alpha''}$. We denote by $q'$ the query obtained from $q$ by replacing the atom $\alpha$ with the atom $\alpha''$, and by $D'$ the database obtained from $D$ by replacing the relation $R_{\alpha}$ with the relation $R_{\alpha''}$. Note that $D_n = D'_n$.

We now prove that for every $E \subseteq D_n$ it holds that $(D_x \cup E) \models q$ if and only if $(D'_x \cup E) \models q'$. Let $E \subseteq D_n$ such that $(D_x \cup E) \models q$. Thus, there is a homomorphism $h$ from $q$ to $D_x \cup E$. In particular, $h$ maps the atom $\alpha$ to an exogenous fact $f \in R^{D}_{\alpha}$. Assume that $h(v) = c_v$ for every variable $v$ of $q$. Hence, for every non-exogenous variable $x_i$ in $\alpha$ we have that $h(x_i) = c_{x_i}$ and for every variable $v_j$ in $\text{Vars}(\beta) \setminus \text{Vars}_x(\alpha)$ we have that $h(v_j) = c_{v_j}$. From the construction of $D'$, if $h$ maps the atom $\alpha$ to a fact $f$ in $R^{D}_{\alpha}$, there is a fact $R_{\alpha''}(c_{x_1}, \ldots, c_{x_n}, c_{v_1}, \ldots, c_{v_m})$ in $D'$ and $h$ maps the atom $\alpha''$ in $q'$ to this fact. Every other atom $\gamma$ of $q'$ also appears in $q$ and we have not changed any other relation of $D$; hence, $h$ maps $\gamma$ to a fact of $D_x \cup E$ if and only if it maps $\gamma$ to a fact of $D'_x \cup E$. Therefore, $h$ is a homomorphism from $q'$ to $D'_x \cup E$, as we conclude that $(D'_x \cup E) \models q'$.

Next, let $E \subseteq D_n$ such that $(D'_x \cup E) \models q'$, as evidenced by a homomorphism $h$. Again, every atom $\gamma$ of $q'$ that is not $\alpha''$ also appears in $q$, and $h$ maps $\gamma$ to a fact of $D_x \cup E$ if and only if it maps $\gamma$ to a fact of $D'_x \cup E$. As for the atom $\alpha''$, the homomorphism $h$ maps it to a fact $f' \in R^{D'}_{\alpha''}$. Assume that $f' = R_{\alpha''}(c_{x_1}, \ldots, c_{x_n}, c_{v_1}, \ldots, c_{v_m})$. From the construction of $D'$, we have that there exists a fact $f$ in $R^{D}_{\alpha}$ such that $f[x_i] = c_{x_i}$ for every $i \in \{1, \ldots, n\}$. Assume that the exogenous variables in $\alpha$ are $u_1, \ldots, u_n$ and $f[u_j] = d_j$ for every $j \in \{1, \ldots, r\}$. Then, if we extend the mapping $h$ to a mapping $h'$ such that $h'(u_j) = d_j$ for every exogenous variable $u_j$ in $\alpha$ (and $h'(x) = h(x)$ for every other variable $x$ in $q$), then $h'$ will map the atom $\alpha$ in $q$ to the fact $f$. Note that this extension does not affect any other atom of $q$ since the exogenous variables of $\alpha$ do not occur in any other atom of $q$. Hence, the mapping $h'$ is a homomorphism from $q$ to $D_x \cup E$, and $(D_x \cup E) \models q$.
We can repeat this process for every exogenous atom of \( q \) and obtain a query \( q' \) satisfying the property of the lemma, such that Shapley\((D, q, f) = \text{Shapley}(D', q', f)\) for every \( f \in D_n \). Finally, we prove that \( q' \) does not have a non-hierarchical path. Let us assume, by way of contradiction, that \( q' \) has a non-hierarchical path induced by the atoms \( \alpha_x \) and \( \alpha_y \). Hence, in the Gaifman graph of \( q' \), there is a path \( x - v_1 - \cdots - v_n - y \) that does not pass through the variables of \( \alpha_x \) and \( \alpha_y \). We claim that the same path exists in the Gaifman graph of \( q \), in contradiction to the fact that \( q \) does not have a non-hierarchical path. Let \( v_i, v_{i+1} \) be two consecutive variables in the path. If \( v_i \) and \( v_{i+1} \) occur together in a non-exogenous atom of \( q' \), then they occur together in the same non-exogenous atom of \( q \), and \( v_i, v_{i+1} \) are also connected in the Gaifman graph of \( q \). Otherwise, \( v_i, v_{i+1} \) occur together in an exogenous atom of \( q' \). Since there are no exogenous variables in \( q' \), both \( v_i \) and \( v_{i+1} \) occur in non-exogenous atoms of \( q \). Moreover, since for every exogenous atom \( \alpha \) in \( q \) there exists a non-exogenous atom \( \beta \) of \( q \) such that \( \text{Vars}(\alpha) \subseteq \text{Vars}(\beta) \), we again conclude that \( v_i \) and \( v_{i+1} \) occur together in the same non-exogenous atom of \( q \), and \( v_i, v_{i+1} \) are also connected in the Gaifman graph of \( q \).

**Example 4.6.** Figure 4.2c illustrates the implications of Lemma 4.2.4 on the query \( q' \) of Example 4.2. We replace the relation \( R(x, v, z) \) with the relation \( T'(v) \) obtained from it by removing the exogenous variables \( x \) and \( z \). As for the atom \( V(t) \), it does not contain any exogenous variables, but the non-exogenous atom \( U(t, r) \) containing all the variables of \( V(t) \) also uses the variable \( r \); hence, we add this variable and obtain a new atom \( U'(t, r) \).

Our final observation is that a query \( q' \) satisfying the properties of Lemma 4.2.4 is hierarchical. This holds true since \( \text{Atoms}_x(q') \) does not contain a non-hierarchical triplet (otherwise, the original \( q \) would contain a non-hierarchical path). Adding an atom \( \alpha \) to a hierarchical query \( q \) such that \( \text{Vars}(\alpha) = \text{Vars}(\beta) \) for some atom \( \beta \) in \( q \) cannot change the non-hierarchical structure of the query.

We summarize the section with the algorithm \texttt{ExoShap}(\( D, q, f \)) (Algorithm 4.1) for computing the Shapley value for a self-join-free CQ that does not have a non-hierarchical path. The algorithm starts by modifying \( q \) and \( D \) according to the steps described throughout this section. First, it replaces the negated exogenous atoms of \( q \) with positive atoms, and the corresponding relations in \( D \) with their complement relations. Then, it combines the exogenous atoms in every connected component of \( g_e(q) \) into a single atom while joining the corresponding relations of \( D \). Finally, it removes the exogenous variables of \( q \), and adds to every exogenous atom the missing variables from the non-exogenous atom containing it. The final database is constructed from the Cartesian product of the projection of every exogenous relation \( R^D_\alpha \) to the attributes corresponding to the non-exogenous variables of \( \alpha \), and the set \( \{c_1, \ldots, c_r \mid c_i \in \text{Dom}(D)\} \), where \( r \) is the number of non-exogenous variables we have added to
Algorithm 4.1 \texttt{ExoShap}(D, q, f)

\begin{algorithmic}
  \For {negated $\alpha \in \text{Atoms}_x(q)$}
    \State Replace $\alpha$ in $q$ with $\overline{\alpha}$
    \State Replace $R_{\alpha}$ in $R^D$ with $\overline{R_{\alpha}}$
  \EndFor
  \For {$\{\alpha_1, \ldots, \alpha_k\} \in \text{ConnectedComponents}(g_x(q))$}
    \State \{$x_1, \ldots, x_n$\} $\leftarrow$ variables occurring in $\alpha_1, \ldots, \alpha_k$
    \State $q'(x_1, \ldots, x_n) :\alpha_1, \ldots, \alpha_k$
    \State Replace $\alpha_1, \ldots, \alpha_k$ in $q$ with $R_{\alpha}(x_1, \ldots, x_n)$
    \State Replace $R^D_{\alpha_1}, \ldots, R^D_{\alpha_k}$ with $R^D_{\overline{\alpha}} = q'(D)$
  \EndFor
  \For {$\alpha \in \text{Atoms}_x(q)$}
    \State Let $\beta \in \text{Atoms}_x(q)$ s.t. $\text{Vars}_x(\alpha) \subseteq \text{Vars}(\beta)$
    \State \{$y_1, \ldots, y_m$\} $\leftarrow$ non-exogenous variables occurring in $\alpha$
    \State $q'(y_1, \ldots, y_m) :\alpha$
    \State Replace $\alpha$ in $q$ with $R_{\alpha'}(x_1, \ldots, x_n)$
    \State Replace $R^D_{\alpha}$ with $R^D_{\overline{\alpha'}} = q'(D) \times \{(c_1, \ldots, c_{n-m}) \mid c_i \in \text{Dom}(D)\}$
  \EndFor
  \State \Return \texttt{Shapley}(D, q, f)
\end{algorithmic}

$\alpha$. Then, the algorithm invokes an algorithm for computing the Shapley value for hierarchical queries.

### 4.3 Application to Probabilistic Databases

We observe that our results in this chapter also apply to the problem of query evaluation over tuple-independent probabilistic databases [DS04]. Fink and Olteanu [FO16] have studied this problem for queries with negation. They considered the class $\text{RA}^-$ of queries that includes the CQ $\neg$'s. When restricting to CQ $\neg$'s, they established that query evaluation is possible in polynomial time for hierarchical CQ $\neg$'s, and it is $\text{FP}^\#P$-complete otherwise. The proofs of this section immediately provide a generalization of their result to account for deterministic relations, where the probability of every fact is 1. The only difference is that instead of using the algorithm for computing the Shapley value for hierarchical CQ $\neg$'s, we will use the algorithm for query evaluation over tuple-independent probabilistic databases for hierarchical CQ $\neg$'s. Hence, we obtain the following result (where $X$ is the set of deterministic relations).

**Theorem 4.7.** Let $S_X$ be a schema and let $q$ be a CQ without self-joins. If $q$ has a non-hierarchical path, then its evaluation over tuple-independent probabilistic databases is $\text{FP}^\#P$-complete. Otherwise, the query can be evaluated in polynomial time.

### 4.4 A Note about Endogenous Relations

In this section we study the implication of endogenous relations to the computation of the Shapley value, as an analogue to exogenous relations. Throughout this chapter, we considered exogenous relations as consisting of exogenous facts only, whereas the
non-exogenous relations in the database may include both endogenous and exogenous facts. We have seen that fixing a set of exogenous relations considerably changes the classification criterion for computing the Shapley value in self-join-free CQs. However, we do not know whether fixing endogenous relations in the schema may affect the computation of the Shapley value as well. Intuitively, endogenous facts in the database make the computation hard, as the Shapley value depends on the change in the query result for every possible subset of endogenous facts in the database.

Recall that in Livshits et al. proof of hardness for the query \( q() : R(x), S(x, y), T(y) \) (denoted as \( q_{RST} \)), they assumed that every fact in relation \( S \) is exogenous. Thus, the complexity of computing the Shapley value where every fact in the database considered endogenous for \( q_{RST} \) (and for every other non-hierarchical self-join-free CQ) is not clear. However, a recent result of Amarilli and Kimelfeld [AK19], shows that counting the number of subsets of facts in a database that satisfy a self-join-free Boolean query is \( \text{FP}^\#P \)-complete for non-hierarchical queries (this problem is known to be tractable for hierarchical queries). They refer to this problem as uniform reliability, and denote the size of \( \text{Mod}(Q, I) := \{ J \subseteq I : J \models Q \} \), which is the set of all subsets of a database instance \( I \) that models the query \( Q \), as \( \text{UR}(Q) \). We find that this result has a strong connection to the problem of computing the Shapley value for self-join-free CQs, where every fact in the database is considered endogenous. Our conjecture is that computing the Shapley value under this assumption is hard for the entire class of non-hierarchical self-join-free CQs as well. However, we do not have a hardness proof for every non-hierarchical CQ, but we do establish the following result that shows hardness for a large class of non-hierarchical CQs.

