Computing Range Consistent Answers to Conjunctive Queries with Aggregation

Aziz Amezian El Khalfioui

A dissertation submitted in fulfilment of the requirements of the degree of *Docteur en Sciences*

Jury

VÉRONIQUE BRUYÈRE University of Mons (Belgium)	President
FLORIS GEERTS University of Antwerp (Belgium)	Reviewer
MIIKA HANNULA University of Helsinki (Finland), and University of Tartu (Estonia)	Reviewer
PARASCHOS KOUTRIS University of Wisconsin-Madison (USA)	Reviewer
HADRIEN MÉLOT University of Mons (Belgium)	Reviewer
JEF WIJSEN University of Mons (Belgium)	Supervisor

Acknowledgements

I would like to start this thesis by thanking my supervisor, Jef Wijsen, to whom I am deeply grateful for his time and support. Working with you has been a real pleasure and, above all, a very enriching experience. Thank you for always being there when needed, always willing to lend a hand.

I would like to thank the jury members: Véronique Bruyère, Floris Geerts, Miika Hannula, Paraschos Koutris and Hadrien Mélot. Thank you for dedicating your time to this thesis.

I would also like to thank all the office co-workers with whom I have shared these last four years. Thank you for always welcoming me with a smile and making these years so enjoyable.

Last but not least, I would like to thank my family for always supporting me and always being there so that I could fully focus on my projects.

Abstract

This thesis focuses on the problem of answering numerical queries issued against databases that may violate primary key constraints. A database repair is defined as any maximal subset of the original database that satisfies all constraints. The range consistent answer to a numerical query is a pair [*glb*, *lub*], where *glb* and *lub* denote the greatest lower bound and least upper bound, respectively, of the query results across all possible repairs. This work studies the computational complexity of determining range consistent answers to conjunctive queries with aggregation. In SQL, such queries follow the SELECT-FROM-WHERE-GROUP BY format, where the WHERE clause consists of a conjunction of equalities, and the SELECT clause may include aggregate functions such as MAX, MIN, SUM, AVG, or COUNT. The thesis investigates sufficient and necessary syntactic conditions under which range consistent answers can be computed via rewriting into first-order aggregate logic.

The thesis consolidates and moderately generalizes the results presented in three publications by Amezian El Khalfioui & Wijsen:

> Amezian El Khalfioui, A., & Wijsen, J. (2023). Consistent query answering for primary keys and conjunctive queries with counting. In *ICDT*, vol. 255 of *LIPIcs*, (pp. 23:1–23:19). Schloss Dagstuhl - Leibniz-Zentrum für Informatik.
>
> Amezian El Khalfioui, A., & Wijsen, J. (2024a). Computing range consistent answers to aggregation queries via rewriting. *Proc. ACM Manag. Data*, *2*(5), 218:1–218:19.
>
> Amezian El Khalfioui, A., & Wijsen, J. (2026). Computing consistent least upper bounds in aggregate logic. In *ICDT, to appear*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Contents

List of Notations

Notation	Page
$\mathsf{vars}(F)$	9
$\mathsf{Key}(F)$	9
$\mathsf{notKey}(F)$	9
$\models_{\mathsf{cqa}}$	10
$\mathsf{free}(q)$	10
$\mathsf{adom}(\mathbf{db})$	10
$\mathsf{dom}(\theta)$	10
$\theta\restriction_V$	10
$(\mathbf{db}, \theta) \models \varphi(\vec{x})$	10
$\mathsf{rset}(\mathbf{db})$	10
$\mathsf{body}(q)$	11
sjfBCQ	11
$\nexists\vec{x}\,[q]$	11
$\exists\vec{x}\,[q]$	11
$q_{[\vec{z}\mapsto\vec{c}]}$	11
$\mathcal{G}aifman(q)$	11
$q_1 \preccurlyeq_{\mathsf{g}} q_2$	11
$q_1 \sim_{\mathsf{g}} q_2$	11
$\mathcal{K}(q)$	11
$F^{+,q}$	12
$F \overset{q}{\rightsquigarrow} x$	12
$\mathbf{Emb}_q(\mathbf{db})$	13
$P_1 \leq_{\mathsf{FO}} P_2$	13
$P_1 \leq_{\mathsf{nr\text{-}datalog}} P_2$	13
$[\![g(\vec{a})]\!]^{\mathbf{db}}$	17
$\mathsf{BCQ}(g())$	19
$\mathsf{frozen}(q)$	31
$\mathsf{MCS}_q(M)$	53
$\mathsf{Ext}(\theta)$	55
$\forall\mathbf{Emb}_q(\mathbf{db})$	59
$i\#t$	86

CHAPTER 1

Introduction

1.1. Motivation

Typically, database systems have a set of integrity constraints (primary keys, foreign keys, etc.) that are intended to ensure the consistency of the information represented in their data. As long as these constraints are respected, we can safely assume that the information obtained from the database is correct.

However, there are situations where we may find ourselves with database instances that do not respect integrity constraints. A typical case where this may occur is when the database instance integrates data from multiple sources that individually satisfy the integrity constraints but collectively violate them.

Example 1.1.1. Consider the relation *Dealers* in the database instance of Fig. 1.1. The primary key for this relation is underlined: Each dealer can work in only one town. This constraint is clearly violated, as Smith works in both Boston and New York. Suppose this database instance integrates data from sources that gather data at a regional level. It is not possible to detect the inconsistency at the source level, since Boston and New York belong to different states. In each of these sources, Smith works in only one town. The inconsistency only becomes apparent once the data are integrated. ◁

We say that a database instance that does not respect its integrity constraints is *inconsistent*. Dealing with inconsistent database instances can be a major problem. Executing a query on such an instance may produce a result that cannot be trusted since the inconsistent data may influence the result. Multiple approaches exist to restore the consistency of the instance depending

Dealers	*Name*	*Town*
†	Smith	Boston
	Smith	New York
†	James	Boston

Stock	*Product*	*Town*	*Qty*
†	Tesla X	Boston	35
	Tesla X	Boston	40
†	Tesla Y	Boston	35
†	Tesla Y	New York	95
	Tesla Y	New York	96

Figure 1.1: Database instance $\mathbf{db}_{\mathsf{Stock}}$. Blocks are separated by dashed lines.

on the type of constraints considered. This process is typically called *repairing*. However, repairing may not be an immediate option, as it may require external information or making the database instance unavailable during the process. An alternative to repairing is Consistent Query Answering (CQA): rather than obtaining all the information from the inconsistent instance, we want to obtain only the part that does not rely on inconsistent data.

1.2. Consistent Query Answering

Consistent query answering (CQA) was introduced in (Arenas et al., 1999) as a principled approach to answering queries on databases that are inconsistent with respect to a given set of integrity constraints. The only integrity constraints we consider in the current work are primary keys. A *block* in a database instance is a $\subseteq$-maximal set of tuples of a same relation R that agree on the primary key of R. A *repair* of a database instance picks exactly one tuple from each block. Given a Boolean query q, $\mathsf{CERTAINTY}(q)$ is then defined as the decision problem that takes a database instance $\mathbf{db}$ as input, and asks whether q holds true in every repair of $\mathbf{db}$. This problem, while commonly studied for Boolean queries, can be readily extended to queries with free variables $\vec{x}$: a *consistent answer* to a query $q(\vec{x})$ is any sequence $\vec{c}$ of constants, of length $|\vec{x}|$, such that $q(\vec{c})$ holds true in every repair.

Example 1.2.1. The database $\mathbf{db}_{\mathsf{Stock}}$ of Fig. 1.1 records the quantity of products in stock in various towns (relation *Stock*) and the town of operation for each dealer (relation *Dealers*). The primary keys are underlined, and blocks are separated by dashed lines. The inconsistencies concern Smith's town of operation, and the stock levels of Tesla X and Tesla Y in, respectively, Boston and New York. For the example database of Fig. 1.1, the following Boolean conjunctive query q_1 holds true if the dealer Smith works in a town where the product Tesla X is stored:

$$q_1 := \exists y \exists z(\mathit{Dealers}(\text{“}\underline{\text{Smith}}\text{”}, y) \wedge \mathit{Stock}(\text{“}\underline{\text{Tesla X}}\text{”}, \underline{y}, z)).$$

In Fig. 1.1, any repair where the dealer Smith works in New York falsifies q_1 and, thus, $\mathsf{CERTAINTY}(q_1)$ returns false when $\mathbf{db}_{\mathsf{Stock}}$ is given as input.

The following conjunctive query q_2 returns the dealers that work in a town where the product Tesla X is stored:

$$q_2(x) := \exists y \exists z(\mathit{Dealers}(\underline{x}, y) \wedge \mathit{Stock}(\text{“}\underline{\text{Tesla X}}\text{”}, \underline{y}, z)).$$

$\mathsf{CERTAINTY}(q_2)$ will return {James} when $\mathbf{db}_{\mathsf{Stock}}$ is given as input since James is the only dealer who works, in every repair, in a town where the product Tesla X is stored. $\triangleleft$

CQA for self-join-free conjunctive queries q, without aggregation, and primary keys has been intensively studied. A systematic study of its complexity for self-join-free conjunctive queries had started already in 2005 (Fuxman & Miller, 2005), and was eventually solved in two journal articles by Koutris & Wijsen (2017, 2021), as follows: for every self-join-free Boolean conjunctive query q, $\mathsf{CERTAINTY}(q)$ is either in FO, L-complete, or coNP-complete, and it is decidable, given q, which case applies. This complexity classification extends to non-Boolean queries by treating free variables as constants. Other extensions beyond this trichotomy deal with foreign keys (Hannula & Wijsen, 2022), more than one key per relation (Koutris & Wijsen, 2020), negated atoms (Koutris & Wijsen, 2018), restricted self-joins (Koutris et al., 2021, 2024; Padmanabha et al., 2024), and data annotated with semiring values (Kolaitis et al., 2025). A more detailed discussion of the latter work appears in Section 6.5. For unions of conjunctive queries q, Fontaine (2015) established interesting relationships between $\mathsf{CERTAINTY}(q)$ and Bulatov's dichotomy theorem for conservative CSP (Bulatov, 2011). Recently, Figueira et al. (2023) proposed a polynomial-time procedure for evaluating $\mathsf{CERTAINTY}(q)$ for conjunctive queries q, including those with self-joins. The procedure correctly handles all polynomial-time cases of $\mathsf{CERTAINTY}(q)$ identified in (Koutris & Wijsen, 2017; Koutris et al., 2021). Nevertheless, for $q = \exists x \exists y \exists z(R(\underline{x}, y, z) \wedge R(\underline{z}, x, y))$, the authors show that $\mathsf{CERTAINTY}(q)$ is in P but their procedure may yield false negatives.

The counting variant $\sharp\mathsf{CERTAINTY}(q)$ asks to count the number of repairs that satisfy some Boolean query q. This counting problem is fundamentally different from the range semantics in the current work. For self-join-free conjunctive queries, $\sharp\mathsf{CERTAINTY}(q)$ exhibits a dichotomy between FP and

$\sharp$P-complete under polynomial-time Turing reductions (Maslowski & Wijsen, 2013). This dichotomy has been shown to extend to queries with self-joins if primary keys are singletons (Maslowski & Wijsen, 2014), and to functional dependencies (Calautti et al., 2022a). Calautti et al. (2019) present a complexity analysis of these counting problems under weaker reductions, in particular, under many-one logspace reductions. The same authors have conducted an experimental evaluation of randomized approximation schemes for approximating the percentage of repairs that satisfy a given query (Calautti et al., 2021). Other approaches to making CQA more meaningful and/or tractable include operational repairs (Calautti et al., 2018, 2022b) and preferred repairs (Kimelfeld et al., 2020; Staworko et al., 2012).

Recent overviews of two decades of theoretical research in CQA can be found in (Bertossi, 2019; Wijsen, 2019; Kimelfeld & Kolaitis, 2024). It is worthwhile to note that theoretical research in $\mathsf{CERTAINTY}(q)$ has stimulated implementations and experiments in prototype systems (Dixit & Kolaitis, 2019; Fan et al., 2023; Fuxman et al., 2005a,b; Amezian El Khalfioui et al., 2020; Kolaitis et al., 2013).

1.3. Range Semantics

It is significant to generalize CQA from Boolean queries, which return either true or false, to *numerical queries*, which return a numeric result. Aggregation queries constitute an important class of numerical queries. However, numerical aggregation queries are likely to return different results on different repairs, and therefore lack a single consistent answer that holds true across all repairs. For this reason, Arenas et al. (2001) have proposed *range semantics*, which provides the greatest lower bound (glb) and the least upper bound (lub) of query answers across all repairs. Specifically, for a *numerical aggregation query* $g()$, the function problems $\mathsf{GLB\text{-}CQA}(g())$ and $\mathsf{LUB\text{-}CQA}(g())$ take a database instance **db** as input, and return, respectively, the glb and the lub of the set that contains each number returned by $g()$ on some repair.

In the current work, we consider numerical aggregation queries $g()$ that take the following form in the extended Datalog syntax of (Cohen et al., 2006, 1999):

$$\mathtt{AGG}(r) \leftarrow q(\vec{u}), \tag{1.1}$$

where the *body* $q(\vec{u})$ is a conjunction of atoms (a.k.a. subgoals), r is either a numeric variable occurring in $\vec{u}$ or a non-negative number, and `AGG` is an aggregate symbol (like `MAX`, `MIN`, `SUM`, `AVG`, `COUNT`). Every aggregate symbol `AGG` in our query language is associated with an aggregate operator, denoted $\mathcal{F}_{\texttt{AGG}}$, which is a function that takes a multiset of numbers, and returns a number. Such a query (1.1) will be called an AGG-query, and is interpreted as follows: let $\theta_1, \theta_2, \ldots, \theta_n$ enumerate all embeddings of the body into **db**, then the query returns $\mathcal{F}_{\texttt{AGG}}(\{\{\theta_1(r), \theta_2(r), \ldots, \theta_n(r)\}\})$. Note that the argument of $\mathcal{F}_{\texttt{AGG}}$ is a multiset, because it is possible that $\theta_i(r) = \theta_j(r)$ for $i \neq j$. For this semantics to be well-defined, each $\theta_i(r)$ must necessarily belong to some numerical domain D. In this case, the aggregate query is said to be *over* D. In this work, we take D to be $\mathbb{Q}_{\geq 0}$.

Example 1.3.1. For the example database of Fig. 1.1, the following query returns the total quantity of cars in stock in Smith's town of operation:

$$\texttt{SUM}(y) \leftarrow \mathit{Dealers}(\text{``}\underline{\text{Smith}}\text{''}, t), \mathit{Stock}(\underline{p, t}, y).$$

In Fig. 1.1, the repair composed of the tuples preceded by † yields the answer 70 (= 35 + 35), which is the smallest result achievable among all repairs. Notably, the greatest result among all repairs is 96, obtained by any repair that contains *Dealers*("Smith", "New York") and *Stock*("Tesla Y", "New York", 96). ◁

An issue arises when a database instance serving as input to GLB-CQA($g()$) or LUB-CQA($g()$) has a repair in which there is no embedding of $q(\vec{u})$—that is, **db** has a repair that falsifies $\exists\vec{u}(q(\vec{u}))$, or equivalently, **db** is a "no"-instance of CERTAINTY($\exists\vec{u}(q(\vec{u}))$). Our semantics so far requires that $\mathcal{F}_{\texttt{AGG}}(\emptyset)$ is defined. However, some aggregate operators, such as $\mathcal{F}_{\texttt{AVG}}$, are not naturally defined over the empty multiset. If $\mathcal{F}_{\texttt{AGG}}(\emptyset)$ is undefined, the problems GLB-CQA($g()$) or LUB-CQA($g()$) may be revised to return a special constant $\perp$ whenever some repair falsifies $\exists\vec{u}(q(\vec{u}))$. Alternatively, this issue can be circumvented by restricting the input to database instances that are "yes"-instances of CERTAINTY($\exists\vec{u}(q(\vec{u}))$).

In this thesis, we investigate the computational complexity of determining GLB-CQA($g()$) and LUB-CQA($g()$). Specifically, we aim to understand under which conditions GLB-CQA($g()$) and LUB-CQA($g()$) are solvable through rewriting in an aggregate logic, denoted AGGR[FOL], which extends first-order logic with aggregate operators along the lines of (Hella et al., 2001). So our central problem takes as input a numerical query $g()$, and asks whether or not there is a numerical query $\varphi()$ in AGGR[FOL] that solves GLB-CQA($g()$) or LUB-CQA($g()$); moreover, when such $\varphi()$ exists, we are interested in constructing it, a task loosely referred to as "(consistent) glb (or lub) rewriting of $g()$ in AGGR[FOL]." A practical motivation for focusing on AGGR[FOL] is that formulas in this logic are well-suited for implementation in SQL, allowing them to benefit from existing DBMS technology.