**Theorem 4.8.** Let \( q \) be a self-join-free non-hierarchical CQ and let \( D \) be a database where every fact is endogenous. If there exists a variable that occurs only in one atom of \( q \) and that atom belongs to a non-hierarchical triplet, then computing \( \text{Shapley}(D, q, f) \) is \( \text{FP}^\#P \)-complete.

**Proof.** \( q \) is not hierarchical, so there exist two variables, \( x \) and \( y \), such that \( A_x \not\subseteq A_y \), \( A_y \not\subseteq A_x \), and \( A_x \cap A_y \neq \emptyset \). Let \( z \) be a variable that occurs only in one atom of \( q \), denoted as \( \alpha_z \), such that \( \alpha_z \in (A_x \cup A_y) \). Clearly, both \( x \) and \( y \) occur in more than one atom of \( q \), hence, \( z \) is not \( x \) or \( y \). Let \( q' \) be a \( Q_{r,s,t} \) query as defined in [AK19], for \( r = |A_x \setminus A_y|, s = |A_x \cap A_y|, \) and \( t = |A_y \setminus A_x| \) (\( Q_{r,s,t} \) are self-join-free CQs of the form:

\[
Q_{r,s,t}() := R_1(x), ..., R_r(x), S_1(x, y), ..., S_s(x, y), T_1(y), ..., T_t(y)
\]

for some natural numbers \( r, s, t > 0 \)). We will now show a reduction from computing \( \text{UR}(q') \) for a database instance \( I \), to computing \( \text{Shapley}(D, q, f) \), where every fact in \( D \) is endogenous. The query \( q' \) satisfies that \( \text{UR}(q') \) is \( \text{FP}^\#P \)-complete by Theorem 3.4 in [AK19]; hence, the indicated reduction will provide the proof for this theorem.

Given a database instance \( I \) that contains \( n \) facts, we construct \( n + 1 \) database
instances \( D^j \) for every \( j \in \{1, ..., n+1\} \) as following. We associate every atom of \( q \) in \( A_x \cup A_y \) with an atom of \( q' \). For every fact in \( I \) that appears in the relation \( R_{\alpha'} \) where \( \alpha' \) is an atom of \( q' \), we add a fact \( f \) to \( D^j \) in the relation \( R_\alpha \), where \( \alpha \) is the corresponding atom to \( \alpha' \) in \( q \). We copy the values in the positions of \( x \) and/or \( y \) in \( \alpha' \) to the corresponding positions of \( x \) and/or \( y \) in \( f \). In every other variable position in \( \alpha \), we add a new constant \( c \) to \( f \), and in every constant position in \( \alpha \) we add the same constant to \( f \). Next, for every atom \( \alpha \) of \( q \) except for \( \alpha_z \) (including other atoms in \( A_x \cup A_y \)), we add to the relation \( R_\alpha \) in \( D^j \) one fact \( f \) as following. In every position that corresponds to a position of a variable that occurs in \( \alpha \), we add the constant \( c \) to \( f \), and in every position of a constant in \( \alpha \), we add the same constant to the corresponding position in \( f \). We will refer to these facts as \textit{helper facts}. Finally, we add to the relation \( R_{\alpha_z} \) in \( D^j \) \( j \) facts that we will refer to as \( j \) “copies”, as following. In every position of a variable in \( \alpha_z \) which is different from \( z \), we add the constant \( c \) in the corresponding position, and in every position of a constant in \( \alpha_z \) we add the same constant. In the positions of the variable \( z \), we add a new constant \( c_{z_i} \) to the \( i \)-th copy of this fact in \( R_{\alpha_z} \) (\( i \in \{1, ..., j\} \)). From now on, we denote the last fact that we added to \( R_{\alpha_z} \) as \( f \).

Next, we observe that the fact \( f \) changes the result of \( q \) over any instance \( D^j \) from false to true in a permutation \( \sigma \) if and only if the following conditions hold:

1. None of the copies of \( f \) are in \( \sigma_f \).

2. All of the helper facts are in \( \sigma_f \).

3. The set of facts in \( \sigma_f \) which are not the helper facts correspond to a subset \( J \subseteq I \), such that \( J \not\models q' \).

We now explain shortly why these three conditions must hold; since \( z \) occurs only in \( \alpha_z \), and the copies of \( f \) differ only in the positions of \( z \), then only the first copy can change \( q \) result when it is added to the database. The second condition must hold, due to the fact that every value in \( f \) is a new constant (except for the constants that occur in \( \alpha_z \)). Therefore, the homomorphism that satisfies \( q \) by adding \( f \) to the database has to map all of \( \alpha_z \) variables to the new constants, including \( x \) and/or \( y \). The connectivity of the atoms among \( A_x \cup A_y \) in \( q \) implies that both \( x \) and \( y \) are mapped to the new constants, while every other variable of \( q \) is always mapped to the new constant \( c \) as well, by the construction of \( D^j \). The new constant appears only in the helper fact of each relation that correspond to an atom in \( A_x \cup A_y \) (except for \( \alpha_z \)), and in relations that correspond to other atoms of \( q \) there is a single helper fact that must exist to satisfy \( q \). Thus, all of the helper facts must appear in \( \sigma_f \). As for the third condition, we can see clearly that if the subset \( J \) satisfy that \( J \models q' \), then the answer to \( q \) would have been true before adding \( f \) to \( D^j \). That is since the homomorphism from \( q' \) to \( J \) that maps \( x \) and \( y \) to values in \( I \) can be extended such that every other variable of \( q \) (including \( z \)) is mapped to \( c \), based on the fact that \( z \) occur in an atom of \( A_x \cup A_y \). Moreover, if \( J \not\models I \), then there is no homomorphism from \( q \) to \( D^j \) where every variable
of \( q \) is mapped to a value in \( I \). Furthermore, if \( x \) is mapped to \( c \), then \( y \) is mapped to \( c \) as well, since there is no fact in \( D_j \) that can be produced by mapping \( x \) to a value in \( I \) and \( y \) to \( c \) or otherwise. Hence, if all the helper facts are in \( \sigma_f \) along with the facts that correspond to the facts in \( J \), the mapping of every variable to the constant \( c \) is the only homomorphism that exists from \( q \) to \( \sigma_f \cup f \), and without \( f \) (or a copy of \( f \)), this homomorphism cannot exist, so we conclude that \( f \) indeed changes \( q \) result in this case.

Next, we denote the number of subsets \( J \subseteq I \) such that \( J \neq I \) and \( |J| = k \) as \( \text{UR}(q', k) \). According to the observation above, we deduce that:

\[
\text{Shapley}(D^j, q, f) = \sum_{k=0}^{n} \frac{\text{UR}(q', k)(m+k)!(n-k+j)!}{(n+m+j)!}
\]

where \( m \) is the number of helper facts (which is also the number of atoms in \( q \) except for \( \alpha_z \)).

Hence, we get a system of \( n+1 \) linear equations:

\[
\begin{pmatrix}
  m!(n+1)! & (m+1)!n! & \ldots & (m+n)!1! \\
  m!(n+2)! & (m+1)!(n+1)! & \ldots & (m+n)!2! \\
  \vdots & \vdots & \ddots & \vdots \\
  m!(2n+1)! & (m+1)!2n! & \ldots & (m+n)!(n+1)!
\end{pmatrix}
\begin{pmatrix}
  \text{UR}(q', 0) \\
  \text{UR}(q', 1) \\
  \vdots \\
  \text{UR}(q', n)
\end{pmatrix}
= \begin{pmatrix}
  \text{Shapley}(D^1, q, f) \cdot (n+m+1)! \\
  \text{Shapley}(D^2, q, f) \cdot (n+m+2)! \\
  \vdots \\
  \text{Shapley}(D^{n+1}, q, f) \cdot (2n+m+1)!
\end{pmatrix}
\]

Let us multiply every column \( i \) of the matrix by the constant \( \frac{i!}{(m+i-1)!} \). We will receive the same matrix from the proof of hardness for \( q_{\text{RS-\neg T}} \) (Lemma 3.1.4) which is non-singular, and therefore its determinant is not zero. Multiplying columns by a constant multiplies the determinant by a constant; hence, this matrix is non-singular as well, so there is a single solution to the equation system. By solving the equation system, we can compute:

\[
\text{UR}(q') = 2^n - \sum_{k=0}^{n} \text{UR}(q', k)
\]

and that concludes the reduction and the proof.

As for the class of self-join-free CQ\(^{-}\), the complexity of computing \( \text{UR}(Q) \) is still an open question for non-hierarchical queries. It is assumed to be a hard problem as well, based on the fact that \( \text{reliability} \) is known to be hard for such queries [FO16]. Again,
we rely on the connection between computing \( UR(Q) \) and computing the Shapley value when every fact in the database is endogenous; hence, we conjecture that computing the Shapley value for non-hierarchical CQ’s remains hard when every fact is considered endogenous. However, we cannot prove that this claim holds for a sub-class of non-hierarchical CQ’s in the same way as we did for CQs (without negation), since we cannot rely on the hardness of computing \( UR(Q) \) in this case.

Finally, we state some of the observations that were achieved by this point, regarding the complexity of computing the Shapley value in a model that accounts for relations in the database that contains only endogenous facts. First, we define endogenous relation to be a relation in the database that contains only endogenous facts. We fix a schema \( S \) and a set of endogenous relations in \( S \). Every relation which is not endogenous may contain both exogenous and endogenous facts.

**Corollary 4.9.** Let \( q \) be a self-join-free CQ\(^{-} \). If there exists a non-hierarchical triplet in \( q \), \((\alpha_x, \alpha_{x,y}, \alpha_y)\), such that \( R_{\alpha_x} \) and \( R_{\alpha_y} \) are the only endogenous relations in the schema, then \( \text{Shapley}(D, q, f) \) is \( \text{FP}^{\#P} \)-complete.

This follows directly from the hardness side proof of the dichotomy theorem in Chapter 3 and in [LBKS20], where the relations \( R_{\alpha_x} \) and \( R_{\alpha_y} \) in the database constructed in this proof contained only endogenous facts. Note that the requirement for \( R_{\alpha_x} \) and \( R_{\alpha_y} \) to be the only endogenous relations is necessary, due to the fact that every other relation in that database except for \( R_{\alpha_x} \) and \( R_{\alpha_y} \), may contain exogenous facts.

Next, we consider a model where every relation in the schema is classified as exogenous or as endogenous relation. The definition of non-hierarchical path in Section 4.1 can be easily extended to this model, by replacing the term non-exogenous atoms with endogenous atoms. Thus, by the same arguments as before, we deduce the following:

**Corollary 4.10.** Let \( q \) be a self-join-free CQ\(^{-} \). If there exists a non-hierarchical path induced by the atoms \( \alpha_x \) and \( \alpha_y \), such that \( R_{\alpha_x} \) and \( R_{\alpha_y} \) are the only endogenous relations in the schema, then \( \text{Shapley}(D, q, f) \) is \( \text{FP}^{\#P} \)-complete.

To conclude, we conjecture that accounting for endogenous relations does not effect the complexity of the Shapley value, unlike accounting for exogenous relations. In other words, we assume that the restriction of relations to contain only endogenous facts (instead of both endogenous and exogenous facts) preserves the hardness of computing the Shapley value. However, we only have partial hardness results in this model, while the full classification of the problem in the endogenous relations model remains an open question.