Chapter 4 of the current thesis is inspired by the PhD thesis of Fuxman (2007) who initiated the complexity study of $\mathsf{GLB\text{-}CQA}(g())$ and $\mathsf{LUB\text{-}CQA}(g())$. Fuxman introduced the class $\mathsf{Cforest}$ of Boolean conjunctive queries, and showed, among other results, that for every *counting query* $g() := \mathtt{SUM}(1) \leftarrow q(\vec{u})$, if the existential closure $\exists\vec{u}(q(\vec{u}))$ of its body belongs to $\mathsf{Cforest}$, then the answers to $\mathsf{GLB\text{-}CQA}(g())$ and $\mathsf{LUB\text{-}CQA}(g())$ on database instances that are "yes"-instances of $\mathsf{CERTAINTY}(\exists\vec{u}(q(\vec{u})))$ can be evaluated by first executing some first-order queries followed by simple counting steps. We refer to this method as *parsimonious aggregation* or *Fuxman's technique*. It can be expressed in a syntactically highly restricted fragment of $\mathsf{AGGR[FOL]}$: one where aggregation is only applied "at the end" on FOL-formulas that themselves do not contain aggregation. It remained an open question to syntactically characterize the class of *all* counting queries $g()$ for which Fuxman's technique applies. This question is resolved as follows: the syntactic class $\mathsf{Cparsimony}$, introduced in Chapter 4, is an extension of $\mathsf{Cforest}$ that contains all (and only) self-join-free conjunctive queries for which Fuxman's technique applies.

As *parsimonious aggregation*, by definition, does not exploit the full expressive power of $\mathsf{AGGR[FOL]}$, it is important to ask for a syntactic characterization of *all* $\mathtt{AGG}$-queries $g() := \mathtt{AGG}(r) \leftarrow q(\vec{u})$ such that $\mathsf{GLB\text{-}CQA}(g())$ can be expressed in $\mathsf{AGGR[FOL]}$; and likewise for $\mathsf{LUB\text{-}CQA}(g())$. Notably, expressibility of either problem in $\mathsf{AGGR[FOL]}$ does not imply expressibility of the other. Furthermore, there is no reason to limit these questions to "yes"-instances of $\mathsf{CERTAINTY}(\exists\vec{u}(q(\vec{u})))$, especially when $\mathcal{F}_{\mathtt{AGG}}(\emptyset)$ is naturally defined—for instance, $\mathcal{F}_{\mathtt{SUM}}(\emptyset) = 0$. These questions are addressed in Chapters 6 and 7 for aggregate operators that are monotone and associative, yielding the following results:

- If $\mathcal{F}_{\mathtt{AGG}}$ is monotone and associative, then the following problem is decidable in quadratic time: given $g()$, can $\mathsf{GLB\text{-}CQA}(g())$ be expressed in $\mathsf{AGGR[FOL]}$?
- If $\mathtt{AGG} = \mathtt{SUM}$, then the following problem is decidable in quadratic time: given $g()$, can $\mathsf{LUB\text{-}CQA}(g())$ be expressed in $\mathsf{AGGR[FOL]}$? Moreover, whenever $\mathcal{F}_{\mathtt{AGG}}$ is monotone and associative, the same decision procedure yields no false positives, but may yield false negatives.

1.4. Organization

This thesis is organized as follows. Chapter 2 introduces some preliminaries, and Chapter 3 introduces the logic $\mathsf{AGGR[FOL]}$ that will serve as the target language for our rewritings, as well as the function problems $\mathsf{GLB\text{-}CQA}(g())$ and $\mathsf{LUB\text{-}CQA}(g())$. In Chapter 4, we formalize the semantic notion of *admitting parsimonious counting*, and then syntactically characterize the class of all counting queries $g() := \texttt{SUM}(1) \leftarrow q(\vec{u})$ that have this property. Moreover, we extend some of our findings to aggregate operators beyond counting. In Chapter 5, we express $\mathsf{GLB\text{-}CQA}(g())$ and $\mathsf{LUB\text{-}CQA}(g())$ in terms of a new construct called *Maximal Consistent Subset* (MCS). This new construct allows us to prove in Chapters 6 and 7, respectively, the two findings mentioned at the end of Section 1.3. Finally, Chapter 8 explores $\mathsf{GLB\text{-}CQA}(g())$ and $\mathsf{LUB\text{-}CQA}(g())$ for numerical queries that are not covered in previous chapters. Chapter 9 concludes the thesis.

1.5. Publications

The work reported in this PhD thesis has already resulted in the following publications:

- Chapter 4 extends a publication at ICDT 2023 (Amezian El Khalfioui & Wijsen, 2023). While the conference version only pertains to counting queries, the corresponding chapter also considers aggregation operators beyond counting.

- The results presented in Chapter 6 appeared in PODS 2025 (Amezian El Khalfioui & Wijsen, 2024a).

- The results of Chapter 7 have been accepted for publication at ICDT 2026 (Amezian El Khalfioui & Wijsen, 2026).

The technical treatment in this thesis differs slightly from these existing and forthcoming publications, as we have unified and generalized certain definitions for consistency and clarity in the context of the thesis.

CHAPTER 2

Preliminaries

We assume denumerable sets **var** and **dom** of variables and constants respectively. The set **dom** includes $\mathbb{Q}_{\geq 0}$, the set of non-negative rational numbers. The set of *numerical variables* is a subset of **var**.

We assume denumerably many *relation names*. Every relation name is associated with a *signature*, which is a triple (n, k, J) where n is the *arity*, $\{1, \ldots, k\}$ is called the *primary key*, and $J \subseteq \{1, \ldots, n\}$ is the set of *numerical positions* (also called *numerical columns*). This relation name is *full-key* if $n = k$. A relation name that is not full-key has *non-primary key positions* $k{+}1, k{+}2, \ldots, n$. Note that each relation name R is associated with exactly one key constraint, which is determined by the signature of R. For an n-tuple $\vec{x} = (x_1, \ldots, x_n)$, we write $|\vec{x}|$ to denote its *arity* n. We often blur the distinction between a sequence $(x_1, \ldots, x_n)$ of distinct variables and the set $\{x_1, \ldots, x_n\}$, which is also denoted $\mathsf{vars}(\vec{x})$. If $\vec{x} = (x_1, \ldots, x_n)$ and $\vec{y} = (y_1, \ldots, y_m)$, then we define their *concatenation* $\vec{x} \cdot \vec{y}$ (or $\vec{x}\vec{y}$) as the tuple $(x_1, \ldots, x_n, y_1, \ldots, y_m)$.

Atoms, facts, and database instances. Let R be a relation name of signature (n, k, J). An *atom* is an expression $R(u_1, \ldots, u_n)$ where each u_i is either a constant or a variable, and for every $j \in J$, u_j is a numerical variable or a number in $\mathbb{Q}_{\geq 0}$. It is common to underline positions of the primary key. An atom is said to be *full-key* if its relation name is full-key. If F is an atom, then $\mathsf{vars}(F)$ is the set of variables that occur in F, and $\mathsf{Key}(F)$ is the set of variables that occur in F at a position of the primary key. Further, we define $\mathsf{notKey}(F) := \mathsf{vars}(F) \setminus \mathsf{Key}(F)$. A *fact* is an atom without variables. A fact with relation name R is also called an R-fact. Two facts $R_1(\underline{\vec{a}_1}, \vec{c}_1)$ and $R_2(\underline{\vec{a}_2}, \vec{c}_2)$ are said to be *key-equal* if $R_1 = R_2$ and $\vec{a}_1 = \vec{a}_2$.

A *database instance* **db** is a finite set of facts. We write $\mathsf{adom}(\mathbf{db})$ for the *active domain of* **db**, i.e., the set of all constants that occur in **db**. If R is a relation name, then the *R-relation of* **db** is the set of all R-facts in **db**. A *database instance* is *consistent* if it does not contain two distinct facts that are key-equal. Primary keys need not be explicitly specified, as they are determined by the predefined relation signatures.

Valuations. A *valuation* over a finite set U of variables is a total mapping θ from U to **dom** such that $\theta(r) \in \mathbb{Q}_{\geq 0}$ for every numerical variable r. For a valuation θ over U, we write $\mathsf{dom}(\theta)$ to denote its *domain U*. A valuation θ over U is extended to every element u in $\mathbf{var} \cup \mathbf{dom}$ by letting $\theta(u) = u$ for every $u \notin U$.

Let θ be a valuation. If F is the atom $R(u_1, \dots, u_n)$, then $\theta(F) := R(\theta(u_1), \dots, \theta(u_n))$. If q is a set of atoms, then $\theta(q) := \{\theta(F) \mid F \in q\}$. Notably, every variable in $\mathsf{vars}(q) \setminus \mathsf{dom}(\theta)$ remains a variable in $\theta(q)$. If θ is a valuation and $V \subseteq \mathsf{dom}(\theta)$, then $\theta \restriction_V$ denotes the restriction of θ to V, i.e., $\mathsf{dom}(\theta \restriction_V) = V$ and for every $x \in V$, we have $\theta \restriction_V (x) = \theta(x)$. Let θ and μ be valuations such that $\mathsf{dom}(\theta) \cap \mathsf{dom}(\mu) = \emptyset$. We write $\theta \cdot \mu$ for the valuation over $\mathsf{dom}(\theta) \cup \mathsf{dom}(\mu)$ that extends both θ and μ.

Partial valuation. Let **db** be a database instance, and $\varphi(\vec{x})$ a first-order formula with free variables $\vec{x}$. Let θ be a valuation. Then we write $(\mathbf{db}, \theta) \models \varphi(\vec{x})$ to denote that θ can be extended to a valuation θ' over $\mathsf{dom}(\theta) \cup \mathsf{vars}(\vec{x})$ such that for $\vec{a} := \theta'(\vec{x})$, we have $\mathbf{db} \models \varphi(\vec{a})$ using standard semantics (see, e.g., (Libkin, 2004, p. 15)). Typically, but not necessarily, $\mathsf{dom}(\theta) \subseteq \mathsf{vars}(\vec{x})$. If φ has no free variables, then we write $\mathbf{db} \models \varphi$ instead of $(\mathbf{db}, \varnothing) \models \varphi$, where $\varnothing$ is the empty valuation.

Repairs and $\mathsf{CERTAINTY}(\varphi)$. A *repair* of a database instance is a $\subseteq$-maximal consistent subset of it. We write $\mathsf{rset}(\mathbf{db})$ for the set of repairs of a database instance **db**. If $\varphi(\vec{x})$ is a first-order formula and θ a valuation, then we write $(\mathbf{db}, \theta) \models_{\mathsf{cqa}} \varphi(\vec{x})$ to denote that for every repair $\mathbf{r}$ of **db**, we have $(\mathbf{r}, \theta) \models \varphi(\vec{x})$. If φ has no free variables, then we write $\mathbf{db} \models_{\mathsf{cqa}} \varphi$ instead of $(\mathbf{db}, \varnothing) \models_{\mathsf{cqa}} \varphi$, where $\varnothing$ is the empty valuation. For a closed formula φ, $\mathsf{CERTAINTY}(\varphi)$ is the decision problem that takes a database instance **db** as input and determines whether or not $\mathbf{db} \models_{\mathsf{cqa}} \varphi$.

Conjunctive Queries. A *self-join-free conjunctive query* q is a closed first-order formula $\exists \vec{u}(R_1(\vec{u}_1) \wedge \cdots \wedge R_n(\vec{u}_n))$, where each $R_i(\vec{u}_i)$ is an atom, $\vec{u}$ is a sequence containing variables occurring in some $\vec{u}_i$, and $i \neq j$ implies $R_i \neq R_j$. We will call the variables in $\vec{u}$ *bound variables.* Every non-bound variable in some $\vec{u}_i$ is called a *free variable*, and we denote by $\mathsf{free}(q)$ the set of free variables occurring in q. The conjunction $R_1(\vec{u}_1) \wedge \cdots \wedge R_n(\vec{u}_n)$, whose free variables are $\vec{u}$, is called the *body* of the query, denoted $\mathsf{body}(q)$. We often blur the distinction between the query q, its body, and the set $\{R_1(\vec{u}_1), \dots, R_n(\vec{u}_n)\}$. For example, if F is an atom of (the body of) q, then $q \setminus \{F\}$ is the query obtained from q by deleting F from its body. A *self-join-free Boolean conjunctive query* q is a self-join-free conjunctive query without free variables. We write sjfBCQ for the set of self-join-free Boolean conjunctive queries. We now introduce operators for turning bound variables into free variables, or vice versa, and for instantiating free variables.

Making bound variables free. Let q be a conjunctive query with $\mathsf{free}(q) = \vec{z}$. Let $\vec{x}$ be a tuple of (not necessarily all) bound variables in q (hence $\vec{x} \cap \vec{z} = \emptyset$). We write $\nexists\vec{x}\,[q]$ for the conjunctive query q' such that $\mathsf{free}(q') = \vec{z} \cup \vec{x}$ and $\mathsf{body}(q') = \mathsf{body}(q)$. Informally, $\nexists\vec{x}\,[q]$ is obtained from q by omitting the quantification $\exists\vec{x}$. For example, if $q(z) = \exists x \exists y (R(x, y) \wedge R(y, z))$, then $\nexists x\,[q] = \exists y (R(x, y) \wedge R(y, z))$.

Binding free variables. Let q be a conjunctive query, and $\vec{x}$ a tuple of (not necessarily all) free variables of q. Then $\exists\vec{x}\,[q]$ denotes the query with the same body as q, but whose set of free variables is $\mathsf{free}(q) \setminus \vec{x}$.

Instantiating free variables. Let q be a conjunctive query, and $\vec{z}$ a tuple of distinct free variables of q. Let $\vec{c}$ be a tuple of constants of arity $|\vec{z}|$. Then $q_{[\vec{z} \mapsto \vec{c}]}$ is the query obtained from q by replacing, for every $i \in \{1, 2, \dots, |\vec{z}|\}$, each occurrence of the ith variable in $\vec{z}$ by the ith constant in $\vec{c}$.

Gaifman Graph. The *Gaifman graph of* q, denoted $\mathcal{G}aifman(q)$, is an undirected simple graph whose vertex-set is $\mathsf{vars}(q)$. There is an edge between x and y if $x \neq y$ and some atom of q contains both x and y. We write $E(\mathcal{G}aifman(q))$ for the edge-set of $\mathcal{G}aifman(q)$.

Guardedness. Let q_1 and q_2 be two queries in sjfBCQ. We say that q_1 *is guarded by* q_2, denoted $q_1 \preccurlyeq_{\mathsf{g}} q_2$, if for every atom F in q_1, there exists an atom G in q_2 such that $\mathsf{vars}(F) \subseteq \mathsf{vars}(G)$. We write $q_1 \sim_{\mathsf{g}} q_2$ if both $q_1 \preccurlyeq_{\mathsf{g}} q_2$ and $q_2 \preccurlyeq_{\mathsf{g}} q_1$. It is straightforward that $q_1 \sim_{\mathsf{g}} q_2$ implies $\mathcal{G}aifman(q_1) = \mathcal{G}aifman(q_2)$ (but the converse does not hold).

Attack graph. Attack graphs for queries q in sjfBCQ were first introduced in (Wijsen, 2012), later generalized in (Koutris & Wijsen, 2017), and have since been used in several studies, e.g., (Figueira et al., 2025; Amezian El Khalfioui & Wijsen, 2024a; Kolaitis et al., 2025). We write $\mathcal{K}(q)$ for the set of functional dependencies that contains $\mathsf{Key}(F) \rightarrow \mathsf{vars}(F)$ whenever $F \in q$. For $F \in q$, we define $F^{+,q} := \{x \in \mathsf{vars}(q) \mid \mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F) \rightarrow x\}$, where $\models$ is

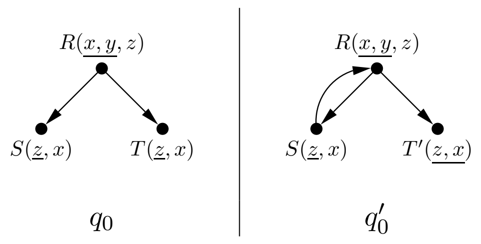

Figure 2.1: Attack graphs for two queries in sjfBCQ. The query q_0' on the right is derived from the query on the left by replacing $T(\underline{x},z)$ with $T'(\underline{x,z})$.

the standard notion of logical implication. An atom F of q is said to *attack* a variable x, denoted $F \overset{q}{\rightsquigarrow} x$, if $\mathcal{G}aifman(q)$ contains a (possibly empty) path between some variable in $\mathsf{notKey}(F)$ and x such that no variable on the path belongs to $F^{+,q}$. A variable x is said to be *unattacked (in q)* if no atom attacks x. The *attack graph* of q is a directed simple graph whose vertices are the atoms of q. There is a directed edge from F to G, denoted $F \overset{q}{\rightsquigarrow} G$, if F attacks some variable of $\mathsf{vars}(G)$. Attack graphs are also defined for self-join-free conjunctive queries in a similar way, the only difference being that $\mathcal{K}(q)$ also contains $\emptyset \to x$ for every free variable $x \in \mathsf{free}(q)$.

Whenever a query in sjfBCQ is clear from the context, we can use a relation name as a shorthand for the unique atom with that relation name in the query. For example, in the following example, R is used as a shorthand for the atom $R(\underline{x},y)$.