48
Chapter 5

Approximation and Relevance

As seen in the previous sections, computing the exact Shapley value is often hard. Hence, in this chapter, we consider its approximate computation. There exists a Multiplicative Fully-Polynomial Randomized Approximation Scheme (FPRAS) for computing the Shapley value for any CQ and, in fact, for any union of CQs [LBKS20]. Here, we show that the addition of negation changes the complexity picture completely. Recall that an FPRAS for a numeric function $f$ is a randomized algorithm $A(x, \epsilon, \delta)$, where $x$ is an input for $f$ and $\epsilon, \delta \in (0, 1)$. The algorithm returns an $\epsilon$-approximation of $f(x)$ with probability at least $1 - \delta$ in time polynomial in $x$, $1/\epsilon$ and $\log(1/\delta)$. More formally, for an additive (or absolute) FPRAS we have that:

$$\Pr \left[ f(x) - \epsilon \leq A(x, \epsilon, \delta) \leq f(x) + \epsilon \right] \geq 1 - \delta,$$

and for a multiplicative (or relative) FPRAS we have that:

$$\Pr \left[ \frac{f(x)}{1 + \epsilon} \leq A(x, \epsilon, \delta) \leq (1 + \epsilon)f(x) \right] \geq 1 - \delta.$$

5.1 Additive vs. Multiplicative Approximation

We start by showing that there exists an additive FPRAS for computing the Shapley value for CQ’s. The additive FPRAS for CQ’s is a generalization of the additive FPRAS for CQs. We observe that when negated atoms are allowed along with self-joins, a fact $f$ may change the query result from false to true in one permutation, while changing the query result from true to false in another permutation. For a random permutation $\sigma$ of the facts in $D_n$, the result of $q(D_x \cup \sigma_f \cup \{f\}) - q(D_x \cup \sigma_f)$ is a random variable $x \in \{-1, 0, 1\}$. By using the Hoeffding bound for sums of independent random variables in bounded intervals [Hoe94], we get an additive FPRAS for computing Shapley($D, q, f$) by taking the average value of $x$ over $O(\log(1/\delta)/\epsilon^2)$ samples of random permutations.

For CQs, an additive FPRAS is also a multiplicative FPRAS [LBKS20], due to the gap property:
Definition 5.1 (Gap Property). A query \( q \) satisfies the gap property if there exists a polynomial \( p \) such that for all databases \( D \) and facts \( f \in D_n \), it holds that \(|\text{Shapley}(D, q, f)|\) is either zero or at least \( 1/p(|D|) \).

While the gap property holds for CQs, we will now show that this property does not hold when negation is added to the picture; hence, this approach for obtaining a multiplicative approximation of the Shapley value is no longer valid.

We now present this result formally and show that the gap property is violated by every CQ with negation.

Theorem 5.2. Let \( q \) be a satisfiable CQ\(^\neg\) with at least one negated atom that contains a variable. Then, there exists a constant \( k \), a sequence \( \{D_n\}_{n=1}^\infty \) of databases and a fact \( f \) such that \(|D_n| = \Theta(n^k)\) and \( 0 < |\text{Shapley}(D_n, q, f)| \leq 2^{-n} \).

Before we bring the full proof for this theorem, we explain the intuition behind it. Observe the following simple CQ:\(^\neg\): \( q() : \neg R(x), \neg S(x) \). We construct a database \( D \) in the following way. For \( i \in \{1, \ldots, n\} \) we add an exogenous fact \( R(i) \) and an endogenous fact \( S(i) \) to the database. In addition, for \( i \in \{0, n+1, \ldots, 2n\} \) we add an endogenous fact \( R(i) \) to the database. We denote \( f = R(0) \), and we will now show that the fact \( f \) does not satisfy the gap property. First, note that \( D_\emptyset \models q \) since for every \( i \in \{1, \ldots, n\} \), there is a homomorphism \( h \) from \( q \) to \( D_\emptyset \), where \( h(x) = i \). For the fact \( f \) to change the query result from false to true, we first need to add all the endogenous facts of the form \( S(i) \) in \( D \) to a permutation. Moreover, the first endogenous fact of the form \( R(i) \) \( (i \in \{0, n+1, \ldots, 2n\}) \) that will be added to the permutation will change the query result from false to true, and no fact could change it back to false; hence, the fact \( f \) has to appear before all these facts in a permutation. Overall, there is exactly one subset \( E \subseteq D_n \), such that \( (D_\emptyset \cup E) \not\models q \) and \( (D_\emptyset \cup E \cup \{f\}) \models q \). It holds that \(|E| = n\) and \(|D_n| = 2n + 1\); thus, \(|D| = \Theta(n)\) and we conclude the following.

\[
0 < |\text{Shapley}(D, q, f)| = \frac{n!n!}{(2n + 1)!} \leq 2^{-n}
\]

However, constructing a database that would violate the gap property is no longer trivial when self-joins are involved. Consider the following CQ:\(^\neg\):

\[
q() : R(x, y), S(z, w), \neg R(x, w), T(z)
\]

We are interested in creating an instance \( D \) such that: (1) \( D_\emptyset \models q \), (2) \(|D_n| = 2n + 1\), and (3) \( (D_\emptyset \cup E) \not\models q \) for a single \( E \subseteq D_n \), which is of size \( n \). The database constructed for the following query is illustrated in Figure 5.1. The gray facts are exogenous facts while the rest are endogenous. The white fact \( f \) is the one that violates the gap property, the green facts are the \( n \) facts that come before \( f \) in a permutation where \( f \) changes the query result (these are the facts of the subset \( E \)), and the yellow facts are the \( n \) endogenous facts that come after \( f \) in such a permutation. We now explain the way we
choose to construct the database. First, for every \( i \in \{1, \ldots n \} \) we add the exogenous facts \( R(c_{xi}, c_{yi}) \), \( S(c_{zi}, c_{ui}) \), \( T(c_{zi}) \), and we denote as \( D' \) the union of all 3n facts. Observe that \( D' \models q \) due to the following homomorphisms: \( h_i(x) = c_{xi}, \ h_i(y) = c_{yi}, \ h_i(z) = c_{zi}, \ h_i(w) = c_{ui} \). Moreover, we add the endogenous fact \( R(c_{xi}, c_{yi}) \) (denoted as \( f_i \)), to assure the existence of \( n \) different facts that would violate every \( h_i \). Note that we could add fewer exogenous facts than we did; nevertheless, we use a uniform method of adding an exogenous fact for every positive atom to demonstrate the general case.

Observe that we created some additional homomorphisms that are not violated by the facts of the form \( R(c_{xi}, c_{yi}) \), such as \( h(x) = c_{xi}, \ h(y) = c_{yi}, \ h(z) = c_{zi}, \ h(w) = c_{ui} \). To avoid the case where \( (D_x \cup \{f_1, \ldots, f_n\}) \models q \), we add to the database in advanced a minimal set of exogenous facts that would violate every new homomorphism (such as the fact \( R(c_{xi}, c_{yi}) \) for example). We denote as \( F \) the set of facts for which \( (D' \cup F) \not\models q \). Note that \( f_1, \ldots, f_n \) belong to \( F \) as well, and these are the only endogenous facts of \( F \), while the rest are exogenous.

Next, we want to create a condition where we have another \( n + 1 \) endogenous facts in the database, such that adding each one of these facts will turn \( q \) result to true (and nothing could change the result back to false). To achieve this, we add \( n + 1 \) minimal sets of facts that would satisfy the query. For every \( i \in \{0, n + 1, \ldots, 2n \} \) we add the exogenous fact \( S(c_{zi}, c_{ui}) \) and the endogenous fact \( T(c_{zi}) \) (denoted as \( f_i \)), such that \( h(x) = c_{xi}, \ h(y) = c_{yi}, \ h(z) = c_{zi}, \ h(w) = c_{ui} \) is a homomorphism. However, it is not trivial why having additional exogenous facts cannot lead to a situation where \( (D_x \cup F) \models q \) (thus there will be no subset \( E \subset D_n \) of size \( n \) for which \( (D_x \cup E) \not\models q \)).

In the following proof we show how we guarantee that \( (D_x \cup F) \not\models q \), and that overall our construction indeed violates the gap property for every possible case.

Proof. (of Theorem 5.2) Let \( q \) be a CQ and let \( \alpha_n \in \text{Neg}(q) \) be a negative atom of \( q \) that contains a variable. Let \( x \) be an arbitrary variable in \( \text{Vars}(\alpha_n) \). We construct \( n \) databases \( D_i \) for every \( i \in \{1, \ldots, n \} \) as follows. For every positive atom \( \alpha \in \text{Pos}(q) \), we add an exogenous fact \( f \) to the relation \( R_\alpha \) in \( D_i \) obtained from \( \alpha \) by mapping every variable \( v \in \text{Vars}(\alpha) \) to the value \( c_{vi} \), where \( c_{vi} \) is a unique constant associated with \( v \) and \( i \) (similarly to the canonical database of a CQ). Every constant in \( \alpha \) is mapped to
itself in \( f \). Next, we consider \( D' \) to be the union of all \( D_i \). Let \( F \) be a minimal set of facts such that \((D' \cup F) \models q\). There must be such \( F \), since we can always add facts to \( R_{\alpha_n} \) until every homomorphism that exists from \( q \) to \( D' \) is violated by the atom \( \alpha_n \).

Let \( f_i \) be the fact in \( F \) that obtained by mapping every variable \( v \) in \( \alpha_n \) to the value \( c_{v_i} \) and every constant in \( \alpha_n \) to itself. Clearly, \( f_i \) is in \( F \), otherwise \((D' \cup F) \not\models q\) by the homomorphism that maps every variable \( v \) to the value \( c_{v_1} \). Moreover, \( f_1, \ldots, f_n \) are different facts in \( F \), since the unique value \( c_{v_i} \) appears only in \( f_i \). We denote the set of values that appear in \( D' \cup F \) as \( \text{Dom}(D') \). Note that the constants that occur in positive atoms of \( q \) and in \( \alpha_n \) belong to \( \text{Dom}(D') \). In the next step, we denote as \( D_q \) a minimal database such that \((D' \cup F \cup D_q) \models q\) and every fact in \( D_q \) contains a value \( c_q \not\in \text{Dom}(D') \). Such \( D_q \) must exist, since \( q \) is satisfiable. We create \( n+1 \) copies \( D_0, D_{n+1}, \ldots, D_{2n} \) of \( D_q \), such that \((\text{Dom}(D_i) \setminus \text{Dom}(D')) \cap (\text{Dom}(D_j) \setminus \text{Dom}(D')) = \emptyset\) for all \( i, j \in \{0, n+1, \ldots, 2n\} \). We denote by \( f_i \) the fact for which \((D' \cup F \cup D_i \setminus \{f_i\}) \not\models q\) for every \( i \in \{0, n+1, \ldots, 2n\} \). As all the databases \( D_0, D_{n+1}, \ldots, D_{2n} \) are copies of \( D_q \) with disjoint sets of new values, it holds that for every new value \( c_1 \in \text{Dom}(D_i) \) there is a corresponding value \( c_j \in \text{Dom}(D_j) \), such that if we replace every value \( c_1 \) in the database \( D' \cup F \cup D_i \) with its corresponding value \( c_j \), we will obtain the database \( D' \cup F \cup D_j \). Finally, we construct a database \( D \) by taking the union of the databases \( D_0, \ldots, D_{2n} \) along with the facts in \( F \). Every fact in \( D \) except for \( \{f_0, \ldots, f_{2n}\} \) will be exogenous. We will show that the fact \( f_0 \) does not satisfy the gap property.

**Claim 1:** \( D_x \models q \). There is a homomorphism \( h \) from \( q \) to \( D_1 \) (and, in fact, to every \( D_i \) for \( i \in \{1, \ldots, n\} \)), and we now show that the same \( h \) is a homomorphism from \( q \) to \( D_x \). Since \( q \) is safe, the variable \( x \) that occurs in \( \alpha_n \) occurs in a positive atom of \( q \) as well. The homomorphism that maps every variable \( v \) to the value \( c_{v_1} \) (and maps \( x \) to \( c_{x_1} \) in particular) is a homomorphism from \( q \) to \( D_x \). This holds true since \( h \) maps the negative atom \( \alpha_n \) to the fact \( f_i \) in \( D \) (by the constriction of \( D_i \)), but \( f_i \not\in D_x \). If a different negative atom \( \alpha \) of \( q \) is mapped to to a fact in \( D_x \), by the construction of \( D_1 \), we conclude that there is a positive atom of \( q \) which is identical to \( \alpha \). This is a contradiction to the assumption that \( q \) is satisfiable. Thus, \( h \) is a homomorphism from \( q \) to \( D_x \) and \( D_x \models q \).