Example 2.0.1. Consider the sjfBCQ query q_0 on the left side of Fig. 2.1. The directed edges represent attacks. We have $R^{+,q_0} = \{x,y\}$, $T^{+,q_0} = \{z,x\}$, and $S^{+,q_0} = \{z,x\}$. The sequence (z) entails $R \overset{q_0}{\rightsquigarrow} S$ and $R \overset{q_0}{\rightsquigarrow} T$. The query q_0' on the right side is obtained from q_0 by replacing the T-atom with a full-key atom, which introduces a cycle in the attack graph. $\triangleleft$

The following result shows the usefulness of attack graphs.

Theorem 2.0.1 (Koutris & Wijsen (2017))**.** *For every query q in* sjfBCQ*,* $\mathsf{CERTAINTY}(q)$ *is in* FO *if and only if the attack graph of q is acyclic.*

Sequential proof. Let q be a query in sjfBCQ. Assume that $\mathcal{K}(q) \models X \to y$. A *sequential proof of $\mathcal{K}(q) \models X \to y$* is a (possibly empty) sequence $(F_1, F_2, \dots, F_n)$ of atoms in q such that $y \in X \cup (\bigcup_{i=1}^{n} \mathsf{vars}(F_i))$ and for every $j \in \{1, 2, \dots, n\}$, we have $\mathsf{Key}(F_j) \subseteq X \cup \left(\bigcup_{i=1}^{j-1} \mathsf{vars}(F_i)\right)$.

Embeddings. Whenever $q(\vec{u})$ is a conjunction of atoms, we write q to denote the closed formula $\exists \vec{u}(q(\vec{u}))$. Let $q(\vec{u})$ now be a self-join-free conjunction of n atoms such that the attack graph of q is acyclic. The following definitions are relative to a fixed topological sort $(F_1, \ldots, F_n)$ of q's attack graph and a fixed database instance $\mathbf{db}$. We define the following sequences of variables for $\ell \in \{1, \ldots, n\}$:

- $\vec{u}_\ell$ contains all (and only) variables of $\bigcup_{i=1}^{\ell} \mathsf{vars}(F_i)$. Thus, $\vec{u}_n = \vec{u}$;
- $\vec{x}_\ell$ contains the variables of $\mathsf{Key}(F_\ell)$ that do not occur in $\bigcup_{i=1}^{\ell-1} \mathsf{vars}(F_i)$; and
- $\vec{y}_\ell$ contains the variables of $\mathsf{notKey}(F_\ell)$ that do not already occur in $\bigcup_{i=1}^{\ell-1} \mathsf{vars}(F_i)$.

Moreover, we define $\vec{u}_0 = ()$, the empty sequence. With this notation, we have that for every $\ell \in \{1, \ldots, n\}$,

$$\vec{u}_\ell = (\vec{u}_{\ell-1}, \vec{x}_\ell, \vec{y}_\ell). \tag{2.1}$$

Let $\ell \in \{1, \ldots, n\}$. An *$\ell$-embedding (of q in* $\mathbf{db}$*)* is a valuation θ over $\vec{u}_\ell$ such that $(\mathbf{db}, \theta) \models q(\vec{u})$. A *0-embedding* is defined to be the empty set. An ℓ-embedding with $\ell = n$ is also called an *embedding* for short. We write $\mathbf{Emb}_q(\mathbf{db})$ for the set containing every embedding of q in $\mathbf{db}$. A subset $M \subseteq \mathbf{Emb}_q(\mathbf{db})$ is said to be *consistent* if $M \models \mathcal{K}(q)$. An *ℓ-key-embedding* (of q in $\mathbf{db}$) is a valuation θ over $\vec{u}_{\ell-1} \cdot \vec{x}_\ell$ such that $(\mathbf{db}, \theta) \models q(\vec{u})$.

Reductions. Let P_1 and P_2 be function problems whose output is a single number. If I is an instance of P_j, we write $P_j(I)$ to denote the output of P_j on input I, where $j \in \{1, 2\}$. We say that P_1 is first-order reducible to P_2, denoted $P_1 \leq_{\mathsf{FO}} P_2$ if there exits a first-order definable function f such that for every instance I of P_1, $f(I)$ is an instance of P_2 such that $P_1(I) = P_2(f(I))$. We write $P_1 \leq_{\mathsf{nr\text{-}datalog}} P_2$ if the reduction can even be expressed by a non-recursive datalog ($\mathsf{nr\text{-}datalog}$) program, as defined in (Abiteboul et al., 1995, page 62–63).

CHAPTER 3

Aggregate Logics and CQA

In this chapter, we first define the notion of *aggregate operator* in Section 3.1. Sections 3.2 and 3.3 will respectively introduce the logics AGGR[FOL] and AGGR[nr-datalog], which will serve as the target languages for our rewritings. In Section 3.4, we formally define consistent lower and upper bounds, denoted LUB-CQA($g(\vec{x})$) and GLB-CQA($g(\vec{x})$), for arbitrary numerical terms $g(\vec{x})$ in AGGR[FOL] with free variables $\vec{x}$. Our study will focus on closed numerical terms $g()$ in AGGR[sjfBCQ], a subclass of AGGR[FOL] introduced in Definition 3.4.1. Finally, in Section 3.5, we discuss how GLB-CQA and CERTAINTY problems are related.

3.1. Aggregating Non-Negative Numbers

Let S be a set. A multiset is said to be *over* S if all of its elements belong to S. A *(positive) aggregate operator* is a function $\mathcal{F}_{\texttt{AGG}}$ (where AGG is an *aggregate symbol*) that maps each nonempty finite multiset over $\mathbb{Q}_{\geq 0}$ to a non-negative rational number. Some aggregate operators may also map the empty multiset to a non-negative rational number, in which case we say that $\mathcal{F}_{\texttt{AGG}}(\emptyset)$ is defined.

In the following definitions, all multisets are understood to be multisets over $\mathbb{Q}_{\geq 0}$.

Associativity. An aggregate operator is *associative* if for all multisets X and Y such that $\mathcal{F}_{\texttt{AGG}}(X)$ is defined, we have:

$$\mathcal{F}_{\texttt{AGG}}(X \uplus Y) = \mathcal{F}_{\texttt{AGG}}(\{\{\mathcal{F}_{\texttt{AGG}}(X)\}\} \uplus Y), \tag{3.1}$$

where $\uplus$ denotes union of multisets. Note that $\mathcal{F}_{\texttt{AGG}}(X)$ is undefined exactly when $X = \emptyset$ and $\mathcal{F}_{\texttt{AGG}}(\emptyset)$ is undefined.

Example 3.1.1. Examples of associative aggregate operators are $\mathcal{F}_{\texttt{SUM}}$, $\mathcal{F}_{\texttt{MAX}}$, and $\mathcal{F}_{\texttt{MIN}}$. On the other hand, $\mathcal{F}_{\texttt{AVG}}$, $\mathcal{F}_{\texttt{COUNT}}$, and $\mathcal{F}_{\texttt{SUM-DISTINCT}}$ are not associative. Note, for instance, that we have $\mathcal{F}_{\texttt{COUNT}}(\{\{5, 6, 7, 8\}\}) = 4$ and $\mathcal{F}_{\texttt{COUNT}}(\{\{\mathcal{F}_{\texttt{COUNT}}(\{\{5, 6, 7\}\}), 8\}\}) = \mathcal{F}_{\texttt{COUNT}}(\{\{3, 8\}\}) = 2$. $\lhd$

If $\mathcal{F}_{\texttt{AGG}}(\emptyset)$ is defined, by letting $X = \emptyset$ in Eq. 3.1, we obtain that if $\mathcal{F}_{\texttt{AGG}}$ is associative, then for all multisets Y, we have $\mathcal{F}_{\texttt{AGG}}(Y) = \mathcal{F}_{\texttt{AGG}}(\{\{\mathcal{F}_{\texttt{AGG}}(\emptyset)\}\} \uplus Y)$. This condition holds for $\mathcal{F}_{\texttt{SUM}}$ and $\mathcal{F}_{\texttt{MAX}}$ over $\mathbb{Q}_{\geq 0}$ if and only if we define $\mathcal{F}_{\texttt{SUM}}(\emptyset) := 0$ and $\mathcal{F}_{\texttt{MAX}}(\emptyset) := 0$. On the other hand, for every $r \in \mathbb{Q}_{\geq 0}$, if we defined $\mathcal{F}_{\texttt{MIN}}(\emptyset)$ to be equal to r, then $\mathcal{F}_{\texttt{MIN}}$ would no longer be associative, because $\mathcal{F}_{\texttt{MIN}}(\{\{r+1\}\}) \neq \mathcal{F}_{\texttt{MIN}}(\{\{r, r+1\}\})$. Throughout this thesis, $\mathcal{F}_{\texttt{SUM}}(\emptyset) = \mathcal{F}_{\texttt{MAX}}(\emptyset) = 0$, while $\mathcal{F}_{\texttt{MIN}}(\emptyset)$ is undefined.

Monotonicity. An aggregate operator $\mathcal{F}_{\texttt{AGG}}$ is *monotone* if for all $m \geq 0$ such that $\mathcal{F}_{\texttt{AGG}}(\{\{x_1, \dots, x_m\}\})$ is defined, and for every (possibly empty) multiset Y, we have:

$$\mathcal{F}_{\texttt{AGG}}(\{\{x_1, \dots, x_m\}\}) \leq \mathcal{F}_{\texttt{AGG}}(\{\{x'_1, \dots, x'_m\}\} \uplus Y) \text{ whenever } x_i \leq x'_i \text{ for every } i. \tag{3.2}$$

Note that $\mathcal{F}_{\texttt{AGG}}(\{\{x_1, \dots, x_m\}\})$ is undefined exactly when $m = 0$ and $\mathcal{F}_{\texttt{AGG}}(\emptyset)$ is undefined. If $\mathcal{F}_{\texttt{AGG}}(\emptyset)$ is defined, by letting $m = 0$ in Eq. 3.2, we obtain that if $\mathcal{F}_{\texttt{AGG}}$ is monotone, then for all multisets Y, we have $\mathcal{F}_{\texttt{AGG}}(\emptyset) \leq \mathcal{F}_{\texttt{AGG}}(Y)$. Again, this condition holds for $\mathcal{F}_{\texttt{SUM}}$ and $\mathcal{F}_{\texttt{MAX}}$ over $\mathbb{Q}_{\geq 0}$ if and only if we define $\mathcal{F}_{\texttt{SUM}}(\emptyset) := 0$ and $\mathcal{F}_{\texttt{MAX}}(\emptyset) := 0$.

Example 3.1.2. Examples of monotone aggregate operators are $\mathcal{F}_{\texttt{SUM}}$, $\mathcal{F}_{\texttt{MAX}}$, and $\mathcal{F}_{\texttt{COUNT}}$. Note that $\mathcal{F}_{\texttt{MIN}}$ is not monotone since $\mathcal{F}_{\texttt{MIN}}(\{\{3\}\}) > \mathcal{F}_{\texttt{MIN}}(\{\{2, 3\}\})$. `COUNT-DISTINCT` also lacks monotonicity: if we increase 3 to 4 in the multiset $\{\{3, 4\}\}$, the number returned by `COUNT-DISTINCT` drops from 2 to 1. $\lhd$

Significantly, an aggregate operator $\mathcal{F}_{\texttt{AGG}}$ that is not monotone in general may become monotone under a restriction of its underlying domain. For example, $\mathcal{F}_{\texttt{PRODUCT}}$, defined by $\mathcal{F}_{\texttt{PRODUCT}}(\{\{x_1, \dots, x_m\}\}) := \Pi_{i=1}^{m} x_i$, is not monotone over $\mathbb{Q}_{\geq 0}$ (because $\mathcal{F}_{\texttt{PRODUCT}}(\{\{1\}\}) > \mathcal{F}_{\texttt{PRODUCT}}(\{\{1, \frac{1}{2}\}\})$), but is monotone over $\mathbb{Q}_{\geq 1}$.

3.2. The Logic AGGR[FOL]

Our treatment of aggregate logic follows the approach in (Hella et al., 2001; Libkin, 2004). We write AGGR[FOL] for the extension of predicate calculus with numerical terms introduced next.

Every formula in classical predicate calculus is also a formula in AGGR[FOL]. In AGGR[FOL], terms are not restricted to variables and constants; they also include numerical terms, as defined below. Let $q(\vec{x}, \vec{y})$ be a formula in AGGR[FOL], where $\vec{x}$, $\vec{y}$ are disjoint sequences that together contain each free variable of q exactly once. A *primitive numerical term* is either a non-negative rational number or a numerical variable in $\vec{x} \cdot \vec{y}$. For every possible aggregate operator $\mathcal{F}_{\mathsf{AGG}}$, a primitive numerical term r, and a formula $q(\vec{x}, \vec{y})$, we have a new *numerical term*

$$g(\vec{x}) := \mathsf{Aggr}_{\mathcal{F}_{\mathsf{AGG}}} \vec{y}\, [r, q(\vec{x}, \vec{y})] \, .$$

Variables $\vec{y}$ that are free in $q(\vec{x}, \vec{y})$ become bound in $\mathsf{Aggr}_{\mathcal{F}_{\mathsf{AGG}}} \vec{y}\, [r, q(\vec{x}, \vec{y})]$; in this respect $\mathsf{Aggr}_{\mathcal{F}_{\mathsf{AGG}}} \vec{y}$ behaves like a sort of quantification over $\vec{y}$. Next, we define the semantics.

Let $\vec{a}$ be a sequence of constants of length $|\vec{x}|$. The value $g(\vec{a})$ on a database instance $\mathbf{db}$, denoted $[\![g(\vec{a})]\!]^{\mathbf{db}}$, is calculated as follows. Let $\{\theta_1, \ldots, \theta_m\}$ be a (possibly empty) $\subseteq$-maximal set of valuations over $\vec{x} \cdot \vec{y}$ such that $\theta_i(\vec{x}) = \vec{a}$ and $(\mathbf{db}, \theta_i) \models q(\vec{x}, \vec{y})$ for $1 \leq i \leq m$. Then $[\![g(\vec{a})]\!]^{\mathbf{db}} = \mathcal{F}_{\mathsf{AGG}}(\{\!\{\theta_1(r), \ldots, \theta_m(r)\}\!\})$. Note that the argument of $\mathcal{F}_{\mathsf{AGG}}$ is in general a multiset, since $\theta_i(r)$ may be equal to $\theta_j(r)$ for $i \neq j$. A numerical term without free variables is also called a *numerical query*, denoted $g()$.

Example 3.2.1. If $g_0(t) := \mathsf{Aggr}_{\mathcal{F}_{\mathsf{SUM}}} (p, z)\, [z, \mathit{Stock}(p, t, z)]$, then on the example database of Fig. 1.1, $g_0(\text{“Boston”}) = 110$, the sum of quantities in Boston. This numerical term can be used in other formulas. For example, $q_0(t, y) := \exists p \exists z (\mathit{Stock}(p, t, z)) \land y = g_0(t)$ returns, for each town t, the total quantity y of products stored in t. $\triangleleft$

3.3. The Logic AGGR[nr-datalog]

We assume that the reader is familiar with the concept of a *nonrecursive datalog program (*nr-datalog *program)*, whose definition can be found in (Abiteboul et al., 1995, pp. 62–63). We now extend such programs by incorporating aggregation. Assume that an nr-datalog program contains the following rule, where r is either a number or a numerical variable:

$$P(\vec{x}, r) \leftarrow R_1(\vec{u}_1), \ldots, R_n(\vec{u}_n). \tag{3.3}$$

Note that by the safety requirement of $\mathsf{nr\text{-}datalog}$, if r is a variable, it also occurs in the body of the rule. We allow such a rule to be replaced with:

$$P(\vec{x}, \mathtt{AGG}(r)) \leftarrow R_1(\vec{u}_1), \ldots, R_n(\vec{u}_n),$$

where $\mathtt{AGG}$ is an aggregate symbol. The semantics on a given database $\mathbf{db}$ is as follows. Let $B(\vec{u})$ be the conjunction of all atoms in the body of the rule, where $\vec{u}$ is a shortest sequence containing every variable that occurs in the body of the rule. Let $\vec{a}$ be a sequence of constants of length $|\vec{x}|$. Let $\{\theta_1, \theta_2, \ldots, \theta_m\}$ be a $\subseteq$-maximal set of valuations over $\vec{u}$ such that $\theta_i(\vec{x}) = \vec{a}$ and $(\mathbf{db}, \theta_i) \models B(\vec{u})$ for $1 \leq i \leq m$. If $m \geq 1$, then the rule derives $P(\vec{a}, n)$ with $n = \mathcal{F}_{\mathtt{AGG}}(\{\{\theta_1(r), \ldots, \theta_m(r)\}\})$. Note that the argument of $\mathcal{F}_{\mathtt{AGG}}$ is in general a multiset, since $\theta_i(r)$ may be equal to $\theta_j(r)$ for $i \neq j$. In case $m = 0$, no such fact is derived.

We write $\mathsf{AGGR[nr\text{-}datalog]}$ for the language consisting of all programs obtained from $\mathsf{nr\text{-}datalog}$ programs by allowing aggregate operators in the head (but not in the body). Example 3.3.1 shows a program in $\mathsf{AGGR[nr\text{-}datalog]}$.