**Claim 2:** \((D_x \cup \{f_1, \ldots, f_n\}) \not\models q \). Assume, by way of contradiction, that this is not the case. Then, there is a homomorphism \( h \) from \( q \) to \( D_x \cup \{f_1, \ldots, f_n\} \). Assume that \( h \) maps the positive atoms of \( q \) to the facts \( g_1, \ldots, g_m \). Clearly, it cannot be the case that \( \{g_1, \ldots, g_m\} \subseteq (D' \cup F) \) (recall that \( D' \) is the union of all \( D_i \) for \( i \in \{1, \ldots, n\} \)). This holds true since \((D' \cup F) \not\models q \); therefore, there exists \( g \) such that \( g \in D_i \) for \( i \in \{0, n+1, \ldots, 2n\} \). Moreover, it cannot be the case that \( \{g_1, \ldots, g_m\} \subseteq (D' \cup F \cup D_i \setminus \{f_i\}) \), since \((D' \cup F \cup D_i \setminus \{f_i\}) \not\models q \); therefore, there is a fact \( g' \in \{g_1, \ldots, g_m\} \) such that \( g \in D_j \) for \( j \in \{0, n+1, \ldots, 2n\} \setminus \{i\} \). Let \( h_i \) be a homomorphism from \( q \) to \( D_x \cup \{f_1, \ldots, f_n\} \) in which the set of variables that are mapped to values in \( \text{Dom}(D_i) \cup \text{Dom}(D') \) is maximal. Since every fact in \( D_j \) contains values from \( \text{Dom}(D_j) \setminus \text{Dom}(D') \), and particulay \( g' \), we deduce that there is a variable \( y \in \text{Vars}(q) \) such that
\( h_1(y) \in (\text{Dom}(D_j) \setminus \text{Dom}(D')) \). Let \( V \subseteq \text{Vars}(q) \) be the subset of variables of \( q \) such that for every \( v \in V \), \( h_i(v) \in \text{Dom}(D_j) \setminus \text{Dom}(D') \) (note that \( y \in V \)). We define the mapping \( h'_i : \text{Vars}(q) \to \text{Dom}(D) \) as follows:

\[
\begin{cases}
  h'_i(v) = c_i, & v \in V, h_i(v) = c_j \\
  h'_i(v) = h_i(v), & \text{otherwise}
\end{cases}
\]

where \( c_i \) is the corresponding value in \( \text{Dom}(D_j) \setminus \text{Dom}(D') \) to \( c_j \). We now prove that \( h'_i \) is a homomorphism from \( q \) to \( D_k \cup \{f_1, \ldots, f_n\} \). Let \( \alpha_1, \ldots, \alpha_k \) be the positive atoms of \( q \) in which variables from \( V \) occur. \( h_i \) maps these atoms to the facts \( \{g_{i1}, \ldots, g_{ik}\} \subseteq D_j \). Since \( D_i \) is a copy of \( D_j \), there are corresponding facts \( \{g'_{i1}, \ldots, g'_{ik}\} \subseteq D_i \). Let \( \alpha_r \) be an atom in \( \alpha_1, \ldots, \alpha_k \) such that \( h_i \) maps \( \alpha_r \) to \( g_{ik} \). Every variable in \( \alpha_r \) is mapped by \( h_i \) to a value in \( \text{Dom}(D_j) \). Hence, the mapping \( h'_i \) maps every variable \( v \in (V \cap \text{Vars}(\alpha_r)) \) to the corresponding value to \( h_i(v) \) in \( \text{Dom}(D_j) \setminus \text{Dom}(D') \), and every other variable in \( v \in \text{Vars}(\alpha_r) \) to \( h_i(v) \in \text{Dom}(D') \), by the construction of \( D \). Therefore, \( h'_i \) maps the atom \( \alpha_r \) to the fact \( g'_{ik} \), which is in \( (D_x \cup \{f_1, \ldots, f_n\}) \) as well. As for the rest of \( q \) positive atoms, they are mapped by \( h'_i \) to same fact that \( h_i \) mapped them. Moreover, if a negative atom \( \alpha \) of \( q \) is mapped by \( h'_i \) to a fact \( g \in (D_x \cup \{f_1, \ldots, f_n\}) \), a variable from \( V \) must occur in \( \alpha \) (otherwise \( h_i \) maps this atom to \( g \) as well). In this case \( g \in (D_i \setminus \{f_i\}) \), and every variable in \( \text{Vars}(\alpha) \) is mapped to a value in \( \text{Dom}(D_i) \). However, \( h_i \) would map \( \alpha \) to the corresponding fact \( g' \in (D_j \setminus \{f_j\}) \), which is a contradiction to the fact that \( h_i \) is a homomorphism. Finally, we deduce that \( h'_i \) is a homomorphism, where the set of variables of \( q \) that are mapped to values in \( \text{Dom}(D_i) \cup \text{Dom}(D') \) is a proper superset of the variables that are mapped by \( h_i \) to values in this domain (at least due to the variable \( y \)), which is a contradiction.

**Claim 3:** \( (D_x \cup E) \models q \) for every \( E \subseteq D_n \) such that \( \{f_1, \ldots, f_n\} \not\subseteq E \). Let \( f_i \not\in E \) such that \( i \in \{1, \ldots, n\} \). Let \( h \) be the homomorphism from \( q \) to \( D_i \) where every variable \( v \) is mapped to \( c_{vi} \). We claim that \( h \) is a homomorphism from \( q \) to \( D_x \cup E \). As we showed before, the only negative atom that \( h \) can map to a fact in \( D_x \cup E \) is \( \alpha_n \), and it can only be mapped to \( f_i \) which does not appear in the database. Hence, \( h \) is a homomorphism from \( q \) to \( D_x \cup E \) as well.

**Claim 4:** for each \( E \subseteq D_n \), \( \{f_0, f_{n+1}, \ldots, f_{2n}\} \cap E \neq \emptyset \), we have that \( (D_x \cup E) \models q \). First, if \( \{f_1, \ldots, f_n\} \not\subseteq E \), then \( D_x \cup E \models q \) according to claim 3; thus, we can assume that \( F \subseteq (D_x \cup E) \). Let \( f_i \in E \) such that \( i \in \{0, n+1, \ldots, 2n\} \). Since \( (D' \cup F \cup D_i) \models q \), there is a homomorphism \( h \) from \( q \) to \( D' \cup F \cup D_i \). We claim that \( h \) is a homomorphism from \( q \) to \( D_x \cup E \) as well. Clearly, every positive atom of \( q \) is mapped by \( h \) to a fact in \( D_x \cup E \) (since \( (D' \cup F \cup D_i) \subseteq (D_x \cup E) \)). If \( h \) maps a negated atom of \( q \) to a fact \( f \in (D_x \cup E) \), then \( f \in (D' \cup F \cup D_i) \). Otherwise, we get that \( h \) maps a variable of \( q \) to a value which is not in \( \text{Dom}(D') \cup \text{Dom}(D_i) \), and that is not possible by the construction of \( D \). However, having that \( f \in (D' \cup F \cup D_i) \) is a contradiction to the existence of the homomorphism \( h \). Hence, \( h \) is a homomorphism from \( q \) to \( D_x \cup E \) as well.
By the four claims indicted above, we conclude that the fact \( f_0 \) must be added in a permutation before any of the facts \( f_{n+1}, \ldots, f_{2n} \) and after all the facts \( f_1, \ldots, f_n \) to affect the query result. Hence, there is exactly one subset \( E \) of endogenous facts in \( D \), containing \( n \) facts, that should appear before \( f \) in a permutation where it changes the query result from false to true; therefore, the number of such permutations is \( \frac{n! \cdot n!}{(2n+1)!} \) (as the total number of endogenous facts is \( 2n+1 \)).

Finally, since we consider data complexity, we can assume that the sizes of each \( D_i \) and \( |\text{Dom}(D_i)| \) are bounded by some constant \( k \). The number of facts in \( F \) is bounded by \( (n \cdot k)^k \), since we consider every possible mapping of the variables to values in \( \text{Dom}(D') \). Hence, the database \( D \) contains \( \Theta(n^k) \) facts. Therefore, we have that \( n = \Omega(\sqrt[3]{|D|}) \), and the Shapley value of \( f_0 \) w.r.t. \( q \) and \( D \) is \( \frac{n! \cdot n!}{(2n+1)!} < \frac{1}{2^n} \). Overall, we have that:

\[
0 < \text{Shapley}(D, q, f_0) \leq 2^{-n} = 2^{-\Theta(|D|)}
\]

and that concludes our proof.

Theorem 5.2 implies that we need at least \( 2^{\Theta(|D|)} \) sample permutations (and, in particular, exponential time) to obtain a multiplicative approximation from the additive one. This does not mean that there is no multiplicative approximation for \( \text{CQ}^- \)'s; however, we will show that there are \( \text{CQ}^- \)'s for which a multiplicative approximation does not exist at all (under conventional complexity assumptions).

## 5.2 Hardness of Multiplicative Approximation

We now explore the complexity of computing a multiplicative approximation for the Shapley value through a connection to the problem of relevance to the query.

**Definition 5.3 (Relevance).** Let \( q \) be a Boolean query and \( D \) a database. A fact \( f \in D_n \) is relevant to \( q \) if \( q(D_x \cup E) \neq q(D_x \cup E \cup \{f\}) \) for some \( E \subseteq D_n \); we then say that \( f \) is positively (resp., negatively) relevant to \( q \) if \( q(D_x \cup E \cup \{f\}) \) is true (resp., false).

This problem of determining whether a fact is relevant to a query is strongly related to the approximation problem, as we cannot obtain a multiplicative approximation in cases where we cannot decide if the Shapley value is zero or not. In turn, deciding on zeroness is related to the relevance problem. Clearly, if \( \text{Shapley}(D, q, f) \neq 0 \), then \( f \) is relevant to \( q \). However, it may be the case that \( f \) is relevant to \( q \) but \( \text{Shapley}(D, q, f) = 0 \), as the following example shows.

**Example 5.4.** Consider the query \( q() : = \neg R(x, y), \neg R(y, x) \) and the following database \( \{R(1, 2), R(2, 1)\} \) where both facts are endogenous. The fact \( R(1, 2) \) is positively relevant for \( E = \emptyset \), and it is negatively relevant for \( E = \{R(2, 1)\} \). Therefore, the number of permutations where \( f \) changes the query result from false to true is equal to the
number of permutations where $f$ changes the result from true to false and we have that $\text{Shapley}(D, q, f) = 0$. ■

Nevertheless, there are cases where the relevance problem coincides with the problem of deciding whether the Shapley value is zero. The Shapley value of a relevant fact $f$ can be zero if and only if $f$ is both positively and negatively relevant. This may be the case if and only if $f$ belongs to a relation that appears both as a positive and a negative atom in the query. We call a relation symbol polarity consistent if it appears in $q$ only in positive atoms or only in negative atoms. A fact over a polarity-consistent relation symbol is relevant to $q$ if and only if $\text{Shapley}(D, q, f) \neq 0$.