Example 3.3.1. Retrieve the town(s) with the largest total quantity of stored products.

$$\begin{aligned}
\mathit{TotalStockPerCity}(t, \mathtt{SUM}(z)) &\leftarrow \mathit{Stock}(p, t, z)\\
\mathit{MaxStock}(\mathtt{MAX}(s)) &\leftarrow \mathit{TotalStockPerCity}(t, s)\\
\mathit{Answer}(t) &\leftarrow \mathit{TotalStockPerCity}(t, s), \mathit{MaxStock}(s)
\end{aligned}$$

Replacing $\mathtt{SUM}(z)$ and $\mathtt{MAX}(s)$ with z and s, respectively, in the above program yields a standard $\mathsf{nr\text{-}datalog}$ program, as required by $\mathsf{AGGR[nr\text{-}datalog]}$. $\lhd$

3.4. Lower and Upper Bounds Across All Repairs

For each numerical term $g(\vec{x}) := \mathsf{Aggr}_{\mathcal{F}_{\mathtt{AGG}}}\vec{y}\,[r, q(\vec{x}, \vec{y})]$, we define $\mathsf{GLB\text{-}CQA}(g(\vec{x}))$ and $\mathsf{LUB\text{-}CQA}(g(\vec{x}))$ relative to a database instance $\mathbf{db}$. Let $q'(\vec{x})$ be the self-join-free conjunctive query $\exists\vec{y}(q(\vec{x}, \vec{y}))$. Let $\vec{a}$ be a sequence of constants of length $|\vec{x}|$.

If $\mathcal{F}_{\mathtt{AGG}}(\emptyset)$ is defined, then

$$[\![\mathsf{GLB\text{-}CQA}(g(\vec{a}))]\!]^{\mathbf{db}} := \min\left\{[\![g(\vec{a})]\!]^{\mathbf{r}} \mid \mathbf{r} \in \mathsf{rset}(\mathbf{db})\right\}. \tag{3.4}$$

If $\mathcal{F}_{\mathtt{AGG}}(\emptyset)$ is undefined, then

(a) $[\![\mathsf{GLB\text{-}CQA}(g(\vec{a}))]\!]^{\mathbf{db}} = \bot$ if some repair of $\mathbf{db}$ falsifies $q'(\vec{a})$; and

(b) $[\![\mathsf{GLB\text{-}CQA}(g(\vec{a}))]\!]^{\mathbf{db}} := \min\{[\![g(\vec{a})]\!]^{\mathbf{r}} \mid \mathbf{r} \in \mathsf{rset}(\mathbf{db})\}$ otherwise.

The case (a) in this definition is justified by the observation that for every repair $\mathbf{r}$ of $\mathbf{db}$, we have $[\![g(\vec{a})]\!]^{\mathbf{r}} = \mathcal{F}_{\mathtt{AGG}}(\emptyset)$ if and only if $\mathbf{r} \not\models q'(\vec{a})$. The value $[\![\mathsf{LUB\text{-}CQA}(g(\vec{a}))]\!]^{\mathbf{db}}$ is defined symmetrically by replacing min with max.

In this work, we will study under which conditions $\mathsf{GLB\text{-}CQA}(g(\vec{a}))$ and $\mathsf{LUB\text{-}CQA}(g(\vec{a}))$ can be expressed in $\mathsf{AGGR}[\mathsf{FOL}]$. We will consider numerical terms of the form $g(\vec{x}) := \mathsf{Aggr}_{\mathcal{F}_{\mathtt{AGG}}}\vec{y}\,[r, q(\vec{x},\vec{y})]$ where $q(\vec{x},\vec{y})$ is a self-join-free conjunction of atoms. For readability, we often express such a numerical term $g(\vec{x})$ using the following Datalog-like syntax:

$$(\vec{x}, \mathtt{AGG}(r)) \leftarrow q(\vec{x},\vec{y}),$$

where the aggregate symbol $\mathtt{AGG}$ is interpreted by the aggregate operator $\mathcal{F}_{\mathtt{AGG}}$, which is often made explicit by writing $\mathcal{F}_{\mathtt{AGG}}$. For instance, $\mathtt{SUM}$ is interpreted by $\mathcal{F}_{\mathtt{SUM}}$, and $\mathtt{MAX}$ by $\mathcal{F}_{\mathtt{MAX}}$.

In the initial technical treatment, we assume that $\vec{x}$ is empty, i.e., we are dealing with numerical terms $g() := \mathsf{Aggr}_{\mathcal{F}_{\mathtt{AGG}}}\vec{y}\,[r, q(\vec{y})]$ without free variables, which are conveniently expressed as $\mathtt{AGG}(r) \leftarrow q(\vec{y})$. In Section 8.5 of Chapter 8, we treat the extension to free variables. We will also exclude numerical queries $g()$ for which there exists a rational number r such that, for every database instance $\mathbf{db}$ (including the empty database instance), $[\![g()]\!]^{\mathbf{db}} = r$, and consequently, $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}} = [\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}} = r$. An example is $\mathtt{SUM}(0) \leftarrow R(\underline{x}, y), S(\underline{y})$. These numerical queries are of no practical interest, and excluding them simplifies the technical development by avoiding the need to treat them as special cases. The following definition pinpoints the class of queries we are interested in.

Definition 3.4.1. $\mathsf{AGGR}[\mathsf{sjfBCQ}]$ is defined as the class of numerical queries $\mathsf{Aggr}_{\mathcal{F}_{\mathtt{AGG}}}\vec{y}\,[r, q(\vec{y})]$ where $q(\vec{y})$ is a self-join-free conjunction of atoms, r is either a numerical variable in $\vec{y}$, or a non-negative rational number and the following condition is satisfied:

> *Nontriviality Condition:* if $\mathcal{F}_{\mathtt{AGG}}(\emptyset)$ is defined, then there is a database instance $\mathbf{db}$ such that $[\![g()]\!]^{\mathbf{db}} \neq \mathcal{F}_{\mathtt{AGG}}(\emptyset)$.

An alternative syntax is $\mathtt{AGG}(r) \leftarrow q(\vec{y})$, where the aggregate symbol $\mathtt{AGG}$ is interpreted by $\mathcal{F}_{\mathtt{AGG}}$. We call $\mathtt{AGG}(r)$ the *head*, and $q(\vec{y})$ the *body*. We write $\mathsf{BCQ}(g())$ for the Boolean query $\exists\vec{y}(q(\vec{y}))$, which belongs to sjfBCQ. $\square$

3.5. When is $\mathsf{GLB\text{-}CQA}(g())$ Not in $\mathsf{AGGR[FOL]}$?

In this section, we provide a sufficient condition for $\mathsf{GLB\text{-}CQA}(g())$ to be inexpressible in $\mathsf{AGGR[FOL]}$ by utilizing its relationship with the decision problem $\mathsf{CERTAINTY}(\mathsf{BCQ}(g()))$. Section 3.5.1 explores this relationship, and Section 3.5.2 focuses on the specific case where the aggregate operator used is monotone and associative.

3.5.1 Reducing CERTAINTY to GLB-CQA

Let $g()$ be a numerical query in $\mathsf{AGGR[sjfBCQ]}$ with head $\mathtt{AGG}(r)$, and let $q = \mathsf{BCQ}(g())$. Theorem 3.5.1 states that, under some mild conditions, we have $\mathsf{CERTAINTY}(q) \leq_{\mathsf{FO}} \mathsf{GLB\text{-}CQA}(g())$. One of these conditions involves a special case of an injective function, called a *separator*.

Definition 3.5.1 (Separator)**.** If ρ is a function from $\mathbb{Q}_{\geq 0}$ to $\mathbb{Q}_{\geq 0}$, and X is a multiset over $\mathbb{Q}_{\geq 0}$, then $\rho(X)$ denotes the multiset obtained by applying ρ to each element of X, preserving multiplicities. That is, if $X = \{\{a_1, \dots, a_n\}\}$, then $\rho(X) = \{\{\rho(a_1), \dots, \rho(a_n)\}\}$.

Let $\mathcal{F}_{\mathtt{AGG}}$ be an aggregate operator such that $\mathcal{F}_{\mathtt{AGG}}(\emptyset)$ is defined. A *separator for* $\mathcal{F}_{\mathtt{AGG}}$ is an injective function ρ from $\mathbb{Q}_{\geq 0}$ to $\mathbb{Q}_{\geq 0}$ such that:

- ρ is first-order computable; and
- for every nonempty multiset X, we have $\mathcal{F}_{\mathtt{AGG}}(\rho(X)) > \mathcal{F}_{\mathtt{AGG}}(\emptyset)$.

□

Example 3.5.1. A separator for $\mathcal{F}_{\mathtt{SUM}}$ is the injective function defined by $\rho(x) = x + 1$. Obviously, if X is a nonempty multiset over $\mathbb{Q}_{\geq 0}$, then we have $\mathcal{F}_{\mathtt{SUM}}(\rho(X)) > 0 = \mathcal{F}_{\mathtt{SUM}}(\emptyset)$. ◁

Example 3.5.2. Let $\mathcal{F}_{\mathtt{ODD}}$ be the aggregate operator such that $\mathcal{F}_{\mathtt{ODD}}(X) = 1$ if $|X|$ is odd, and $\mathcal{F}_{\mathtt{ODD}}(X) = 0$ otherwise. Clearly, $\mathcal{F}_{\mathtt{ODD}}(\emptyset) = 0$. We argue that $\mathcal{F}_{\mathtt{ODD}}$ has no separator. To this end, let X be a nonempty multiset such that $\mathcal{F}_{\mathtt{ODD}}(X) = 0$. Then, for every injection ρ, we have $\mathcal{F}_{\mathtt{ODD}}(\rho(X)) = 0$. ◁

Example 3.5.3. Define a *nonGoldbach number* as an even natural number greater than 3 that is not the sum of two prime numbers. Define the aggregate operator $\mathcal{F}_{\mathtt{GOLDBACH}}$ as follows:

- $\mathcal{F}_{\mathtt{GOLDBACH}}(\emptyset) = 0$;

- $\mathcal{F}_{\texttt{GOLDBACH}}(\{\{a\}\}) = \begin{cases} a & \text{if } a \text{ is greater than some nonGoldbach number;} \\ 0 & \text{otherwise.} \end{cases}$
- for $n \geq 1$, $\mathcal{F}_{\texttt{GOLDBACH}}(\{\{a_1, \dots, a_n\}\}) = \sum_{i=1}^{n} \mathcal{F}_{\texttt{GOLDBACH}}\{\{a_i\}\}$.

Clearly, $\mathcal{F}_{\texttt{GOLDBACH}}$ is computable, monotone, and associative. However, it is not known whether $\mathcal{F}_{\texttt{GOLDBACH}}$ has a separator, because it is not known whether nonGoldbach numbers exist. $\triangleleft$

Theorem 3.5.1. *Let $g()$ be a numerical query in* $\mathsf{AGGR[sjfBCQ]}$ *with head* $\texttt{AGG}(r)$*, and let* $q = \mathsf{BCQ}(g())$. *There is a first-order reduction from the problem* $\mathsf{CERTAINTY}(q)$ *to* $\mathsf{GLB\text{-}CQA}(g())$ *if one of the following conditions holds:*

(a) $\mathcal{F}_{\texttt{AGG}}(\emptyset)$ *is undefined;*

(b) $\mathcal{F}_{\texttt{AGG}}(\emptyset)$ *is defined, r is a variable, and* $\mathcal{F}_{\texttt{AGG}}$ *has a separator; or*

(c) $\mathcal{F}_{\texttt{AGG}}(\emptyset)$ *is defined, r is a constant, and for every positive integer n, we have* $\mathcal{F}_{\texttt{AGG}}(\{\{\overbrace{r, r, \dots, r}^{n \text{ times}}\}\}) > \mathcal{F}_{\texttt{AGG}}(\emptyset)$.

Proof. Let $\mathbf{db}$ be a database instance that is input to $\mathsf{CERTAINTY}(q)$. We distinguish the three possibilities:

Case that (a) holds. Then, $\mathbf{db}$ is a "yes"-instance of $\mathsf{CERTAINTY}(q)$ if and only if $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}} \neq \bot$.

Case that (b) holds. We can assume a separator ρ for $\mathcal{F}_{\texttt{AGG}}$. Let $\rho(\mathbf{db}, q, r)$ be the $\subseteq$-minimal database instance such that for every fact $R(a_1, \dots, a_n)$ of $\mathbf{db}$, if $R(t_1, \dots, t_n)$ is the (unique) R-atom of q, then $\rho(\mathbf{db}, q, r)$ contains $R(b_1, \dots, b_n)$, where for $i \in \{1, \dots, n\}$,

$$b_i = \begin{cases} \rho(a_i) & \text{if } t_i = r\text{; and} \\ a_i & \text{otherwise.} \end{cases}$$

Informally, $\rho(\mathbf{db}, q, r)$ is obtained from $\mathbf{db}$ by applying ρ to each rational number in every r-column, while leaving other columns unchanged. It is easily seen that for every database instance $\mathbf{db}$, we have that $\mathbf{r}$ is a repair of $\mathbf{db}$ if and only if $\rho(\mathbf{r}, q, r)$ is a repair of $\rho(\mathbf{db}, q, r)$. Since ρ is first-order computable, we have that $\rho(\mathbf{db}, q, r)$ is first-order computable. It is easily verified that $\mathbf{db}$ is a "yes"-instance of $\mathsf{CERTAINTY}(q)$ if and only if $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\rho(\mathbf{db}, q, r)} > \mathcal{F}_{\texttt{AGG}}(\emptyset)$.

Case that (c) holds. Then, **db** is a "yes"-instance of $\mathsf{CERTAINTY}(q)$ if and only if $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}} > \mathcal{F}_{\mathtt{AGG}}(\emptyset)$.

This concludes the proof of Theorem 3.5.1. □

The following two theorems state inexpressibility results for $\mathsf{GLB\text{-}CQA}(g())$ and $\mathsf{LUB\text{-}CQA}(g())$ in $\mathsf{AGGR[FOL]}$.

Theorem 3.5.2. *Let $g()$ be a numerical query in $\mathsf{AGGR[sjfBCQ]}$, and let $q = \mathsf{BCQ}(g())$. If the attack graph of q is cyclic, and one of the conditions (a), (b), (c) in Theorem 3.5.1 holds, then $\mathsf{GLB\text{-}CQA}(g())$ is not expressible in $\mathsf{AGGR[FOL]}$.*

Proof. Assume that the attack graph of q is cyclic. Then, $\mathsf{CERTAINTY}(q)$ is known to be L-hard under first-order reductions (Koutris & Wijsen, 2017). By Theorem 3.5.1, $\mathsf{GLB\text{-}CQA}(g())$ is L-hard under first-order reductions, and therefore is not Hanf-local. By (Libkin, 2004, Corollary 8.26 and Exercise 8.16), every query in $\mathsf{AGGR[FOL]}$ is Hanf-local and, thus, $\mathsf{GLB\text{-}CQA}(g())$ is not expressible in $\mathsf{AGGR[FOL]}$. □

Theorem 3.5.3. *Let $g()$ be a numerical query in $\mathsf{AGGR[sjfBCQ]}$, and let $q = \mathsf{BCQ}(g())$. If the attack graph of q is cyclic, and condition (a) in Theorem 3.5.1 holds, then $\mathsf{LUB\text{-}CQA}(g())$ is not expressible in $\mathsf{AGGR[FOL]}$.*

Proof. The proof is analogous to that of Theorem 3.5.2. □

3.5.2 Application to Monotone Associative Aggregation

The following theorem concerns the existence of separators for aggregate operators that are monotone and associative.

Theorem 3.5.4. *Let $\mathcal{F}_{\mathtt{AGG}}$ be an aggregate operator that is monotone and associative, and such that $\mathcal{F}_{\mathtt{AGG}}(\emptyset)$ is defined. Then the following are equivalent:*

(A) $\mathcal{F}_{\mathtt{AGG}}$ has a separator;

(B) $\mathcal{F}_{\mathtt{AGG}}(X) \neq \mathcal{F}_{\mathtt{AGG}}(\emptyset)$ for some multiset X; and

(C) $\mathcal{F}_{\mathtt{AGG}}(\{\{r\}\}) \neq \mathcal{F}_{\mathtt{AGG}}(\emptyset)$ for some $r \in \mathbb{Q}_{\geq 0}$.

Proof. First we show that condition (A) implies condition (B). Assume that condition (A) holds true. Then, $\mathcal{F}_{\mathtt{AGG}}$ has a separator ρ. Let Y be a nonempty multiset of non-negative rational numbers. By Definition 3.5.1, $\mathcal{F}_{\mathtt{AGG}}(\rho(Y)) >$

$\mathcal{F}_{\mathtt{AGG}}(\emptyset)$, hence $\mathcal{F}_{\mathtt{AGG}}(\rho(Y)) \neq \mathcal{F}_{\mathtt{AGG}}(\emptyset)$. By letting $X = \rho(Y)$, condition (B) holds true.