**Example 5.5.** Consider again the queries of our running example (Example 2.2). Clearly, in the queries $q_1$ and $q_2$, every relation is polarity consistent, as the queries are self-join-free. The same holds for the query $q_3$, as Adv and Reg occur only in positive atoms while TA occurs only in negative atoms. The query $q_4$, on the other hand, contains both polarity-consistent relations (i.e., Adv) and relations that occur in both positive and negative atoms (i.e., TA and REG). In this case, a fact $f$ in the relation Adv is relevant to $q_4$ if and only if $\text{Shapley}(D, q_4, f) > 0$. However, for a fact $f$ in TA it may be the case that $f$ is relevant to $q_4$ while $\text{Shapley}(D, q_4, f) = 0$.

It is straightforward to show that the relevance to a CQ without negation can be decided in polynomial time. The problem is known to be NP-complete for Datalog programs with recursion [BS17]. We now show that there exists a CQ $\neg q$ containing a polarity-consistent relation $T$, such that the relevance of a $T$-fact to $q$ is NP-complete. (Hence, so is the problem of deciding if the Shapley value is zero.)

Consider the following CQ $\neg q$:

$$ q_{\neg RST \neg R}(z), \neg R(x), \neg R(y), R(z), R(w), S(x, y, z, w) $$

We prove the following.

**Proposition 5.6.** Deciding whether $f \in T^D$ is relevant to $q_{\neg RST \neg R}$, given $D$ and $f$, is NP-complete.

The following lemma states the NP-completeness of $(2^+, 2^-, 4^{++})$-SAT. (We found it easier to prove it directly rather than showing that it falls on the negative side.

![Figure 5.2: The database constructed in the proof of Proposition 5.6 for $(x_1 \lor x_2) \land (\neg x_1 \lor \neg x_3) \land (x_3 \lor x_4 \lor \neg x_1 \lor \neg x_2)$.

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>c</td>
</tr>
<tr>
<td>b</td>
<td>2</td>
<td>a</td>
</tr>
<tr>
<td>c</td>
<td>3</td>
<td>b</td>
</tr>
<tr>
<td>d</td>
<td>4</td>
<td>c</td>
</tr>
</tbody>
</table>
of Schaefer’s dichotomy theorem [Sch78].) As a preface to our proof, we define the
$(3^+, 2^-)$-SAT problem: given a monotone 3CNF formula $\varphi$ where every literal is positive,
and a monotone 2CNF formula $\varphi'$ where every literal is negative, defined over the
same variables as $\varphi$, is $\varphi \land \varphi'$ satisfiable? We first prove that this problem is
NP-complete using a reduction from the 3-colorability problem. Next, we define the
$(2^+, 2^-, 4^-)$-SAT problem, where the input is a conjunction of clauses of the following
forms: (1) $(x_i \lor x_j)$, (2) $(\neg x_i \lor \neg x_j)$, or (3) $(x_i \lor x_j \lor \neg x_k \lor \neg x_l)$. We prove that
this problem is NP-complete using a reduction from the $(3^+, 2^-)$-SAT problem. Then,
we will construct a reduction from the $(2^+, 2^-, 4^-)$-SAT problem to that of deciding
whether $f$ is relevant.

Lemma 5.2.1. The $(2^+, 2^-, 4^-)$-SAT problem is NP-complete.

Proof. Given an undirected graph $G = (V, E)$ and a set of three colours $C = \{c_1, c_2, c_3\}$,
we will build a $(3^+, 2^-)$-CNF formula denoted as $\varphi$. For every $v \in V$ and every
$c_i \in C$, we introduce a variable $x_i^v$. For every vertex $v \in V$, we introduce a clause
$x_1^v \lor x_2^v \lor x_3^v$. For every edge $(u, v) \in E$ and for every $c_i \in C$, we introduce a clause
$\neg x_i^u \lor x_i^v$. Finally, for every two colours $c_i \neq c_j$ and every $v \in V$, we introduce a clause
$\neg x_i^v \lor \neg x_j^v$. We say that $G$ has a valid 3-colouring $h$, if $h$ is a mapping $h : V \to C$,
such that every vertex in $V$ is mapped to a single color in $C$, and every edge $(u, v)$ in
$G$ satisfies that $h(u) \neq h(v)$.

The correspondence of the reduction is rather clear. If $G$ has a valid 3-colouring $h$,
the assignment $z$ assigning the value 1 to every variable $x_i^v$ such that $h(v) = c_i$ and the
value 0 to the rest of the variables satisfies every clause in $\varphi$. Every clause of the form
$(x_1^u \lor x_2^u \lor x_3^u)$ is satisfied since $h$ assigns a color to each vertex. Every clause of the
form $(\neg x_i^u \lor \neg x_j^v)$ is satisfied since in a valid colouring, two adjacent vertices cannot
be mapped to the same color. In addition, $h$ maps every vertex to a single color in $C$;
therefore, each clause of the form $(\neg x_i^v \lor \neg x_j^v)$ is satisfied as well. Overall, we have
that $\varphi$ is satisfiable.

Next, assume that $\varphi$ is satisfiable, and let $z$ be a satisfying assignment. We claim
that the coloring $h$ defined by $h(v) = c_i$ if $z(x_i^v) = 1$ is a valid 3-coloring of $G$. Since all
the clauses of the form $(x_1^v \lor x_2^v \lor x_3^v)$ are satisfied, for every vertex $v$, the assignment
$z$ assigns the value 1 to at least one variable $x_i^v$. Moreover, since the clauses of the form
$(\neg x_1^v \lor \neg x_2^v)$ are satisfied, for every $v$ the assignment $z$ assigns the value 1 to at most
one variable $x_i^v$. Hence, we conclude that $z$ assigns the value 1 to exactly one variable
of the form $x_i^v$ for every $v \in V$, and $h$ is indeed a coloring. Finally, since the clauses
of the form $(\neg x_i^u \lor \neg x_i^v)$ are satisfied, it cannot be the case that $z(x_i^u) = z(x_i^v) = 1$;
hence, $h$ does not map two vertices connected by an edge in $G$ to the same color.

Next, we reduce the $(3^+, 2^-)$-SAT problem to the $(2^+, 2^-, 4^-)$-SAT problem.
Given an input $\varphi$ to the first problem, we build an input $\varphi'$ to the second problem

---

1This proof is inspired by a proof given in https://cs.stackexchange.com/questions/16634/complexity-of-monotone-2-sat-problem.
in the following way. Every clause in \( \varphi \) of the form \((\neg x_i \lor \neg x_j)\) remains the same in \( \varphi' \). Every clause of the form \((x_i \lor x_j \lor x_k)\) in \( \varphi \) is replaced by three clauses in \( \varphi' \): (1) \((x_i \lor x_j \lor \neg y \lor \neg y)\), (2) \((x_k \lor y)\), and (3) \((\neg x_k \lor \neg y)\), where \( y \) is a new unique variable introduced for every clause in \( \varphi \). We claim that the clause \((x_i \lor x_j \lor x_k)\) is satisfiable if and only if the formula \((x_i \lor x_j \lor \neg y \lor \neg y) \land (x_k \lor y) \land (\neg x_k \lor \neg y)\) is satisfiable. This holds true since in every satisfying assignment \( z \) to the original clause, we either have that \( z(x_k) = 1 \), in which case we satisfy the new formula by defining \( z(y) = 0 \), or we have that \( z(x_k) = 0 \), in which case \( z(x_i) = 1 \) or \( z(x_j) = 1 \) and by defining \( z(y) = 1 \) we again satisfy the new formula. Now, given a satisfying assignment to the new formula, we either have that \( z(x_i) = 1 \) or \( z(x_j) = 1 \) in which case the original formula is clearly satisfied, or we have that \( z(y) = 0 \) in which case \( z(x_k) = 1 \) (otherwise, the clause \((x_k \lor y)\) is not satisfied) and again, the original formula is satisfied. Since we use different variables \( y \) for different clauses, we can assign the required value to each one of these variables, and that concludes our proof.

Next, we give a reduction from \((2^+, 2^-, 4^+)\)-SAT to the problem of deciding whether a fact \( f \in T^D \) is relevant to \( q_{\text{RST-R}} \). Given a formula \( \varphi \in (2^+, 2^-, 4^+) \), we build the input database \( D \) to our problem as follows: for every variable \( x_i \) in \( \varphi \) we add an endogenous fact \( R(i) \), and an exogenous fact \( T(i) \) to \( D \). For every clause \((x_i \lor x_j)\) in \( \varphi \), we add an exogenous fact \( S(i, j, a, a) \) where \( a \) is a new constant. For every clause \((\neg x_i \lor \neg x_j)\) we add an exogenous fact \( S(b, b, i, j) \) where \( b \) is a new constant. For every \((x_i \lor x_j \lor \neg x_k \lor \neg x_l)\) in \( \varphi \) we add an exogenous fact \( S(1, j, k, 1) \). In addition, we add the exogenous facts \( S(d, d, c, c), R(a), R(c), T(a) \) where \( c \) and \( d \) are new constants, and an endogenous fact \( T(c) \) which we denote as \( f \). Figure 5.2 illustrates the database constructed for the formula \((x_1 \lor x_2) \land (\neg x_1 \lor \neg x_3) \land (x_3 \lor x_4 \lor \neg x_1 \lor \neg x_2)\). The gray facts are exogenous.

We now show that Shapley \((D, q, f) \neq 0 \) if and only if \( \varphi \) is satisfiable. In fact, we show that Shapley \((D, q, f) > 0 \) if and only if \( \varphi \) is satisfiable, since \( T \) appears only as a positive atom in \( q \) and \( f \) can only be positively relevant to \( q \). Observe that \( D_x \models |Q \) since every (exogenous) fact \( S(i, j, a, a) \), along with the exogenous facts \( R(a), T(a) \) satisfies \( q \). In the example of Figure 5.2, this is due to the existence of the facts \( S(1, 2, a, a), R(a) \) and \( T(a) \) and the absence of the facts \( R(1) \) and \( R(2) \) in \( D_x \). We assume here that every \( \varphi \) contains at least one clause of the form \((x_i \lor x_j)\) (otherwise, there is no fact \( S(i, j, a, a) \) in \( D \)). We can assume that since the satisfiability problem is trivial for \((2^+, 2^-, 4^+)\) formulas that do not contain at least one clause of the form \((x_i \lor x_j)\) (as all such formulas are satisfied by the assignment \( z \) where \( z(x) = 0 \) for every variable \( x \)). Hence, the \((2^+, 2^-, 4^+)\)-SAT problem remains hard under this assumption.

Assume that \( \varphi \) is satisfiable by an assignment \( z \), and consider the set \( E = \{ R(i) \mid z(x_i) = 1 \} \). We claim that \((D_x \cup E) \models q \). For every exogenous fact \( S(i, j, a, a) \), at least one of facts \( R(i) \) or \( R(j) \) is in \( E \), since the clause \((x_i \lor x_j)\) is satisfied; hence, \( S(i, j, a, a), T(a) \) and \( R(a) \) cannot jointly satisfy \( q \). Moreover, for every \( S(b, b, i, j) \), at most one
of the facts \( R(i) \) and \( R(j) \) are in \( E \), since the clause \((\neg x_i \lor \neg x_j)\) is satisfied as well. Finally, for every \( S(i, j, k, l) \), it holds that if both \( R(k) \) and \( R(l) \) are in \( E \), then at least one of \( R(i) \) and \( R(j) \) is in \( E \) as well, since the clause \((x_i \lor x_j \lor \neg x_k \lor \neg x_l)\) is satisfied.

On the other hand, it holds that \((D_x \cup E \cup f) \models q\), since the facts \( S(d, d, c, c), R(c) \) and \( T(c) \) are in \((D_x \cup E \cup \{f\})\) while the fact \( R(d) \) is not. Therefore, we conclude that \( f \) is relevant to \( q_{\text{RST-R}} \).