Next we show that condition (B) implies condition (C). Assume that condition (B) holds true. Then, there is a cardinality-minimal multiset $X = \{\{x_1, \ldots, x_n\}\}$ such that $\mathcal{F}_{\mathtt{AGG}}(X) \neq \mathcal{F}_{\mathtt{AGG}}(\emptyset)$. Let $X' = \{\{x_1, \ldots, x_{n-1}\}\}$. Since X is chosen as cardinality-minimal, we have $\mathcal{F}_{\mathtt{AGG}}(X') = \mathcal{F}_{\mathtt{AGG}}(\emptyset)$. Since $X = X' \uplus \{\{x_n\}\}$, by associativity of $\mathcal{F}_{\mathtt{AGG}}$, $\mathcal{F}_{\mathtt{AGG}}(X) = \mathcal{F}_{\mathtt{AGG}}(\{\{\mathcal{F}_{\mathtt{AGG}}(X'), x_n\}\})$. It follows $\mathcal{F}_{\mathtt{AGG}}(X) = \mathcal{F}_{\mathtt{AGG}}(\{\{\mathcal{F}_{\mathtt{AGG}}(\emptyset), x_n\}\})$. By associativity, $\mathcal{F}_{\mathtt{AGG}}(X) = \mathcal{F}_{\mathtt{AGG}}(\{\{x_n\}\})$. Consequently, $\mathcal{F}_{\mathtt{AGG}}(\{\{x_n\}\}) \neq \mathcal{F}_{\mathtt{AGG}}(\emptyset)$, and thus condition (C) holds true.

Finally, we show that condition (C) implies condition (A). Assume that condition (C) holds true. Then, there is $r \in \mathbb{Q}_{\geq 0}$ such that $\mathcal{F}_{\mathtt{AGG}}(\{\{r\}\}) \neq \mathcal{F}_{\mathtt{AGG}}(\emptyset)$, hence, by monotonicity of $\mathcal{F}_{\mathtt{AGG}}$,

$$\mathcal{F}_{\mathtt{AGG}}(\emptyset) < \mathcal{F}_{\mathtt{AGG}}(\{\{r\}\}). \tag{3.5}$$

Let ρ be the injective function from $\mathbb{Q}_{\geq 0}$ to $\mathbb{Q}_{\geq 0}$ such that $\rho(x) = x + r$. We show next that ρ is a separator for $\mathcal{F}_{\mathtt{AGG}}$. To this end, let X be a nonempty multiset of non-negative rational numbers, and let x be an element of X. Clearly, $r \leq \rho(x) = x + r$, hence, by monotonicity of $\mathcal{F}_{\mathtt{AGG}}$,

$$\mathcal{F}_{\mathtt{AGG}}(\{\{r\}\}) \leq \mathcal{F}_{\mathtt{AGG}}(\{\{\rho(x)\}\}). \tag{3.6}$$

Since $\rho(x)$ belongs to $\rho(X)$, by monotonicity of $\mathcal{F}_{\mathtt{AGG}}$,

$$\mathcal{F}_{\mathtt{AGG}}(\{\{\rho(x)\}\}) \leq \mathcal{F}_{\mathtt{AGG}}(\rho(X)). \tag{3.7}$$

From (3.5), (3.6), and (3.7), it follows that $\mathcal{F}_{\mathtt{AGG}}(\emptyset) < \mathcal{F}_{\mathtt{AGG}}(\{\{\rho(X)\}\})$. Consequently, ρ is a separator for $\mathcal{F}_{\mathtt{AGG}}$, and hence condition (A) holds true. $\square$

The following lemma shows some desirable consequences of the *Nontriviality Condition* in Definition 3.4.1.

Lemma 3.5.5. *Let $g()$ be a query in* $\mathsf{AGGR[sjfBCQ]}$ *with head* $\mathtt{AGG}(r)$, *for an aggregate operator* $\mathcal{F}_{\mathtt{AGG}}$ *that is monotone and associative, and such that* $\mathcal{F}_{\mathtt{AGG}}(\emptyset)$ *is defined. Then,*

- $\mathcal{F}_{\mathtt{AGG}}$ *has a separator; and*
- *if r is a constant, then* $\mathcal{F}_{\mathtt{AGG}}(\{\{\overbrace{r, r, \ldots, r}^{n \text{ times}}\}\}) > \mathcal{F}_{\mathtt{AGG}}(\emptyset)$ *for every positive integer n.*

Proof. By the *Nontriviality Condition* in Definition 3.4.1, there must be a multiset X with $X \neq \emptyset$ such that $\mathcal{F}_{\texttt{AGG}}(X) \neq \mathcal{F}_{\texttt{AGG}}(\emptyset)$. It follows from Theorem 3.5.4 that $\mathcal{F}_{\texttt{AGG}}$ has a separator.

Assume from here on that r is a constant. For every nonnegative integer i, define $B_i := \{\{\overbrace{r, r, \dots, r}^{i \text{ times}}\}\}$. By the *Nontriviality Condition* in Definition 3.4.1, and by the monotonicity of $\mathcal{F}_{\texttt{AGG}}$, there must be a positive integer n such that $\mathcal{F}_{\texttt{AGG}}(\emptyset) < \mathcal{F}_{\texttt{AGG}}(B_n)$. By monotonicity, $\mathcal{F}_{\texttt{AGG}}(\emptyset) \leq \mathcal{F}_{\texttt{AGG}}(B_1) \leq \mathcal{F}_{\texttt{AGG}}(B_2) \leq \cdots$. Hence, there exists $m \in \{1, 2, \dots, n\}$ such that

$$\mathcal{F}_{\texttt{AGG}}(\emptyset) = \mathcal{F}_{\texttt{AGG}}(B_{m-1}) < \mathcal{F}_{\texttt{AGG}}(B_m).$$

Since $\{\{r\}\} = \emptyset \uplus \{\{r\}\}$, by associativity of $\mathcal{F}_{\texttt{AGG}}$,

$$\mathcal{F}_{\texttt{AGG}}(\{\{r\}\}) = \mathcal{F}_{\texttt{AGG}}(\{\{\mathcal{F}_{\texttt{AGG}}(\emptyset), r\}\}). \tag{3.8}$$

Since $B_m = B_{m-1} \uplus \{\{r\}\}$, and by associativity of $\mathcal{F}_{\texttt{AGG}}$,

$$\mathcal{F}_{\texttt{AGG}}(B_m) = \mathcal{F}_{\texttt{AGG}}(\{\{\mathcal{F}_{\texttt{AGG}}(B_{m-1}), r\}\}). \tag{3.9}$$

Since $\mathcal{F}_{\texttt{AGG}}(\emptyset) = \mathcal{F}_{\texttt{AGG}}(B_{m-1})$, it follows from (3.8) and (3.9) that $\mathcal{F}_{\texttt{AGG}}(B_m) = \mathcal{F}_{\texttt{AGG}}(\{\{r\}\})$. Consequently, $\mathcal{F}_{\texttt{AGG}}(\emptyset) < \mathcal{F}_{\texttt{AGG}}(\{\{r\}\})$. By monotonicity of $\mathcal{F}_{\texttt{AGG}}$, we have $\mathcal{F}_{\texttt{AGG}}(\emptyset) < \mathcal{F}_{\texttt{AGG}}(\{\{r\}\}) \leq \mathcal{F}_{\texttt{AGG}}(\{\{r, r\}\}) \leq \cdots$. This concludes the proof. □

Consequently, the following result applies to every numerical $\mathsf{AGGR[sjfBCQ]}$ query using a monotone and associative aggregate operator.

Theorem 3.5.6. *Let $g()$ be a numerical query in* $\mathsf{AGGR[sjfBCQ]}$ *with head* $\texttt{AGG}(r)$*, and let* $q = \mathsf{BCQ}(g())$*, such that* $\mathcal{F}_{\texttt{AGG}}$ *is monotone and associative. If the attack graph of q is cyclic, then* $\mathsf{GLB\text{-}CQA}(g())$ *is not expressible in* $\mathsf{AGGR[FOL]}$.

Proof. Immediate consequence of Theorem 3.5.2 and Lemma 3.5.5. □

CHAPTER 4

Parsimony

In this chapter, inspired by the work of Fuxman (2007), we study the expressibility of $\mathsf{GLB\text{-}CQA}(g())$ and $\mathsf{LUB\text{-}CQA}(g())$ in what can be thought of as a syntactically highly restricted subclass of $\mathsf{AGGR}[\mathsf{FOL}]$, for $\mathtt{AGG}$-queries $g()$ in $\mathsf{AGGR}[\mathsf{sjfBCQ}]$. This subclass is limited in expressive power to applying simple aggregation steps to the results of genuine first-order queries—a method we refer to as *parsimonious aggregation*, or *Fuxman's technique*. We first consider *counting queries*, i.e., numerical queries with head $\mathtt{SUM}(1)$, and then extend our study to other aggregate operators. In this chapter, we also assume that the database instances $\mathbf{db}$ for which $\mathsf{GLB\text{-}CQA}(g())$ and $\mathsf{LUB\text{-}CQA}(g())$ are computed satisfy $\mathbf{db} \models_{\mathsf{cqa}} \mathsf{BCQ}(g())$, which implies that every aggregation is over a non-empty multiset. Under this assumption, it does not matter whether or not $\mathcal{F}_{\mathtt{AGG}}(\emptyset)$ is defined. The motivation for starting with this simplified setting is primarily historical: for many years—until the work of Amezian El Khalfioui & Wijsen (2023)—it remained an open question which aggregation queries could be handled by Fuxman's technique. This question is resolved in the current chapter. Once this question is resolved, our attention will turn to the expressibility of $\mathsf{GLB\text{-}CQA}(g())$ and $\mathsf{LUB\text{-}CQA}(g())$ in plain $\mathsf{AGGR}[\mathsf{FOL}]$—without syntactic restrictions—over database instances whose repairs may falsify $\mathsf{BCQ}(g())$. These studies are presented in subsequent chapters, which can be read independently of the current one.

In his PhD thesis, Fuxman (2007) showed that for some counting queries $g()$ and database instances $\mathbf{db}$ such that $\mathbf{db} \models_{\mathsf{cqa}} \mathsf{BCQ}(g())$, $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}}$ and $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}}$ can be computed by executing FOL queries, followed by simple counting steps. To illustrate his approach, consider the database in Fig. 4.1, which stores information about employees and their department buildings. Consider the numerical query $g() := \mathtt{SUM}(1) \leftarrow E(\underline{x}, y, z), D(\underline{z}, \text{“A”})$ and the following query in SQL:

```
SELECT   COUNT(DISTINCT Emp) AS CNT
FROM     E, D
WHERE    E.Dept = D.Dept AND Building = 'A'
```

On our example database **db** of Fig. 4.1, this SQL query returns 3, which also turns out to be the answer to $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}}$. Importantly, it can be shown that this is not by accident: on *every* database instance, the latter SQL query will return the correct answer to $\mathsf{LUB\text{-}CQA}(g())$. Note that the latter SQL query uses `COUNT(DISTINCT Emp)`, which means that duplicates are removed, which is a standard practice in relational algebra.

We now explain how to obtain $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}}$ for our example query. To this end, consider the following query:

```
SELECT Emp
FROM   E, D
WHERE  E.Dept = D.Dept AND Building = 'A'
```

For our example database, the *consistent answers* to this SQL query are in the following table, which we call C:

C	*Emp*
	Suzy

Note that Grety is not a consistent answer because Grety is not an answer in some repair. From (Koutris & Wijsen, 2017), it follows that computing the consistent answers to the latter SQL query is in FO (i.e., the class of problems that can be solved by a first-order query), using a technique known as *consistent first-order rewriting.* The lower bound 1 is now found by executing the following query on C (and, again, this is not by accident):

```
SELECT   COUNT(DISTINCT Emp) AS CNT
FROM     C
```

Since C can be expressed in SQL, we can actually construct a single SQL query that computes $\mathsf{GLB\text{-}CQA}(g())$.

In general, a counting query $g()$ is said to *admit parsimonious counting* if it is possible to evaluate $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}}$ and $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}}$ as previously described, for every database instance **db** such that $\mathbf{db} \models_{\mathsf{cqa}} \mathsf{BCQ}(g())$. Thus, our example showed that $g() := \mathtt{SUM}(1) \leftarrow E(\underline{x}, y, z), D(\underline{z}, \text{“A”})$ admits parsimonious counting. A formal definition of parsimonious counting will be

E

Emp	*Gender*	*Dept*
Suzy	F	HR
Anny	F	HR
Anny	F	IT
Grety	F	IT
Lucy	F	MIS

D

Dept	*Building*
HR	A
IT	A
IT	B
MIS	B

Figure 4.1: Example database. Primary keys are underlined.

given later on (Definition 4.1.1). In this introduction, we content ourselves by saying that parsimonious counting, if possible, computes GLB-CQA($g()$) and LUB-CQA($g()$) by executing two first-order queries (one for lower bounds, and one for upper bounds), followed by simple counting steps.

In his doctoral dissertation, Fuxman (2007) defined a class of self-join-free conjunctive queries, called Cforest, and, among other results, proved the following.

Theorem 4.0.1 (Fuxman (2007)). *Let $g()$ be a counting query in the class* AGGR[sjfBCQ]. *If* BCQ($g()$) $\in$ Cforest*, then $g()$ admits parsimonious counting.*

The class Cforest has been used in several studies on consistent query answering. It was an open question whether Cforest contains *all* self-join-free Boolean conjunctive queries $\exists \vec{y}\,(q(\vec{y}))$ such that $g() := \texttt{SUM}(1) \leftarrow q(\vec{y})$ admits parsimonious counting. We will answer this question negatively in Section 4.1.4. More fundamentally, we introduce a new syntactic class, called Cparsimony, which includes Cforest, and contains *all* (and only) the self-join-free Boolean conjunctive queries $\exists \vec{y}\,(q(\vec{y}))$ such that $g() := \texttt{SUM}(1) \leftarrow q(\vec{y})$ admits parsimonious counting. That is, we prove the following theorem.

Theorem 4.0.2 (Main theorem about parsimonious counting). *For every counting query $g()$, it holds that $g()$ admits parsimonious counting if and only if* BCQ($g()$) *is in* Cparsimony.

The chapter is organized as follows. First, Section 4.1 introduces the semantic notion of parsimonious counting, our new syntactic class of queries, called Cparsimony, and shows that for every counting query $g()$, BCQ($g()$) $\in$ Cparsimony is a sufficient and necessary condition for $g()$ to admit parsimonious counting. Moreover, we also show that Cforest is strictly included in Cparsimony, and we illustrate how parsimonious counting can be expressed in AGGR[FOL]. Section 4.2 then extends the notion of parsimonious counting to numerical queries with a head other than $\texttt{SUM}(1)$, which we refer to as *parsimonious aggregation.* We show that for every numerical query $g()$ with a monotone aggregation operator, $\mathsf{BCQ}(g()) \in \mathsf{Cparsimony}$ is a sufficient (but not necessary) condition for $g()$ to admit parsimonious aggregation. We also illustrate how parsimonious aggregation is expressible in $\mathsf{AGGR[FOL]}$.

4.1. Parsimonious Counting

Let $g()$ be a counting query in $\mathsf{AGGR[sjfBCQ]}$. The following proposition states that computing $\mathsf{LUB\text{-}CQA}(g())$ can be NP-hard, even if $\mathsf{BCQ}(g())$ has a consistent first-order rewriting.

Proposition 4.1.1. *There is a counting query $g()$ such that $\mathsf{BCQ}(g())$ has a consistent first-order rewriting and $\mathsf{LUB\text{-}CQA}(g())$ is NP-hard to compute.*

Proof. The following problem is NP-complete (Garey & Johnson, 1979).

3-DIMENSIONAL MATCHING (3DM)

INSTANCE: A set $M \subseteq A_1 \times A_2 \times A_3$, where A_1, A_2, A_3 are disjoint sets having the same number n of elements.

QUESTION: Does M contain a matching, that is, a subset $M' \subseteq M$ such that $|M'| = n$ and no two elements of M' agree in any coordinate?

Consider the numerical query

$$g() := \mathtt{SUM}(1) \leftarrow \bigcup_{i=1}^{3} \{R_i(\underline{x_i}, y), S_i(\underline{x_i}, y)\}\,.$$

Let $q = \mathsf{BCQ}(g())$. The edge-set of q's attack graph is empty. Therefore, q's attack graph is acyclic. Observe that the S_i-atoms serve to render the attack graph acyclic. By Theorem 2.0.1, q has a consistent first-order rewriting. Let $M \subseteq A_1 \times A_2 \times A_3$ be an instance of $\mathsf{3DM}$. Let $\mathbf{db}_M$ be the database instance that includes $\bigcup_{i=1}^{3}\{R_i(\underline{a_i}, a_1a_2a_3), S_i(\underline{a_i}, a_1a_2a_3)\}$ for every $a_1a_2a_3$ in M. Clearly, $\mathbf{db}_M$ is first-order computable from M.

We now show the following property: M has a matching if and only if $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}_M} = n$. Before delving into the proof, we provide an example.

Example 4.1.1. Let $A_1 = \{a, b\}$, $A_2 = \{d, e\}$, $A_3 = \{f, g\}$, and $M = \{adf, aeg, beg\}$. Thus, $n = |A_1| = 2$. Then, we construct relations as follows.

$\mathbf{db}_M$:

$R_1 = S_1$	$\underline{x_1}$	y	
	a	adf	$*$
	a	aeg	
	b	beg	$*$

$R_2 = S_2$	$\underline{x_2}$	y	
	d	adf	$*$
	e	aeg	
	e	beg	$*$

$R_3 = S_3$	$\underline{x_3}$	y	
	f	adf	$*$
	g	aeg	
	g	beg	$*$

It can be verified that $[\![\textsf{LUB-CQA}(g())]\!]^{\mathbf{db}_M} = 2$. The tuples marked with $*$ form a repair $\mathbf{r}$ on which $[\![g()]\!]^{\mathbf{r}} = 2$. In particular, there are two embeddings in $\mathbf{r}$ mapping (x_1, x_2, x_3, y) to (a, d, f, adf) and (b, e, g, beg). This answer corresponds to the 3-dimensional matching $\{adf, beg\}$. $\lhd$

$\boxed{\Longrightarrow}$ Assume M' is a matching of M. Let $\mathbf{r}$ be a database instance that includes, for every $a_1a_2a_3 \in M'$, the set $\bigcup_{i=1}^{3}\{R_i(\underline{a_i}, a_1a_2a_3),\ S_i(\underline{a_i}, a_1a_2a_3)\}$. Since no two elements of M' agree on any coordinate, $\mathbf{r}$ is consistent. Moreover, since $n = |M'| = |A_1| = |A_2| = |A_3|$, $\mathbf{r}$ contains a tuple of every block of $\mathbf{db}_M$. Therefore, $\mathbf{r}$ is a repair of $\mathbf{db}_M$. Clearly, $[\![g()]\!]^{\mathbf{r}} = n$. It is also obvious to see that there is no repair $\mathbf{s}$ of $\mathbf{db}_M$ on which $[\![g()]\!]^{\mathbf{s}} = k$ with $k > n$.

$\boxed{\Longleftarrow}$ Let $\mathbf{r}$ be a repair of $\mathbf{db}_M$ such that $[\![g()]\!]^{\mathbf{r}} = n$. Let θ be the embedding of q in $\mathbf{r}$ mapping (x_1, x_2, x_3, y) to $(a_1, a_2, a_3, b_1b_2b_3)$. For each $i \in \{1, 2, 3\}$, since q contains the atom $R_i(\underline{x_i}, y)$, $\mathbf{r}$ contains $R_i(\underline{a_i}, b_1b_2b_3)$, and hence $a_i = b_i$ by our construction. It follows $b_1b_2b_3 = a_1a_2a_3$.

Let θ_a and θ_b be two embeddings of q in $\mathbf{r}$ mapping, respectively, the sequence (x_1, x_2, x_3, y) to $(a_1, a_2, a_3, a_1a_2a_3)$ and $(b_1, b_2, b_3, b_1b_2b_3)$. Since q contains $R_i(\underline{x_i}, y)$, $\mathbf{r}$ contains $R_i(\underline{a_i}, a_1a_2a_3)$ and $R_i(\underline{b_i}, b_1b_2b_3)$. If $a_i = b_i$, then, since $\mathbf{r}$ is consistent, we have $a_1a_2a_3 = b_1b_2b_3$. Consequently, no two distinct embeddings of q in $\mathbf{r}$ agree on any coordinate among x_1, x_2, and x_3. Since $[\![g()]\!]^{\mathbf{r}} = n$, it follows that the set containing $\theta(y)$ for every embedding θ of q in $\mathbf{r}$ is a matching of M' of size n. This concludes the proof. $\square$

Note that the foregoing proof carries over from 3-DIMENSIONAL MATCHING to 2-DIMENSIONAL MATCHING. That is, if

$$g() := \texttt{SUM}(1) \leftarrow R_1(\underline{x_1}, y), S_1(\underline{x_1}, y), R_2(\underline{x_2}, y), S_2(\underline{x_2}, y),$$

then $\textsf{BCQ}(g())$ has a consistent first-order rewriting, whereas $\textsf{LUB-CQA}(g())$ is as hard as 2-DIMENSIONAL MATCHING.

The following definition introduces the semantic notion of *parsimonious counting*, which was illustrated by the running example that introduced this chapter. Informally, for a counting query $g()$ in $\textsf{AGGR[sjfBCQ]}$ that admits parsimonious counting, it will be the case that on every database instance $\mathbf{db}$ such that $\mathbf{db} \models_{\textsf{cqa}} \textsf{BCQ}(g())$, $[\![\textsf{GLB-CQA}(g())]\!]^{\mathbf{db}}$ and $[\![\textsf{LUB-CQA}(g())]\!]^{\mathbf{db}}$ can be computed by a first-order query followed by a simple counting step.

Definition 4.1.1 (Parsimonious counting)**.** Let $g()$ be a counting query in $\textsf{AGGR[sjfBCQ]}$. Let $q = \textsf{BCQ}(g())$. Let $\vec{x}$ be a (possibly empty) sequence of distinct bound variables of q. We say that $g()$ *admits parsimonious counting on* $\vec{x}$ if the following hold (let $q'(\vec{x}) = \nexists\vec{x}\,[q]$):

(A) q has a consistent first-order rewriting;

(B) $q'(\vec{x})$ has a consistent first-order rewriting (call it $\varphi(\vec{x})$); and

(C) for every database instance **db** such that $\mathbf{db} \models_{\mathsf{cqa}} q$, the following conditions (a) and (b) are equivalent:

  (a) $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}} = m$ and $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}} = n$.

  (b) Both the following hold:

    (i) m is the number of distinct tuples $\vec{d}$, of arity $\vec{x}$, such that $\mathbf{db} \models \varphi(\vec{d})$; and

    (ii) n is the number of distinct tuples $\vec{d}$ such that $\mathbf{db} \models q'(\vec{d})$.

We say that $g()$ *admits parsimonious counting* if it admits parsimonious counting on some sequence $\vec{x}$ of bound variables. □

Significantly, since Definition 4.1.1 contains a condition that must hold for every database instance **db**, it does not give us an efficient procedure for deciding whether a given $g() \in \mathsf{AGGR[sjfBCQ]}$ admits parsimonious counting.

From the proof of Proposition 4.1.1 and the paragraph after that proof, it follows that under standard complexity assumptions, for $k \geq 2$,

$$g_k() := \texttt{SUM}(1) \leftarrow \bigcup_{i=1}^{k} \{R_i(\underline{x_i}, y), S_i(\underline{x_i}, y)\}$$

does not admit parsimonious counting, even though $\mathsf{BCQ}(g_k())$ has a consistent first-order rewriting.

4.1.1 The Class Cparsimony

The notion of parsimonious counting is a semantic property defined for counting queries. A natural question is to syntactically characterize the class of counting queries that admit parsimonious counting. In this section, we will answer this question for counting queries in $\mathsf{AGGR[sjfBCQ]}$. This is the best we can currently hope for, because consistent query answering for primary keys and conjunctive queries with self-joins is a notorious open problem for which no tools are known (e.g., attack graphs are not helpful in the presence of self-joins). We now define our new syntactic class $\mathsf{Cparsimony}$, which uses the following notion of *frozen variable.*

Definition 4.1.2 (Frozen variable)**.** Let q be a self-join-free conjunctive query.

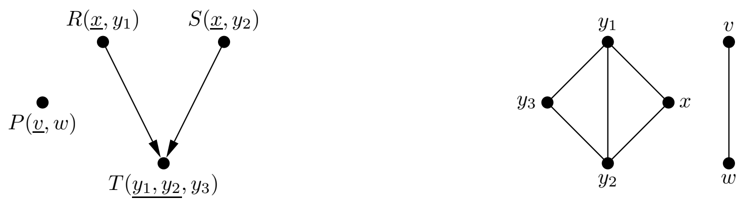

Figure 4.2: Attack graph *(left)* and Gaifman graph *(right)* of $q = \exists x \exists y_1 \exists y_2 \exists y_3 \exists v \exists w (R(\underline{x}, y_1) \wedge S(\underline{x}, y_2) \wedge T(\underline{y_1, y_2}, y_3) \wedge P(\underline{v}, w))$.

We say that a bound variable y of q is *frozen in q* if there exists a sequential proof of $\mathcal{K}(q) \models \emptyset \rightarrow y$ such that $F \overset{q}{\not\rightsquigarrow} y$ for every atom F that occurs in the sequential proof. We write $\mathsf{frozen}(q)$ for the set of all bound variables of $\mathsf{vars}(q)$ that are frozen in q. A bound variable that is not frozen in q is called *nonfrozen in q*. □

Example 4.1.2. Let $q = \exists x(R(\underline{c}, x) \wedge S(\underline{c}, x))$. We have $R \overset{q}{\not\rightsquigarrow} x$. Therefore, $(R(\underline{c}, x))$ is a sequential proof of $\mathcal{K}(q) \models \emptyset \rightarrow x$ that uses no atom attacking x. Hence, x is frozen. ◁

Definition 4.1.3 (The class $\mathsf{Cparsimony}$)**.** We define $\mathsf{Cparsimony}$ as the set of self-join-free Boolean conjunctive queries q satisfying the following conditions:

(I) the attack graph of q is acyclic; and

(II) there is a tuple $\vec{x}$ of bound variables of q such that:

  (1) $\mathcal{K}(q) \models \vec{x} \rightarrow \mathsf{vars}(q)$; and
  (2) for every atom F in q, every (possibly empty) path in the Gaifman graph of q between a variable of $\mathsf{notKey}(F)$ and a variable of $\vec{x}$ uses a variable in $\mathsf{Key}(F) \cup \mathsf{frozen}(q)$.

We will say that such an $\vec{x}$ is an *id-set* for q. We will say that an id-set $\vec{x}$ is *minimal* if any sequence obtained from $\vec{x}$ by omitting one or more variables is no longer an id-set. □

Informally, id-sets $\vec{x}$ will play the role of $\vec{x}$ in Definition 4.1.1: they identify the values that have to be counted to obtain $\mathsf{GLB\text{-}CQA}(g())$ and $\mathsf{LUB\text{-}CQA}(g())$.

We now illustrate Definition 4.1.3 by some examples. Then Proposition 4.1.2 implies that every query q in $\mathsf{Cparsimony}$ has a unique minimal id-set that can be easily constructed from q's attack graph.

Example 4.1.3. In the paragraph following the proof of Proposition 4.1.1, we considered the query $q = \exists x_1 \exists x_2 \exists y (\bigwedge_{i=1}^{2} (R_i(\underline{x_i}, y) \wedge S_i(\underline{x_i}, y)))$. The edge-set of q's attack graph is empty. No variable is frozen. According to condition (1) in Definition 4.1.3, every id-set (if any) must contain x_1. However, no id-set can contain x_1, because for the atom $R_2(\underline{x_2}, y)$, the edge $\{y, x_1\}$ in the Gaifman graph of q is a path between a variable of $\mathsf{notKey}(R_2)$ and x_1 that uses no variable of $\mathsf{Key}(R_2)$. We conclude that q is not in $\mathsf{Cparsimony}$. $\triangleleft$

Example 4.1.4. The query $q = \exists x \exists y \exists v (R(\underline{x}, y) \wedge S(\underline{y}, v) \wedge T(\underline{v}, y) \wedge P_1(\underline{c}, y) \wedge P_2(\underline{c}, y))$ belongs to $\mathsf{Cparsimony}$. The attack graph of q has a single attack from S to T. The Gaifman graph of q has two undirected edges: $\{x, y\}$ and $\{y, v\}$. The variable y is frozen, because $(P_1(\underline{c}, y))$ is a sequential proof of $\mathcal{K}(q) \models \emptyset \rightarrow y$ and $P_1 \overset{q}{\not\rightsquigarrow} y$.

It can be verified that (x) is an id-set. Note that (y, x) is a path in the Gaifman graph of q between $y \in \mathsf{notKey}(T)$ and x that uses no variable of $\mathsf{Key}(T) = \{v\}$. However, that path uses the frozen variable y. $\triangleleft$

Example 4.1.5. Let q be the following self-join-free Boolean conjunctive query

$$q = \exists x \exists y_1 \exists y_2 \exists y_3 \exists v \exists w (R(\underline{x}, y_1) \wedge S(\underline{x}, y_2) \wedge T(\underline{y_1, y_2}, y_3) \wedge P(\underline{v}, w)).$$

The attack graph and the Gaifman graph of q are shown in Fig. 4.2. We now argue that q is in $\mathsf{Cparsimony}$. First, the attack graph of q is acyclic. We next argue that (xv) is an id-set for q. Condition (1) in Definition 4.1.3 is obviously satisfied for $\vec{x} = (xv)$. It is easily verified that condition (2) is also verified. In particular, for the atom $T(\underline{y_1, y_2}, y_3)$, every path between y_3 and x uses either y_1 or y_2. $\triangleleft$

Example 4.1.6. Let $q = \exists x \exists y (R_1(\underline{x}, y) \wedge R_2(\underline{x}, y) \wedge S_1(\underline{y}, x) \wedge S_2(\underline{y}, x))$. The attack graph of q contains no edges and, thus, is acyclic. It can be verified that no variable is frozen. We claim that q is not in $\mathsf{Cparsimony}$, because it has no id-set. Indeed, from condition (1) in Definition 4.1.3, it follows that every id-set must contain either x or y (or both). For the atom $S_1(\underline{y}, x)$, the empty path is a path between a variable in $\mathsf{notKey}(S_1)$ to x that uses no variable in $\mathsf{Key}(S_1)$. It follows by condition (2) that no id-set can contain x. From $R_2(\underline{x}, y)$, by similar reasoning, we conclude that no id-set can contain y. It follows that q has no id-set. $\triangleleft$

Proposition 4.1.2. *Let q in $\mathsf{Cparsimony}$, and let $\vec{x}$ be a minimal id-set for it. Let $N = \bigcup\{\mathsf{notKey}(R) \mid R \in q\}$. Let V be a $\subseteq$-minimal subset of $\mathsf{vars}(q)$ that includes, for every unattacked atom F of q, every bound variable of $\mathsf{Key}(F) \setminus N$. Then,* *(A)* $V = \mathsf{vars}(\vec{x})$*; and*

(B) whenever F, G *are unattacked atoms that are weakly connected in* q*'s attack graph,* $\mathsf{Key}(F) \cap \vec{x} = \mathsf{Key}(G) \cap \vec{x}$.

Example 4.1.7. Let $q = \exists x \exists y (R(\underline{x}, y) \wedge S(\underline{x}, y) \wedge T(\underline{y}))$. The attack graph of q contains no edges, and hence has three weak components. If we construct V as in the statement of Proposition 4.1.2, we first compute $N = \{y\}$, and then $V = \{x\}$. $\triangleleft$

Example 4.1.8. Let $q = \exists x \exists y \exists z_1 \exists z_2 (R(\underline{x}, y, z_1) \wedge S(\underline{x, y}, z_2) \wedge T(\underline{z_1, z_2}) \wedge P(\underline{x}, y))$. The attack graph of q contains two edges: $R(\underline{x}, y, z_1)$ and $S(\underline{x, y}, z_2)$ both attack $T(\underline{z_1, z_2})$. Thus, the attack graph of q has two weak components. If we construct V as in the statement of Proposition 4.1.2, we first compute $N = \{y, z_1, z_2\}$, and then $V = \{x\}$. $\triangleleft$

The proof for Proposition 4.1.2 is provided in Section B.1 of Appendix B. The following proposition settles the complexity of checking whether a query belongs to $\mathsf{Cparsimony}$.

Proposition 4.1.3. *The following decision problem is in quadratic time: Given a self-join-free Boolean conjunctive query* q*, decide whether or not* q *belongs to* $\mathsf{Cparsimony}$.

Proof. Let q be a self-join-free Boolean conjunctive query. It is possible, in quadratic time in the size of q, to compute the attack graph of q (Wijsen, 2012), test whether condition (I) in Definition 4.1.3 is satisfied, and construct V as in the statement of Proposition 4.1.2. By adding to q, for every variable x, a fresh atom $P_x(\underline{x})$, the attack graph allows determining the set n_x of all atoms not attacking x. Membership of x in $\mathsf{frozen}(q)$ is determined by testing whether $\mathcal{K}(n_x) \models \emptyset \rightarrow x$, which is in linear time in the size of n_x. If condition (I) is not satisfied, answer "no"; otherwise proceed as follows. For every atom F in q, test whether, in q's Gaifman graph, some variable of V is reachable from some variable in $\mathsf{notKey}(F)$ without using variables from $\mathsf{Key}(F) \cup \mathsf{frozen}(q)$. This can be done in quadratic time. If any such reachability test succeeds, then it is correct to answer "no", because q has no id-set; otherwise answer "yes". The correctness follows from Proposition 4.1.2: if q has an id-set, then V is an id-set. Therefore, if V falsifies condition (2) in Definition 4.1.3, then V is not an id-set, and hence q has no id-set. Finally, we note that, by construction, V will satisfy condition (1) in Definition 4.1.3. $\square$

4.1.2 $\mathsf{Cparsimony}$ Admits Parsimonious Counting

In this section, we show the if-direction of Theorem 4.0.2, which is the following theorem.