Now, assume that \( \varphi \) is not satisfiable. Let \( E \subseteq (D_n \setminus \{f\}) \). Recall that the only endogenous facts in \( D_n \setminus \{f\} \) are the facts \( R(i) \) for \( i \in \{1, \ldots, n\} \). We now define the assignment \( z \) such that \( z(x_i) = 1 \) if and only if \( R(i) \in E \). Since \( z \) is not a satisfying assignment, at least one clause \( c \) in \( \varphi \) is not satisfied. If \( c \) is of the form \((x_i \lor x_j)\), then none of \( R(i), R(j) \) is in \( E \), in which case the exogenous facts \( S(i, j, a, a), R(a) \) and \( T(a) \) satisfy \( q \). If \( c \) is of the form \((\neg x_i \lor \neg x_j)\), then both \( R(i) \) and \( R(j) \) are in \( E \), and they satisfy \( q \) jointly with the facts \( S(b, b, 1, j) \) and \( T(i) \) (as the fact \( R(b) \) is not in \( D \)). Otherwise, \( c \) is of the form \((x_i \lor x_j \lor \neg x_k \lor \neg x_l)\), in which case none of \( R(i) \) or \( R(j) \) is in \( E \), while both \( R(k), R(l) \) are in \( E \); hence, the facts \( S(i, j, k, l), T(k), R(k) \) and \( R(l) \) jointly satisfy \( q \). In all of these cases, we conclude that \((D_x \cup E) \models q\); thus, adding \( f \) in a permutation after the facts of \( E \) would not affect the query result, and \( f \) is not relevant to \( q_{\text{RST-R}} \). This concludes our proof of Proposition 5.6.

**Corollary 5.7.** Given a database \( D \) and a fact \( f \in T^D \), the decision problem of whether \( \text{Shapley}(D, q_{\text{RST-R}}, f) = 0 \) is \text{coNP-complete}.

The existence of a multiplicative FPRAS for \( \text{Shapley}(D, q_{\text{RST-R}}, f) \) would imply the existence of a randomized algorithm that, for every \( \delta \in (0, 1) \), returns zero if \( \text{Shapley}(D, q_{\text{RST-R}}, f) = 0 \) and a value \( v \neq 0 \) otherwise, with probability at least \( 1 - \delta \). Hence, we could obtain a randomized algorithm for deciding if \( \text{Shapley}(D, q_{\text{RST-R}}, f) = 0 \) from a multiplicative FPRAS for \( \text{Shapley}(D, q_{\text{RST-R}}, f) \), in contradiction to the result of Corollary 5.7.

### 5.3 More on the Complexity of Relevance

In this section, we further investigate the relevance problem for \( \text{CQ}^- \). Observe that in Proposition 5.6 (and Corollary 5.7) we consider a fact that belongs to a polarity-consistent relation; however, the query is not polarity-consistent as it contains a relation that appears both in a positive and a negative atom of \( q \) (i.e., the relation \( R \)). What about the cases where every relation of \( q \) is polarity-consistent? We show that the problem of deciding whether a fact is relevant (and thereof the problem of deciding whether the Shapley value is zero) can always be solved in polynomial time for polarity-consistent queries. Hence, we conclude that having a non polarity-consistent relation is a necessary condition for hardness of both problems.

**Proposition 5.8.** Let \( q \) be a polarity-consistent \( \text{CQ}^- \). Given a database \( D \) and a fact \( f \), the following decision problems are solvable in polynomial time:
• Is \( f \) relevant to \( q \)?

• Is \( \text{Shapley}(D, q, f) = 0 \)?

Since \( q \) is polarity-consistent, the relevance to \( q \) is the same as the Shapley value being nonzero. Hence, to prove the proposition, we introduce the algorithm \( \text{IsPosRelevant} \) (depicted as Algorithm 5.1) for deciding whether a fact \( f \) is positively relevant to \( q \). The algorithm \( \text{IsNegRelevant} \) for deciding whether a fact is negatively relevant is very similar (depicted as Algorithm 5.2). In the algorithms, we denote by \( \text{Dom}(D) \) the set of constants used in the facts of \( D \). Moreover, we denote by \( \text{Neg}_q(D_n) \) the set of facts in \( D_n \) that appear in relations associated with negative atoms of \( q \).

We now prove the correctness of \( \text{IsPosRelevant} \) and \( \text{IsNegRelevant} \) for deciding whether a fact is positively or negatively relevant to \( q \). We start with \( \text{IsPosRelevant} \) and prove the following.

**Lemma 5.3.1.** Let \( q \) be a polarity-consistent \( CQ \). Then, \( \text{IsPosRelevant}(D, q, f) \) returns true, given \( D \) and \( f \), if and only if \( f \) is positively relevant to \( q \).

**Proof.** Assume that \( f \) is positively relevant to \( q \). Thus, there exists \( E \subseteq D_n \) such that \((D_x \cup E) \not\models q \) while \((D_x \cup E \cup \{f\}) \models q \). Hence, there is a homomorphism \( h \) from the variables of \( q \) to the constants of \( D \) such that every positive atom and none of the negative atoms of \( q \) is mapped to a fact of \( D_x \cup E \cup \{f\} \). We claim that the algorithm will return true in the iteration of the for loop when \( h \) is selected. By the definition of \( h \) we have that \( P \subseteq (E \cup \{f\}) \), while for every \( f' \in N \) it holds that \( f' \notin (E \cup \{f\}) \). Moreover, since \((D_x \cup E) \not\models q \), the homomorphism \( h \) maps a positive atom of \( q \) to \( f \); hence \( f \in P \). Since \( q \) is polarity consistent, by adding a set of facts corresponding to negative atoms of \( q \) we cannot change the query result from false to true. Therefore, the fact that \((D_x \cup E) \not\models q \) implies that \((D_x \cup E \cup (\text{Neg}_q(D_n) \setminus N)) \not\models q \). Since no fact of \( N \) appears in \( E \), the set \( D_x \cup (P \setminus \{f\}) \cup (\text{Neg}_q(D_n) \setminus N) \) can be obtained from \( D_x \cup E \cup (\text{Neg}_q(D_n) \setminus N) \) by removing a set of facts corresponding to positive atoms of \( q \), and, again, since \( q \) is polarity consistent, we conclude that \((D_x \cup (P \setminus \{f\}) \cup (\text{Neg}_q(D_n) \setminus N)) \not\models q \).

Next, assume that the algorithm returns true. Thus, there exists a mapping \( h \) from the variables of \( q \) to the constants of \( D \) such that \((D_x \cup (P \setminus \{f\}) \cup (\text{Neg}_q(D_n) \setminus N)) \not\models q \). Let \( E = ((P \setminus \{f\}) \cup (\text{Neg}_q(D_n) \setminus N)) \). We will now show that \((D_x \cup E \cup \{f\}) \models q \) and since \((D_x \cup E) \not\models q \), this will conclude our proof. By the definition of \( N \) and since \( h \) does not map any negative atom of \( q \) to a fact in \( D_x \), we have that \( h \) does not map any negative atom of \( q \) to a fact of \( D_x \cup (\text{Neg}_q(D_n) \setminus N) \). Moreover, since \( h \) maps every positive atom of \( q \) to a fact in \( D \), we have that every positive atom of \( q \) is mapped by \( h \) to a fact in \( D_x \cup P \cup \{f\} \). Therefore, we conclude that \( h \) is a homomorphism mapping every positive atom and none of the negative atoms of \( q \) to facts of \( D_x \cup (P \setminus \{f\}) \cup (\text{Neg}_q(D_n) \setminus N) \cup \{f\} \) and we have that \((D_x \cup E \cup \{f\}) \models q \).
Algorithm 5.1 IsPosRelevant($D, q, f$)

for $h: \text{Vars}(q) \rightarrow \text{Dom}(D)$ do

    if $h$ maps an atom $\alpha \in \text{Neg}(q)$ to some $f' \in D_x$ then
        continue
    if $h$ maps an atom $\alpha \in \text{Pos}(q)$ to some $f' \notin D$ then
        continue
    $P = \{f' \in D_n \mid h$ maps an atom $\alpha \in \text{Pos}(q)$ to $f'\}$
    $N = \{f' \in D_n \mid h$ maps an atom $\alpha \in \text{Neg}(q)$ to $f'\}$
    if $f \notin P$ then
        continue
    if $(D_x \cup (P \setminus \{f\}) \cup (\text{Neg}_q(D_n) \setminus N)) \not|= q$ then
        return false
    return true

Algorithm 5.2 IsNegRelevant($D, q, f$)

for $h: \text{Vars}(q) \rightarrow \text{Dom}(D)$ do

    if $h$ maps an atom $\alpha \in \text{Neg}(q)$ to some $f' \in D_x$ then
        continue
    if $h$ maps an atom $\alpha \in \text{Pos}(q)$ to some $f' \notin D$ then
        continue
    $P = \{f' \in D_n \mid h$ maps an atom $\alpha \in \text{Pos}(q)$ to $f'\}$
    $N = \{f' \in D_n \mid h$ maps an atom $\alpha \in \text{Neg}(q)$ to $f'\}$
    if $f \in P$ then
        continue
    if $(D_x \cup P \cup (\text{Neg}_q(D_n) \setminus N) \cup \{f\}) \not|= q$ then
        return false
    return true

We now move on the the algorithm IsNegRelevant and prove its correctness. The algorithm is very similar to IsPosRelevant, except for the fact that we now look for a homomorphism that does not maps any positive atom of $q$ to $f$, and in the last test, we check whether the query is not satisfied by $D_x \cup P \cup (\text{Neg}_q(D_n) \setminus N) \cup \{f\}$.

Lemma 5.3.2. Let $q$ be a polarity-consistent CQ$^\neg$. Then, IsNegRelevant($D, q, f$) returns true, given $D$ and $f$, if and only if $f$ is negatively relevant to $q$.

Proof. Assume that $f$ is negatively relevant to $q$. Thus, there exists $E \subseteq D_n$ such that $(D_x \cup E) |= q$ while $(D_x \cup E \cup \{f\}) \not|= q$. Hence, there is a homomorphism $h$ from the variables of $q$ to the constants of $D$ such that every positive atom and none of the negative atoms of $q$ is mapped to a fact of $D_x \cup E$, while $h$ maps $f$ to a negative atom of $q$. We claim that the algorithm will return true in the iteration of the for loop when $h$ is selected. By the definition of $h$ we have that $P \subseteq E$, while for every $f' \in N$ it holds that $f' \notin E$. Moreover, since $(D_x \cup E \cup \{f\}) \not|= q$, the homomorphism $h$ does not map any positive atom of $q$ to $f$; hence $f \notin P$. Since $q$ is polarity consistent, by adding a set of facts corresponding to negative atoms of $q$ we cannot change the
query result from false to true. Therefore, the fact that \((D_x \cup E \cup \{f\}) \not\models q\) implies that \((D_x \cup E \cup \{f\} \cup (\text{Neg}_q(D_n) \setminus N)) \not\models q\). Since no fact of \(N\) appears in \(E\), the set \(D_x \cup P \cup \{f\} \cup (\text{Neg}_q(D_n) \setminus N)\) can be obtained from \(D_x \cup E \cup \{f\} \cup (\text{Neg}_q(D_n) \setminus N)\) by removing a set of facts corresponding to positive atoms of \(q\), and, again, since \(q\) is polarity consistent, we conclude that \((D_x \cup P \cup \{f\} \cup (\text{Neg}_q(D_n) \setminus N)) \not\models q\).

Next, assume that the algorithm returns true. Thus, there exists a mapping \(h\) from the variables of \(q\) to the constants of \(D\) such that \((D_x \cup P \cup (\text{Neg}_q(D_n) \setminus N) \cup \{f\}) \not\models q\). Let \(E = (P \cup (\text{Neg}_q(D_n) \setminus N))\). We will now show that \((D_x \cup E) \models q\) and since \((D_x \cup E \cup \{f\}) \not\models q\), this will conclude our proof. By the definition of \(N\) and since \(h\) does not map any negative atom of \(q\) to a fact in \(D_x\), we have that \(h\) does not map any negative atom of \(q\) to a fact of \(D_x \cup (\text{Neg}_q(D_n) \setminus N)\). Moreover, since \(h\) maps every positive atom of \(q\) to a fact in \(D\), we have that every positive atom of \(q\) is mapped by \(h\) to a fact in \(D_x \cup P\). Therefore, we conclude that \(h\) is a homomorphism mapping every positive atom and none of the negative atoms of \(q\) to facts of \(D_x \cup P \cup (\text{Neg}_q(D_n) \setminus N)\) and we have that \((D_x \cup E) \models q\).