Theorem 4.1.4. *Let $q \in \mathsf{Cparsimony}$ and let $g()$ be a counting query such that $\mathsf{BCQ}(g()) = q$. Then $g()$ admits parsimonious counting.*

We use a number of helping lemmas and constructs. The following lemma says that if $\vec{x}$ is an id-set for a query q in $\mathsf{Cparsimony}$, then for a consistent database $\mathbf{db}$, if $g()$ is the counting query such that $\mathsf{BCQ}(g()) = q$, the answers to $[\![g()]\!]^{\mathbf{db}}$ can be obtained by counting the number of distinct $\vec{x}$-values, while variables not in $\vec{x}$ can be ignored.

Lemma 4.1.5. *Let q be a sjfBCQ in $\mathsf{Cparsimony}$ and $\vec{x}$ be an id-set for q. Let $\mathbf{db}$ be a consistent database instance. Let θ_1, θ_2 be two embeddings of q in $\mathbf{db}$. If $\theta_1(\vec{x}) = \theta_2(\vec{x})$, then $\theta_1 = \theta_2$.*

Proof. From condition (1) in Definition 4.1.3, we have that $\mathcal{K}(q) \models \vec{x} \rightarrow \mathsf{vars}(q)$. Since $\mathbf{db}$ is a consistent database instance and $\theta_1(\vec{x}) = \theta_2(\vec{x})$, it follows that $\theta_1 = \theta_2$. $\square$

We now present the notion of *optimistic repair*, which was originally introduced by Fuxman (2007). Informally, a repair $\mathbf{r}$ of a database $\mathbf{db}$ is an optimistic repair with respect to a conjunctive query $q(\vec{x})$ if every tuple that is an answer to $q(\vec{x})$ on $\mathbf{db}$ is also an answer to $q(\vec{x})$ on $\mathbf{r}$. The converse obviously holds true because conjunctive queries are monotone and repairs are subsets of the original database instance.

Definition 4.1.4 (Optimistic repair)**.** Let $q(\vec{x})$ be a conjunctive query. Let $\mathbf{db}$ be a database instance. We say that a repair $\mathbf{r}$ of $\mathbf{db}$ is an *optimistic repair* with respect to $q(\vec{x})$ if for every tuple $\vec{a}$ of constants, of arity $|\vec{x}|$, $\mathbf{db} \models q(\vec{a})$ implies $\mathbf{r} \models q(\vec{a})$ (the converse implication is obviously true). $\square$

The following lemmas gives a sufficient condition for the existence of optimistic repairs.

Lemma 4.1.6. *Let q be a query in $\mathsf{Cparsimony}$, and let $\vec{x}$ be an id-set for q. Let $q'(\vec{x})$ be the query $\nexists\vec{x}\,[q]$. Let $\mathbf{db}$ be a database instance such that $\mathbf{db} \models_{\mathsf{cqa}} q$. Let $\{\vec{d_1}, \ldots, \vec{d_\ell}\}$ be a subset of $\mathsf{adom}(\mathbf{db})^{|\vec{x}|}$ such that $\mathbf{db} \models q'(\vec{d_i})$ for every $i \in \{1, \ldots, \ell\}$. Let r be a non-negative rational number or a numerical variable in $\mathsf{vars}(q)$. For each $i \in \{1, \ldots, n\}$, there exists an embedding θ_i of q in $\mathbf{db}$ satisfying the following conditions:*

(a) $\theta_i(\vec{x}) = \vec{d_i}$;

(b) for every $j \in \{1, \ldots, n\}$, *for every atom* F *in* q, *if* $\theta_i(F)$ *and* $\theta_j(F)$ *are key-equal, then they are equal; and*

(c) for every embedding θ *of* q *in* $\mathbf{db}$ *such that* $\theta(\vec{x}) = \vec{d_i}$, $\theta(r) \leq \theta_i(r)$.

The proof of Lemma 4.1.6 is provided in Section B.2 of Appendix B. Note that condition (c) of Lemma 4.1.6 is not used in this proof but will be useful in later sections. Indeed, for counting queries, we will always use $r = 1$.

Lemma 4.1.7. *Let* q *be a query in* $\mathsf{Cparsimony}$, *and let* $\vec{x}$ *be an id-set for* q. *Let* $q'(\vec{x})$ *be the query* $\nexists\vec{x}\,[q]$. *Let* $\mathbf{db}$ *be a database instance such that* $\mathbf{db} \models_{\mathsf{cqa}} q$. *Then,* $\mathbf{db}$ *has an optimistic repair with respect to* $q'(\vec{x})$.

Proof. Let $\{\vec{d_1}, \ldots, \vec{d_n}\}$ be a maximal subset of $\mathsf{adom}(\mathbf{db})^{|\vec{x}|}$ such that $\mathbf{db} \models q'(\vec{d_i})$ for every $i \in \{1, \ldots, n\}$. By Lemma 4.1.6, for each $i \in \{1, \ldots, n\}$, there exists an embedding θ_i of q in $\mathbf{db}$ satisfying the following conditions:

(a) $\theta_i(\vec{x}) = \vec{d_i}$; and

(b) for every $j \in \{1, \ldots, n\}$, for every atom F in q, if $\theta_i(F)$ and $\theta_j(F)$ are key-equal, then $\theta_i(F) = \theta_j(F)$.

From the last property, it follows that there is a repair $\mathbf{r}$ of $\mathbf{db}$ such that for every $i \in \{1, \ldots, n\}$, θ_i is an embedding of q in $\mathbf{r}$. Since $\mathbf{r} \models q'(\vec{d_i})$ for every $i \in \{1, \ldots, n\}$, the repair $\mathbf{r}$ is an optimistic repair of $\mathbf{db}$ with respect to $q'(\vec{x})$. $\square$

We now present the notion of *pessimistic repair*, also borrowed from (Fuxman, 2007). Informally, a repair of a database $\mathbf{db}$ is a pessimistic repair with respect to a conjunctive query $q(\vec{x})$ if every answer to $q(\vec{x})$ on $\mathbf{r}$ is a consistent answer to $q(\vec{x})$ on $\mathbf{db}$. The converse trivially holds true.

Definition 4.1.5 (Pessimistic repair)**.** Let $q(\vec{x})$ be a conjunctive query. Let $\mathbf{db}$ be a database instance. We say that a repair $\mathbf{r}$ of $\mathbf{db}$ is a pessimistic repair with respect to $q(\vec{x})$ if for every tuple $\vec{a}$ of constants, of arity $|\vec{x}|$, if $\mathbf{r} \models q(\vec{a})$, then $\mathbf{db} \models_{\mathsf{cqa}} q(\vec{a})$. $\square$

The following lemma gives a sufficient condition for the existence of pessimistic repairs.

Lemma 4.1.8. *Let* q *be a query in* $\mathsf{Cparsimony}$, *and let* $\vec{x}$ *be an id-set for* q. *Let* $q'(\vec{x})$ *be the query* $\nexists\vec{x}\,[q]$. *Let* $\mathbf{db}$ *be a database instance. Then,* $\mathbf{db}$ *has a pessimistic repair with respect to* $q'(\vec{x})$.

Proof. From (Koutris & Wijsen, 2017), it follows that whenever $\vec{x}$ is a sequence of unattacked variables of a self-join-free Boolean conjunctive query, then every database instance **db** has a repair **r** such that for every sequence $\vec{d}$ of constants, of arity $|\vec{x}|$, $\mathbf{r} \models q'(\vec{d})$ implies $\mathbf{db} \models_{\mathsf{cqa}} q'(\vec{d})$. The proof is now straightforward by using that all variables of an id-set are unattacked (by item (b) in Lemma B.1.3, provided in Section B.1 of Appendix B). $\square$

The following example illustrates the preceding constructs and lemmas.

Example 4.1.9. Let $g() := \texttt{SUM}(1) \leftarrow R(\underline{x}, y), S(\underline{x}, y)$. Let $q = \mathsf{BCQ}(g())$. Let **db** be the following database instance:

R	$\underline{x}$	y
	a_1	b_1
	a_1	b_2
	a_2	b_2

S	$\underline{x}$	y
	a_1	b_1
	a_2	b_2

Clearly, **db** has two repairs, which are $\mathbf{r}_1 := \mathbf{db} \setminus \{R(\underline{a_1}, b_2)\}$ and $\mathbf{r}_2 := \mathbf{db} \setminus \{R(\underline{a_1}, b_1)\}$.

We first determine the answers to $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}}$ and $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}}$ in a naive way without using parsimonious counting, but by enumerating repairs. For $i \in \{1, 2\}$, let $M(\mathbf{r}_i)$ be the set of tuples of constants (c_1, c_2) such that there is an embedding θ of q in $\mathbf{r}_i$ mapping (x, y) to (c_1, c_2). We have:

$$\begin{aligned} M(\mathbf{r}_1) &= \{(a_1, b_1), (a_2, b_2)\} \\ M(\mathbf{r}_2) &= \{(a_2, b_2)\} \end{aligned}$$

Clearly, $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}} = 1$ and $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}} = 2$.

It can be easily verified that $q \in \mathsf{Cparsimony}$ with an id-set (x). We next compute $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}}$ and $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}}$ by means of parsimonious counting. To this end, let $q'(x) = \nexists x\,[q]$, and let $\varphi(x)$ be a consistent first-order rewriting for $q'(x)$. If we execute these queries on **db**, we obtain:[1]

$$\begin{aligned} q'(\mathbf{db}) &= \{a_1, a_2\} \\ \varphi(\mathbf{db}) &= \{a_2\} \end{aligned}$$

As stated in Theorem 4.1.4, the set $q'(\mathbf{db})$ yields the upper bound 2 and the set $\varphi(\mathbf{db})$ yields the lower bound 1. It is important to understand that parsimonious counting obtains these bounds directly on **db**, without computing any repair.

[1]$\varphi(\mathbf{db})$ is a shorthand for the set of all tuples (c) such that $\mathbf{db} \models \varphi(c)$.

We elaborate this example further to illustrate the constructs of optimistic and pessimistic repairs. We have:

$$\begin{aligned} q'(\mathbf{r}_1) &= \{a_1, a_2\} \\ q'(\mathbf{r}_2) &= \{a_1\} \end{aligned}$$

Note that the consistent answer to $q'(x)$ on $\mathbf{db}$ (i.e., the set $\varphi(\mathbf{db})$ used previously) is equal to $q'(\mathbf{r}_1) \cap q'(\mathbf{r}_2) = \{a_1\}$. We see that $\mathbf{r}_1$ is an optimistic repair with respect to $q'(x)$, and $\mathbf{r}_2$ a pessimistic repair with respect to $q'(x)$. $\triangleleft$

We finish this section by a proof of Theorem 4.1.4.

Proof of Theorem 4.1.4. Let $q \in \mathsf{Cparsimony}$, $\vec{x}$ be an id-set for q and $g()$ be a counting query such that $\mathsf{BCQ}(g()) = q$. We have to prove that $g()$ admits parsimonious counting. It suffices to show that conditions (A), (B), and (C) in Definition 4.1.1 are satisfied for the id-set $\vec{x}$ of q. As in Definition 4.1.1, let $q'(\vec{x}) = \nexists\vec{x}\,[q]$.

Since q is in $\mathsf{Cparsimony}$, q has an acyclic attack graph. It follows from Theorem 2.0.1 that q has a consistent first-order rewriting. Thus, condition (A) in Definition 4.1.1 is satisfied. It is known (Koutris & Wijsen, 2017) that the attack graph of $q'(\vec{x})$ is a subgraph of the attack graph of q. Informally, no new attacks are introduced when bound variables are made free. It follows that $q'(\vec{x})$ has an acyclic attack graph, and therefore, by Theorem 2.0.1, a consistent first-order rewriting. Thus, condition (B) in Definition 4.1.1 is satisfied. In the remainder of the proof, we show that condition (C) in Definition 4.1.1 is satisfied. To this end, let $\mathbf{db}$ be an arbitrary database instance such that $\mathbf{db} \models_{\mathsf{cqa}} q$.

Let D be the active domain of $\mathbf{db}$. Let f be a function that maps every subset $\mathbf{s}$ of $\mathbf{db}$ to the cardinality of the set $\{\vec{a} \in D^{|\vec{x}|} \mid \mathbf{s} \models q'(\vec{a})\}$. Clearly, for every repair $\mathbf{r}$ of $\mathbf{db}$, we have $\mathbf{r} \subseteq \mathbf{db}$ and hence, since conjunctive queries are monotone, $f(\mathbf{r}) \leq f(\mathbf{db})$. Moreover, since repairs are consistent, it follows by Lemma 4.1.5 that for every repair $\mathbf{r}$ of $\mathbf{db}$, if $[\![g()]\!]^{\mathbf{r}} = i$, then $i = f(\mathbf{r})$.

By Lemma 4.1.7, we can assume an optimistic repair $\mathbf{r}_o$ of $\mathbf{db}$ with respect to $q'(\vec{x})$. By Definition 4.1.4 of an optimistic repair, for every tuple $\vec{a}$ of constants, of arity $|\vec{x}|$, we have $\mathbf{r}_o \models q'(\vec{a})$ if and only if $\mathbf{db} \models q'(\vec{a})$. It follows $f(\mathbf{r}_o) = f(\mathbf{db})$. Consequently, for every repair $\mathbf{r}$ of $\mathbf{db}$, $f(\mathbf{r}) \leq f(\mathbf{r}_o)$. It follows that $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}} = f(\mathbf{db})$.

By Lemma 4.1.8, we can assume a pessimistic repair $\mathbf{r}_p$ of $\mathbf{db}$ with respect to $q'(\vec{x})$. Let $\varphi(\vec{x})$ be a consistent first-order rewriting of $q'(\vec{x})$. By Definition 4.1.5 of a pessimistic repair, the following hold:

- $\mathbf{r}_p \models q(\vec{a})$ if and only if $\mathbf{db} \models \varphi(\vec{a})$. Therefore, $f(\mathbf{r}_p)$ is the cardinality of the set $S := \{\vec{a} \in D^{|\vec{x}|} \mid \mathbf{db} \models \varphi(\vec{a})\}$.
- for every repair $\mathbf{r}$ of $\mathbf{db}$, $f(\mathbf{r}_p) \leq f(\mathbf{r})$.

It follows that $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}} = |S|$. From this, it is correct to conclude that condition (C) in Definition 4.1.1 is satisfied. This concludes the proof. $\square$

4.1.3 Completeness of Cparsimony

In this section, we show the only-if-direction of Theorem 4.0.2, which is the following theorem.

Theorem 4.1.9. *For every counting query $g()$ in* AGGR[sjfBCQ] *that admits parsimonious counting,* BCQ$(g())$ *belongs to* Cparsimony.

The following two lemmas state some properties of queries $g()$ that admit parsimonious counting on some $\vec{x}$.

Lemma 4.1.10. *Let $g()$ be a counting query in* AGGR[sjfBCQ]*, and $q =$* BCQ$(g())$*. If $g()$ admits parsimonious counting, then the attack graph of q is acyclic.*

Proof. Proof by contraposition. If the attack graph of q is cyclic, then by Theorem 2.0.1, q has no consistent first-order rewriting, and therefore $g()$ does not admit parsimonious counting. $\square$

Lemma 4.1.11. *Let $g()$ be a counting query in* AGGR[sjfBCQ]*, and $q =$* BCQ$(g())$*. Let $\vec{x}$ be a (possibly empty) sequence of bound variables of q. If there exists a consistent database instance* $\mathbf{db}$ *and two embeddings θ_1, θ_2 of q in* $\mathbf{db}$ *such that $\theta_1(\vec{x}) = \theta_2(\vec{x})$ but $\theta_1 \neq \theta_2$, then $g()$ does not admit parsimonious counting on $\vec{x}$.*

Proof. Let $\mathbf{db}$ be a consistent database instance and two embeddings θ_1, θ_2 of q in $\mathbf{db}$ such that $\theta_1(\vec{x}) = \theta_2(\vec{x})$ but $\theta_1 \neq \theta_2$. Let $q'(\vec{x}) = \nexists\vec{x}\,[q]$ and $q^*(\vec{x}, \vec{w}) = \nexists\vec{w}\,[q']$ where $\vec{w}$ is a shortest vector containing every variable of $\mathsf{vars}(q)$ that does not appear in $\vec{x}$. It follows that $\theta_1(\vec{w}) \neq \theta_2(\vec{w})$. Let D be the active domain of $\mathbf{db}$. Define:

$$\begin{array}{rcl} M & := & \{\vec{b} \in D^{|\vec{x}|} \mid \mathbf{db} \models q'(\vec{b})\}, \\ M^* & := & \{\vec{b} \cdot \vec{c} \in D^{|\vec{x}|} \cdot D^{|\vec{w}|} \mid \mathbf{db} \models q^*(\vec{b}, \vec{c})\}. \end{array}$$

Clearly, for every $\vec{b} \in M$, there exists $\vec{c} \in D^{|\vec{w}|}$ such that $\vec{b} \cdot \vec{c} \in M^*$. Therefore, $|M| \leq |M^*|$. Moreover, from $\mathbf{db} \models q^*(\theta_1(\vec{x}), \theta_1(\vec{w}))$ and $\mathbf{db} \models q^*(\theta_2(\vec{x}), \theta_2(\vec{w}))$, it follows $|M| < |M^*|$. Since $\mathbf{db}$ is consistent, $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}} = [\![g()]\!]^{\mathbf{db}} = |M^*|$. From $|M| \neq |M^*|$, it is correct to conclude that $g()$ does not admit parsimonious counting on $\vec{x}$. □

The following lemma concerns condition (1) in Definition 4.1.3.