Finally, we state that both algorithms terminate in polynomial time since the number of mappings from the variables of \(q\) to the constants of \(D\) is polynomial in the size of \(D\) when considering data complexity.

Next, we show that relevance can be determined efficiently not only for polarity-consistent queries. We denote as \(\text{PC}_1\) the class of all \(\text{CQ}\) that can be turned into a polarity-consistent query by removing a single atom. For example, the query \(q() : R(x, y), R(y, z), \neg R(z, x), \neg S(y, x)\) belongs to the class \(\text{PC}_1\), since if we remove the atom \(\neg R(z, x)\) from \(q\), the resulting query is polarity-consistent. We now prove the following.

**Proposition 5.9.** Let \(q\) be a \(\text{CQ}\) in \(\text{PC}_1\). Then, the problem of deciding whether \(f\) is relevant to \(q\) is solvable in polynomial time, given \(D\) and \(f\).

In the following proof we are going to use a reduction to \(\text{Horn-SAT}\), which is the problem of deciding whether a given set of propositional Horn clauses is satisfiable or not. A Horn clause is a clause with at most one positive literal, and any number of negative literals. A Horn formula is a propositional formula formed by conjunction of Horn clauses. The problem of Horn satisfiability is known to be solvable in linear time [DG84]. We now show a reduction from the problem of deciding relevance w.r.t. \(q\) in \(\text{PC}_1\) to the Horn-SAT problem. Let \(q\) be such query. We denote as \(R\) the single non polarity-consistent relation symbol that occurs in \(q\), and assume (without the loss of generality) that \(R\) occurs positively only once in \(q\). Given a database \(D\), we denote as \(f_1, \ldots, f_n\) the endogenous facts in \(R^D\). Let \(h_1, \ldots, h_m\) be all possible mappings \(h_i : \text{Vars}(q) \rightarrow \text{Dom}(D)\) (recall that \(m\) is bounded by a polynomial in the size of \(D\), as we consider data complexity). Every \(h_i\) potentially maps the atoms of \(q\), particularly the \(R\)-atoms, to a set of facts in \(D\). For every \(h_i\), if \(h_i\) maps at least one atom of \(q\) to an endogenous fact in \(R^D\), we construct a clause \(\varphi_i = \neg x_j \lor x_{i_1} \lor x_{i_2} \lor \cdots \lor x_{i_t}\), where
\(f_j\) is the fact obtained from the positive \(R\)-atom by the mapping \(h_i\), and \(f_{t_1}, \ldots, f_{t_r}\) are the facts obtained from the negative \(R\)-atoms by \(h_i\). If \(h_i\) maps an \(R\)-atom to a fact which is not in \(R^D\) (or to an exogenous fact), the corresponding literal will not appear in \(\varphi_i\). Observe that every clause \(\varphi_i\) consists of at most one negative literal, so every \(\varphi_i\) is a Horn clause. Finally, we denote \(\varphi\) to be the conjunction of all the clauses \(\varphi_i\), which is a Horn formula.

Next, we show how can we determine whether a certain endogenous fact \(f\) is positively (resp, negatively) relevant to \(q\). We do that by extending Algorithm 5.1 (resp, 5.2), so it will apply for PC1 queries as well. First, we go over all the mappings \(h_1, \ldots, h_m\), and consider only the ones where all of the positive atoms are mapped to a fact in \(D\), none of the negative atoms of \(q\) are mapped to a fact in \(D_x\), and a positive (resp, negative) atom of \(q\) is mapped to \(f\). Let \(h\) be such homomorphism. In the terms of Algorithms 5.1 and 5.2, we first check if \(P \cap N = \emptyset\), and if not, we continue to the next mapping. In the next step, we check if the violation condition holds: \((D_x \cup (P \setminus \{f\}) \cup (\text{Neg}(D_n) \setminus N)) \not\models q\) (resp, \((D_x \cup P \cup (\text{Neg}(D_n) \setminus N) \cup \{f\}) \models q\)), where the facts in \(R^D\) are not considered as part of the set \(\text{Neg}(D_n)\). If this condition holds, we determine that \(f\) is relevant. Otherwise, we assign the value True to determine that \(f\) is relevant. Based on the same arguments from the proof of Lemma 5.3.1 (resp, Lemma 5.3.2). As for the case where the violation condition does not hold at some point, we prove the following.

**Lemma 5.3.3.** Let \(f \in D_n\), and assume that the violation condition does not hold for every mapping \(h\). Then, there exists a mapping \(h\) from \(q\) to \(\text{Dom}(D)\) such that \(\varphi_h\) is satisfiable, if and only if \(f\) is relevant to \(q\).

**Proof.** Let \(h\) be such a homomorphism, and assume that \(\varphi_h\) is satisfiable by the assignment \(z\). Consider the following set of facts: \(R^D_z := \{f_i : z(x_i) = 1\}\). If we check whether \(f\) is positively relevant, then \((D_x \cup P \cup (\text{Neg}(D_n) \setminus N) \cup R^D_z) \models q\). That is since \((D_x \cup P \cup (\text{Neg}(D_n) \setminus N)) \models q\), and \(N \cap R^D_z = \emptyset\) due to the construction of \(\varphi_h\) where every variable corresponding to a fact in \(N\) gets the value False. Moreover, \((D_x \cup (P \setminus \{f\}) \cup (\text{Neg}(D_n) \setminus N) \cup R^D_z) \not\models q\). This holds true since every clause \(\varphi_i\) is satisfied in \(\varphi_h\) by the assignment \(z\); therefore, none of the mappings \(h_1, \ldots, h_m\) is a homomorphism from \(q\) to \(D_x \cup (P \setminus \{f\}) \cup (\text{Neg}(D_n) \setminus N) \cup R^D_z\). That is due to the construction of every \(\varphi_i\), where the clause is satisfied only if there is either a negative
R-atom that is mapped to a fact, or a positive R-atom that is not mapped to a fact. If we check whether \( f \) is negatively relevant, then \((D_x \cup P \cup (\text{Neg}(D_n) \setminus N) \cup R^D_x \cup \{ f \}) \not\models q\), and \((D_x \cup P \cup (\text{Neg}(D_n) \setminus N) \cup R^D_x \cup \{ f \}) \not\models q\). In this case, the fact that \( \varphi_h \) is satisfiable implies that none of the mappings \( h_1, \ldots, h_m \) will remain a homomorphism after adding \( f \) to the database. \( h \) is a homomorphism before adding \( f \), and by that we conclude the correctness of the claim.

Next, we assume that \( f \) is positively (resp, negatively) relevant. Let \( E \) be a subset of endogenous facts such that \((D_x \cup E) \not\models q\) and \((D_x \cup E \cup \{ f \}) \models q\) (resp, \((D_x \cup E) \models q\) and \((D_x \cup E \cup \{ f \}) \not\models q\), and let \( h \) be a homomorphism from \( q \) to \( D_x \cup E \cup \{ f \} \) (resp, to \( D_x \cup E \)). Clearly, \((P \setminus \{ f \}) \subseteq E\), and \( N \cap E = \emptyset \). We claim that \( \varphi_h \) is satisfiable. Since the violation condition does not hold, we have that \((D_x \cup (P \setminus \{ f \}) \cup (\text{Neg}(D_n) \setminus N)) \not\models q\) (resp, \((D_x \cup P \cup (\text{Neg}(D_n) \setminus N) \cup \{ f \}) \models q\). By adding the set of facts \( \text{Neg}(D_n) \) to \( D_x \cup (P \setminus \{ f \}) \) we cannot change the query result from false to true. Assume by the way of contradiction that \( \varphi_h \) is not satisfiable. Hence, for every subset \( E' \subseteq (E \cap R^D) \), there is a homomorphism from the \( R \)-atoms of \( q \) to \( D_x \cup E' \). This is a contradiction to the fact that \((D_x \cup E) \not\models q\), (resp, \((D_x \cup E \cup \{ f \}) \not\models q\), as the only possible way to violate a homomorphism from \( q \) to \( D_x \cup (P \setminus \{ f \}) \cup (\text{Neg}(D_n) \setminus N) \) (resp, \( D_x \cup P \cup (\text{Neg}(D_n) \setminus N) \cup \{ f \} \)) is by mapping a negative \( R \)-atom to a fact in \( R^D \).

Thus, there is a subset of endogenous facts \( E' \subseteq (R^D \cap E) \) such that every mapping \( h_1, \ldots, h_m \) that maps every positive atom of \( q \) to a fact in \( D_x \cup E \), maps a negated \( R \)-atom to a fact in \( E' \) as well. If \( f \) is positively relevant, the following assignment \( z \) for which \( z(x_i) = 1 \) if \( f_i \in (E' \setminus \{ f \}) \) and \( z(x_i) = 0 \) otherwise, will satisfy \( \varphi_h \). This is due to the fact that every clause \( \varphi_i \) is satisfied if and only if \( h_i \) is not a homomorphism. If \( f \) is negatively relevant, again, there is a subset of endogenous facts \( E' \subseteq (R^D \cap E) \) such that there is no homomorphism from the \( R \)-atoms of \( q \) to \( D_x \cup E' \cup \{ f \} \). The following assignment \( z \) for which \( z(x_i) = 1 \) if \( f_i \in (E' \cup \{ f \}) \) and \( z(x_i) = 0 \) otherwise, will again satisfy \( \varphi_h \).

From this Lemma we conclude that the extended algorithm will return true if and only if \( f \) is relevant. Overall, the relevance problem can be determine in polynomial time for every endogenous fact w.r.t. \( q \), based on the reduction to the Horn-SAT problem. The proof for the case where the single non polarity-consistent relation of \( q \) occurs negatively only once is very similar. In this case, the reduction is to the Dual-Horn-SAT problem where every clause has at most one negative literal, and any number of positive literals, which can be solved efficiently as well.

We conclude with a non formal conjecture: the relevance problem is hard for most CQ\(^{-}\)'s which are neither polarity consistent nor in PC\(_1\). This assumption is based on the connection between the relevance problem and SAT that arises from the proves of Propositions 5.6 and 5.9. While the Horn-SAT problem is solvable in polynomial time, the majority of SAT variations are known to be hard [Sch78]. We believe that the structure of non polarity-consistent CQ\(^{-}\)'s (which are not in PC\(_1\)) allows to reduce
the relevance problem to hard variations of SAT. Nonetheless, the full classification of the relevance problem for CQ$^-$ remains an open question for future research.

### 5.4 Results Summary

In Figure 5.3 we summarize our results regarding the relevance problem and the existence of a multiplicative approximation for the Shapley value. We can say that a multiplicative approximation definitely exists only for queries that satisfy the gap property (i.e., for CQs without negation), whereas we do not know if a multiplicative approximation exists for CQ$^-$’s that violate the gap property (i.e., for every query with a negated atom (5.2)). Moreover, there is a class of CQ$^-$ queries for which the relevance problem is hard that includes $q^{RST-R}$ (5.6); a multiplicative approximation to the Shapley value cannot exist for those queries. Next, we state that if a CQ$^-$ is polarity-consistent, then the relevance problem can be decided in polynomial time (5.8). However, polarity-consistency is not a necessary condition for deciding relevance efficiently, as evident by the class PC$_1$ (5.9).

### 5.5 An Insight into UCQs with Negation

Interestingly, while the relevance problem can be solved in polynomial time for any polarity-consistent CQ, this is no longer the case when considering a union of polarity-consistent CQs. Specifically, we show that the relevance to the UCQ$^-$ $q^{SAT}() := q_1() \lor \ldots \lor q_n()$...
Given a database \( D \) that

\[ q_2() \lor q_3() \lor q_4() \]

is NP-complete, where:

\[ q_1() : C(x_1, x_2, x_3, v_1, v_2, v_3), T(x_1, v_1), T(x_2, v_2), T(x_3, v_3) \]

\[ q_2() : V(x), \neg T(x, 1), \neg T(x, 0) \]

\[ q_3() : T(x, 1), T(x, 0) \]

\[ q_4() : R(0) \]

In particular, we show that it is hard to decide whether the fact \( R(0) \) is relevant to \( \text{qsat} \).

**Proposition 5.10.** Given a database \( D \) and a fact \( f \in R^D \), deciding whether \( f \) is relevant to \( \text{qsat} \) is NP-complete.

**Proof.** We construct a reduction from the satisfiability problem for 3CNF formulas. The input to the satisfiability problem is a formula \( \varphi = (c_1 \land \cdots \land c_m) \) over the variables \( x_1, \ldots, x_n \), where each \( c_i \) is a clause of the form \( (l_1 \lor l_2 \lor l_3) \), and each \( l_j \) is either a positive literal \( x_k \) or a negative literal \( \neg x_k \) for some \( k \in \{1, \ldots, n\} \). Given such an input, we build an input database \( D \) to our problem as follows. For every variable \( x_i \) we add an exogenous fact \( V(i) \), and two endogenous facts \( T(i, 1) \) and \( T(i, 0) \). In addition, for every clause \( (l_i \lor l_j \lor l_k) \) where \( l_t = x_t \) or \( l_t = \neg x_t \) for each \( t \in \{i, j, k\} \), we add an exogenous fact \( C(i, j, k, v_i, v_j, v_k) \), such that \( v_t = 1 \) if \( l_t = \neg x_t \) and \( v_t = 0 \) if \( l_t = x_t \). Finally, we add the endogenous fact \( R(0) \) which we denote as \( f \). We claim that \( f \) is relevant to \( q \) if and only if \( \varphi \) is satisfiable.

Observe that \( E \models q \) for every \( E \subseteq D \) such that \( f \in E \), since \( f \) satisfies the query \( q_4 \) by itself. Hence, \( f \) is relevant (and, more precisely, positively relevant) if and only if there exist \( E \subseteq D_n \) such that \( (D_x \cup E) \not\models q \). Now, assume that \( \varphi \) is satisfiable by the assignment \( z \). We will show that \( f \) is relevant to \( q \). Consider the subset \( E \subseteq D_n \) that contains every fact \( T(i, v_i) \) such that \( z(x_i) = v_i \). Since \( z \) is a truth assignment, it assigns a single value to each variable; hence, it is straightforward that \( (D_x \cup E) \not\models q_2 \) and \( (D_x \cup E) \not\models q_3 \). Regarding the query \( q_1 \), since \( z \) is a satisfying assignment, for every clause in \( \varphi \) there is at least one literal \( l_i \) such that \( z(x_i) = 0 \) if \( l_i = \neg x_i \) and \( z(x_i) = 1 \) if \( l_i = x_i \). Therefore, the fact \( T(x_i, z(x_i)) \) does not appear in \( E \), and we have that \( (D_x \cup E) \not\models q \). We conclude that \( (D_x \cup E) \not\models q \) while \( (D_x \cup E \cup \{f\}) \models q \) and that concludes our proof of the first direction.

As for the other direction, given a subset \( E \subseteq D_n \) such that \( (D_x \cup E) \not\models q \) while \( (D_x \cup E \cup \{f\}) \models q \), we define an assignment \( z \) such that \( z(x_i) = 1 \) if \( T(i, 1) \in E \) and \( z(x_i) = 0 \) if \( T(i, 0) \in E \). Since \( (D_x \cup E) \not\models q \), it cannot be the case that \( E \) contains two facts \( T(i, 1) \) and \( T(i, 0) \) (or, otherwise, \( (D_x \cup E) \models q_3 \)) and it cannot be the case that none of \( T(i, 1) \) and \( T(i, 0) \) belongs to \( E \) for some \( x_i \) (as otherwise, \( (D_x \cup E) \models q_2 \)). Hence, \( z \) is a truth assignment. It is only left to show that \( z \) is a satisfying assignment. Assume, by way of contradiction, the a clause \( (l_i, l_j, l_k) \) is not satisfied. In this case, \( z(x_i) = 0 \) if \( l_t = x_t \) and \( z(x_i) = 1 \) if \( l_t = \neg x_t \) for each \( t \in \{i, j, k\} \). Since \( E \) contains a fact
Figure 5.4: Relevance and multiplicative approximation - UCQs with negation

Since the relation $R$ is polarity-consistent and only occurs as a positive atom in $q_{\text{SAT}}$, we again conclude the following.

**Corollary 5.11.** Given a database $D$ and a fact $f \in R^D$, the decision problem of whether Shapley($D, q_{\text{SAT}}, f$) = 0 is coNP-complete.

Note that while every individual CQ$^\neg$ in the query $q_{\text{SAT}}$ is polarity-consistent, the whole query is not, as the relation $T$ appears as a positive atom in $q_1$ and $q_3$ and as a negative atom in $q_2$. If a UCQ$^\neg$ $q$ is such that the whole query is polarity-consistent, then the relevance problem is solvable in polynomial time. This is due to the fact that a fact $f$ is relevant to such a UCQ $q$ if and only if it is relevant to at least one of the CQs in $q$. Hence, we can use our algorithms $\text{IsPosRelevant}$ and $\text{IsNegRelevant}$ for every individual CQ$^\neg$ in $q$ to decide whether a fact $f$ is relevant to $q$. In this case, we cannot preclude the existence of a multiplicative approximation. In Figure 5.4 we illustrate our results regarding UCQ$^\neg$'s.
Chapter 6

Conclusions

We have investigated the complexity of computing the Shapley value for CQs and UCQs with negation. In particular, we have generalized a dichotomy by Livshits et al. [LBKS20] to classify the class of all CQs with negation and without self-joins. We further generalized this dichotomy to account for exogenous relations that are allowed to contain only exogenous facts. We have also studied the complexity of approximating the Shapley value in a multiplicative manner. The presence of negation makes this approximation fundamentally harder than the monotonic case, since the gap property (that unifies the additive and multiplicative FPRAS task) no longer holds. We have shown the hardness of approximation by making the connection to the problem of deciding relevance to a query, and by establishing hardness results for that problem. On the other hand, we have shown that relevance can be determined efficiently for a substantial class of queries, for which we cannot preclude the existence of a multiplicative FPRAS.

This work leaves open several immediate directions for future research. In particular, we do not yet have a dichotomy for the class of CQs with self-joins (with or without negation). We know from past research that self-joins may cast dichotomies considerably more challenging to prove [DS12]. However, we do state that self-joins preserve hardness for a certain class of queries, and suggest an approach for computing the Shapley value for a query that consists of a single self-join. In addition, we do not have a full understanding of the constraint of endogenous relations as an analogue of exogenous relations; nevertheless, we showed that this problem tightly relates to the problem of uniform reliability for conjunctive queries. Finally, we leave open some fundamental questions about the complexity bounds of Shapley approximations: Is there a multiplicative FPRAS in the absence of the gap property? Are there cases where the relevance problem is tractable but a multiplicative approximation is computationally hard (beyond some ratio)? And lastly, what is the exact criterion that determines for which CQs (or UCQs) with negation the relevance problem is hard?
Bibliography


עם שלילה, ובפרקה הרביעית והחמישיית, נודע בהשלכות מושכות והשפעה של טבלאות אקסוגניות על אבחון סיבוכיות. נדון בחישובה מקורב של עדות של יובלוב, ובפרקים החמישי והשישי, נבורק תכשיטי יזון סיכום התמצאות דינו ב BITS ואפליקציות למשתתפי המחקר.

iv
עבוריВидеושידлярائهمות פרוג (ללא כלים), ויתק לחשימאות קירוב ובפל סעisher שמקורות לקירוב הביאו לשליש סתמולאש קירוב טסב ventas שמקורות לקירוב הביאו לשליש סתמולאש קירוב טסב ventas שמקורות לקירוב הביאו לשליש סתמולאש קירוב טסב ventas שמקורות לקירוב הביאו לשליש סתמולאש קירוב טסב ventas שמקורות לקירוב הביאו לשליש סתמולאש קירוב טסב ventas שמקורות לקירוב הביאו לשליש סתמולאש קירוב טסב ventas שמקורות לקירוב הביאו לשליש סתמולאש קירוב טסב ventas שמקורות לקירוב הביאו לשליש סתמולאש קירוב טסב ventas שמקורות לקירוב הביאו לשליש סתמולאש קירוב טסב ventas שמקורות לקירוב הביאו לשליש סתמולאש קירוב טסב ventas שמקורות לקירוב הביאו לשליש סתמולאש קירוב טסב ventas שמקורות לקירוב הביאו לשליש סתמולאש קירוב טסב ventas שמקורות לקירוב הביאו לשליש סתמולאש קירוב טסב ventas שמקורות לקירוב הביאו לשליש סתמולאש קירוב טסב ventas שמקורות לקירוב הביאו לשליש סתמולאש קירוב טסב ventas שמקורות לקירוב הביאו לשליש סתמולאש קירוב טסב ventas שמקורות לקירוב הביאו לשליש סתמולאש קירוב טסב ventas שמקורות לקירוב הביאו לשליש סתמולאש קירוב טסב ventas שמקורות לקירוב הביאו לשליש סתמולאש קירוב טסב ventas שמקורות לקירוב הביאו L
לרשומות המקור התחקלאים שונים בטבלתuchaחקלאיםעשוייםלהיותערכישפלי.םינוןביטכןלשמםהערכישה cháחקלאיםאםםיתכןשהם će البعיונם על שאלתעהאשתהשאלה,אשרנוקבעתעלפיאליותיוצרוהםמייצאים,האםםיתכןשההם��ובים.לשם מסר מיום בוו (0)شفח תחתון רשומה שלא כל 함ך תחקלאה אשתהשאלה או (0)شفח תחתון רשומה שלא כל 함ך תחקלאה אשתהשאלה (בבלבד),ועבר ישלון רשומה המיידית יוצר ימים מסים דגל יקול בלא מפרס רשומה לשדרה ומפרס רשומה לשדרה,ולשם מסר מיום בוו (0)شفח תחתון רשומה שלא כל 함ך תחקלאה אשתהשאלה או (0)شفח תחתון רשומה שלא כל 함ך תחקלאה אשתהשאלה (בבלבד).

ליבשיץ ואחרים חקרו את סיבוכיות החישוב של ערך השפליעבורהשאילתותצירוף.라도 עבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה,QueryParam הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק داخل זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן חלק בתוך זרם תחקלאים או מספר הפרסים בוובן, היא אשתהשהשאלה.השחקנים במודל של ערך השפליעבורה רשומותיהן Часть внутренних и внешних структур.
תקציר

A short summary of the contents of the document is provided here.

The summary includes:

1. A brief overview of the main research question and methodology.
2. A description of the data used and the statistical techniques applied.
3. An analysis of the results and their implications.

The summary is written in hebrew and provides a concise summary of the document.

The full text of the document is available in the provided PDF file.
חשב על אחוזי שרするために ב워
שאלות צירוף עם שלילה
חיבור על מחקר
לשם مليולי חלקי של הדרישות לבקלת התואר
מוניטרות למdıים במדעי המחשב

אלון רשף

הﳓ לחוגל הטכנולוגי — מרכז טכנולוגיה לישראל
תשי”א חhiba ספטמבר 2020
חישוב ערך שליל של רשומות עבורה
שאילחות צירוף עם שלילה

אלון רשף