Lemma 4.1.12. *Let $g()$ be a numerical query in $\mathsf{AGGR[sjfBCQ]}$, and $q = \mathsf{BCQ}(g())$. Let $\vec{x}$ be a (possibly empty) sequence of bound variables of q. If $g()$ admits parsimonious counting on $\vec{x}$, then $\vec{x}$ satisfies condition (1) in Definition 4.1.3.*

Proof. The proof is by contraposition. Assume that $\vec{x}$ does not satisfy condition (1) in Definition 4.1.3. We can assume a variable $v \in \mathsf{vars}(q)$ such that $\mathcal{K}(q) \not\models \vec{x} \rightarrow v$. Let θ, μ be two valuations over $\mathsf{vars}(q)$ such that for every variable $u \in \mathsf{vars}(q)$,

$$\theta(u) = \mu(u) \text{ if and only if } \mathcal{K}(q) \models \vec{x} \rightarrow u. \tag{4.1}$$

Let $\mathbf{db} = \theta(q) \cup \mu(q)$. We show that $\mathbf{db}$ is a consistent database instance, i.e., whenever two facts in $\mathbf{db}$ are key-equal, then they are equal. Since q is self-join-free, it suffices to consider any atom F in q such that for every $u \in \mathsf{Key}(F)$, $\theta(u) = \mu(u)$. By Eq. (4.1), it follows $\mathcal{K}(q) \models \vec{x} \rightarrow \mathsf{Key}(F)$. Since $\mathcal{K}(q)$ contains $\mathsf{Key}(F) \rightarrow \mathsf{vars}(F)$, it follows $\mathcal{K}(q) \models \vec{x} \rightarrow \mathsf{vars}(F)$ by Armstrong's transitivity axiom. By Eq. (4.1), $\theta(F) = \mu(F)$.

Since $\mathcal{K}(q) \models \vec{x} \rightarrow \vec{x}$, we have $\theta(\vec{x}) = \mu(\vec{x})$ by Eq. (4.1). Since $\mathcal{K}(q) \not\models \vec{x} \rightarrow v$, it follow that $\theta(v) \neq \mu(v)$ by Eq. (4.1) and, thus, $\theta \neq \mu$. By Lemma 4.1.11, it is correct to conclude that $g()$ does not admit parsimonious counting on $\vec{x}$. □

The following two lemmas, and their corollary, concern condition (2) in Definition 4.1.3.

Lemma 4.1.13. *Let $g()$ be a numerical query in $\mathsf{AGGR[sjfBCQ]}$, and $q = \mathsf{BCQ}(g())$. Let $\vec{x}$ be a (possibly empty) sequence of bound variables of q, and let $q'(\vec{x}) = \nexists\vec{x}\,[q]$. If $g()$ admits parsimonious counting on $\vec{x}$, then, for every database instance $\mathbf{db}$ such that $\mathbf{db} \models_{\mathsf{cqa}} q$, $\mathbf{db}$ has an optimistic repair with respect to $q'(\vec{x})$.*

Proof. Assume that $g()$ admits parsimonious counting on $\vec{x}$. Let $\mathbf{db}$ be a database instance such that $\mathbf{db} \models_{\mathsf{cqa}} q$.

Define

$$S := \{\vec{d} \in D^{|\vec{x}|} \mid \mathbf{db} \models q'(\vec{d})\}, \tag{4.2}$$

where D be the active domain of $\mathbf{db}$. By our hypothesis that $g()$ admits parsimonious counting on $\vec{x}$, it follows by condition (C) in Definition 4.1.1 that

$$[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}} = |S|. \tag{4.3}$$

By Definition of $\mathsf{LUB\text{-}CQA}(g())$, we can assume a repair $\mathbf{r}$ of $\mathbf{db}$ such that $[\![g()]\!]^{\mathbf{r}} = [\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}}$. Define

$$S_{\mathbf{r}} := \{\vec{d} \in D^{|\vec{x}|} \mid \mathbf{r} \models q'(\vec{d})\}. \tag{4.4}$$

By our hypothesis that $g()$ admits parsimonious counting on $\vec{x}$, it follows by condition (C) in Definition 4.1.1 that

$$[\![g()]\!]^{\mathbf{r}} = [\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{r}} = |S_{\mathbf{r}}|. \tag{4.5}$$

Since conjunctive queries are monotone and $\mathbf{r} \subseteq \mathbf{db}$, it follows $S_{\mathbf{r}} \subseteq S$. Since $|S_{\mathbf{r}}| = |S|$ by (4.3) and (4.5), it follows $S_{\mathbf{r}} = S$. From $S \subseteq S_{\mathbf{r}}$, it follows that $\mathbf{r}$ is an optimistic repair with respect to $q'(\vec{x})$. □

Lemma 4.1.14. *Let $g()$, q, $\vec{x}$, $q'(\vec{x})$ as in the statement of Lemma 4.1.13. Assume that $\vec{x}$ violates condition (2) in Definition 4.1.3. Then, there exists a database $\mathbf{db}$ such that $\mathbf{db} \models_{\mathsf{cqa}} q$ and $\mathbf{db}$ has no optimistic repair with respect to $q'(\vec{x})$.*

The proof for Lemma 4.1.14 is provided in Section B.3 of Appendix B.

Corollary 4.1.14.1. *Let $g()$, q, $\vec{x}$, $q'(\vec{x})$ as in the statement of Lemma 4.1.13. If $g()$ admits parsimonious counting on $\vec{x}$, then $\vec{x}$ satisfies condition (2) in Definition 4.1.3.*

Proof. Immediately from Lemmas 4.1.13 and 4.1.14. □

Before giving a proof of Theorem 4.1.9, we illustrate the preceding results with an example.

Example 4.1.10. Let $g() := \texttt{SUM}(1) \leftarrow R(\underline{x}, z, y), S(\underline{y}, x), T(\underline{y}, x)$. Let $q = \mathsf{BCQ}(g())$. We will argue that q is not in $\mathsf{Cparsimony}$, and then illustrate that $g()$ does not admit parsimonious counting.

The only attacks in q are $R \overset{q}{\rightsquigarrow} S$ and $R \overset{q}{\rightsquigarrow} T$. Assume for the sake of contradiction that $q \in \mathsf{Cparsimony}$. Then, following Proposition 4.1.2, the minimal id-set for q is the empty sequence (). However, since $\mathcal{K}(q) \equiv \{x \rightarrow y, x \rightarrow z, y \rightarrow x\}$, condition (1) in Definition 4.1.3 is violated for $\vec{x} = ()$. We conclude by contradiction that $q \notin \mathsf{Cparsimony}$.

We now argue, without using Theorem 4.1.9, that $g()$ does not admit parsimonious counting. Conditions (A) and (B) in Definition 4.1.1 of parsimonious counting are satisfied for every choice of $\vec{x}$ in $\{(), (x), (y), (x, y)\}$. However, we will show that condition (C) is not satisfied. To this end, let $\vec{x}$ be a sequence of bound variables of q. Let $q'(\vec{x}) = \nexists \vec{x}\,[q]$. First, suppose that $\vec{x} \in \{(x), (x, y)\}$. Consider the following database instance $\mathbf{db}$:

R	$\underline{x}$	z	y
	a	d	e
	b	d	e
	c	d	f

S	$\underline{y}$	x
	e	a
	e	b
	f	c

T	$\underline{y}$	x
	e	a
	e	b
	f	c

We have that $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}} = 1$ and $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}} = 2$, but it can be easily verified that $|q'(\mathbf{db})| = 3$, which is distinct from the upper bound 2.

Assume next that $\vec{x} \in \{(y), (x, y)\}$. Consider the following database instance $\mathbf{db}$:

R	$\underline{x}$	z	y
	a	d	e
	a	d	f
	b	d	g

S	$\underline{y}$	x
	e	a
	f	a
	g	b

T	$\underline{y}$	x
	e	a
	f	a
	g	b

Now we have that $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}} = 2$ and $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}} = 2$, but $|q'(\mathbf{db})| = 3$.

The only remaining case to be considered is $\vec{x} = ()$. In that case $q'(\vec{x}) = q$. Consider the following database instance $\mathbf{db}$:

R	$\underline{x}$	z	y
	a	d	e
	b	d	f

S	$\underline{y}$	x
	e	a
	f	b

T	$\underline{y}$	x
	e	a
	f	b

Since $\mathbf{db}$ is a consistent database instance, the only repair of $\mathbf{db}$ is $\mathbf{db}$ itself. We have that $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}} = 2$ and $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}} = 2$. It can be easily verified that $|q'(\mathbf{db})| = 1$, which is distinct from the upper bound 2.

Finally, we claim (without proof) that 2-DIMENSIONAL MATCHING (2DM) can be first-order reduced to computing $\mathsf{LUB\text{-}CQA}(g())$. Therefore, since 2DM is NL-hard (Chandra et al., 1984), $g()$ cannot admit parsimonious counting under standard complexity assumptions. $\triangleleft$

We can now give the proof of Theorem 4.1.9.

Proof of Theorem 4.1.9. Assume that $g()$ admits parsimonious counting. Let $q = \mathsf{BCQ}(g())$. Then, q has a tuple $\vec{x}$ of bound variables such that for the query $q'(\vec{x}) := \nexists \vec{x}\,[q]$, the conditions (A), (B), and (C) in Definition 4.1.1 are satisfied. By Lemma 4.1.10, q has an acyclic attack graph, and thus condition (I) in Definition 4.1.3 is satisfied. By Lemma 4.1.12, condition (1) in Definition 4.1.3 is satisfied for $\vec{x}$. By Corollary 4.1.14.1, condition (2) in Definition 4.1.3 is satisfied by $\vec{x}$. Since we have shown that q satisfies all conditions in Definition 4.1.3, we conclude $q \in \mathsf{Cparsimony}$. $\square$

4.1.4 Comparison with Cforest

In this section, we introduce Cforest and show, without using Theorem 4.0.1, that $\mathsf{Cforest} \subsetneq \mathsf{Cparsimony}$. Theorem 4.0.1 then follows by Theorem 4.1.4.

Definition 4.1.6 (Cforest)**.** Let q be a self-join-free Boolean conjunctive query. The *Fuxman graph* of q is a directed graph whose vertices are the atoms of q. There is a directed edge from an atom F to an atom G if $F \neq G$ and $\mathsf{notKey}(F)$ contains a bound variable that also occurs in G. The class Cforest contains all (and only) boolean self-join free conjunctive queries q whose Fuxman graph is a directed forest satisfying, for every directed edge from F to G, $\mathsf{Key}(G) \subseteq \mathsf{notKey}(F)$. $\square$

Theorem 4.1.15. $\mathsf{Cforest} \subsetneq \mathsf{Cparsimony}$.

The proof of Theorem 4.1.15 is provided in Section B.4 of Appendix B.

4.1.5 Parsimonious Counting in AGGR[FOL]

Finally, we show that the expressions in (i) and (ii) of Definition 4.1.1 can be expressed in AGGR[FOL]. To this end, let $q'(\vec{x})$ and $\varphi(\vec{x})$ be as in Definition 4.1.1. Then, the values m and n in (i) and (ii) are computed by the following AGGR[FOL] terms, respectively:

$$\phi_{\mathsf{glb}}() := \mathsf{Aggr}_{\mathcal{F}_{\mathtt{SUM}}}\vec{x}\,[1, \varphi(\vec{x})]\,;$$
$$\phi_{\mathsf{lub}}() := \mathsf{Aggr}_{\mathcal{F}_{\mathtt{SUM}}}\vec{x}\,\left[1, q'(\vec{x})\right].$$

4.2. Parsimonious Aggregation

A natural question is whether the main ideas behind parsimonious counting can be applied to numerical queries whose head differs from SUM(1). In this section, we will generalize the notion of parsimonious counting to *parsimonious aggregation.*

Definition 4.2.1 (Parsimonious Aggregation)**.** Let $g()$ be a numerical query in $\mathsf{AGGR}[\mathsf{sjfBCQ}]$ with head $\mathtt{AGG}(r)$. Let $q = \mathsf{BCQ}(g())$. Let $\vec{x}$ be a (possibly empty) sequence of distinct bound variables of q. We say that $g()$ admits *parsimonious aggregation* on $\vec{x}$ if the following hold (let $q'(\vec{x}) = \nexists\vec{x}\,[q]$):

(A) q has a consistent first-order rewriting;

(B) $q'(\vec{x})$ has a consistent first-order rewriting (call it $\varphi(\vec{x})$); and

(C) for every database instance $\mathbf{db}$ such that $\mathbf{db} \models_{\mathsf{cqa}} q$, the following conditions (a) and (b) are equivalent:

(a) $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}} = m$ and $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}} = n$.

(b) Both the following conditions hold:

(i) *Minimality Condition:* Let $\vec{c}_1, \dots, \vec{c}_k$ enumerate all the elements $\vec{c}$ of $\mathsf{adom}(\mathbf{db})^{|\vec{x}|}$ such that $\mathbf{db} \models \varphi(\vec{c})$ holds. For every $i \in \{1, \dots, k\}$, let M_i be the set of embeddings of q in $\mathbf{db}$ that map $\vec{x}$ to $\vec{c}_i$, and define $v^{(i)}_{\min} := \min\{\theta(r) \mid \theta \in M_i\}$. Then, $\mathcal{F}_{\mathtt{AGG}}(\{\{v^{(1)}_{\min}, \dots, v^{(k)}_{\min}\}\}) = m$.

(ii) *Maximality Condition:* Let $\vec{d}_1, \dots, \vec{d}_\ell$ enumerate all the elements $\vec{d}$ of $\mathsf{adom}(\mathbf{db})^{|\vec{x}|}$ such that $\mathbf{db} \models q'(\vec{d})$ holds. For every $i \in \{1, \dots, \ell\}$, let N_i be the set of embeddings of q in $\mathbf{db}$ that map $\vec{x}$ to $\vec{d}_i$, and define $v^{(i)}_{\max} := \max\{\theta(r) \mid \theta \in N_i\}$. Then, $\mathcal{F}_{\mathtt{AGG}}(\{\{v^{(1)}_{\max}, \dots, v^{(\ell)}_{\max}\}\}) = n$.

We say that $g()$ *admits parsimonious aggregation* if it admits parsimonious aggregation on some sequence $\vec{x}$ of bound variables. □

Note that parsimonious counting is a particular case of parsimonious aggregation where $\mathtt{AGG}(r) = \mathtt{SUM}(1)$.

4.2.1 Cparsimony Admits Parsimonious Aggregation

In this section, we prove that the if-direction in Theorem 4.0.2 can be extended to parsimonious aggregation for numerical queries with monotone aggregate operator, as stated by the following theorem.

Theorem 4.2.1. *Let $g()$ be a numerical query using a monotone aggregate operator such that $\mathsf{BCQ}(g()) \in \mathsf{Cparsimony}$. Then, $g()$ admits parsimonious aggregation.*

We use a number of helping lemmas and constructs. We first introduce $\mathtt{MAX}$-*optimistic repairs*, a variant of the optimistic repair notion.

Definition 4.2.2 (MAX-Optimistic repair)**.** Let $q(\vec{x})$ be a conjunctive query. Let r be a non-negative rational number or a numerical variable in $\mathsf{vars}(q)$. Let $\mathbf{db}$ be a database instance. We say that a repair $\mathbf{r}$ of $\mathbf{db}$ is a MAX-*optimistic repair* with respect to $q(\vec{x})$ and r if both the following conditions hold:

1. $\mathbf{r}$ is an optimistic repair with respect to $q(\vec{x})$; and
2. for every $\vec{d} \in \mathsf{adom}(\mathbf{db})^{|\vec{x}|}$ such that $\mathbf{r} \models q(\vec{d})$, if M is the set of embeddings of q in $\mathbf{db}$ that map $\vec{x}$ to $\vec{d}$, then there is an embedding θ^* of q in $\mathbf{r}$ such that $\theta^*(\vec{x}) = \vec{d}$ and $\theta^*(r) = \max\{\theta(r) \mid \theta \in M\}$.

□

The following lemma gives a sufficient condition for the existence of MAX-optimistic repairs.

Lemma 4.2.2. *Let $q \in \mathsf{Cparsimony}$, $\vec{x}$ an id-set for q, and $q'(\vec{x}) = \nexists\vec{x}\,[q]$. Let r be a non-negative rational number or a numerical variable in $\mathsf{vars}(q)$. Let $\mathbf{db}$ be a database instance such that $\mathbf{db} \models_{\mathsf{cqa}} q$. Then, $\mathbf{db}$ has a* MAX-*optimistic repair with respect to $q'(\vec{x})$ and r.*

Proof. Let $\vec{d_1}, \dots, \vec{d_n}$ enumerate all the elements $\vec{d}$ of $\mathsf{adom}(\mathbf{db})^{|\vec{x}|}$ such that $\mathbf{db} \models q'(\vec{d})$ holds. By Lemma 4.1.6, for each $i \in \{1, \dots, n\}$, there exists an embedding θ_i of q in $\mathbf{db}$ satisfying the following conditions:

(a) $\theta_i(\vec{x}) = \vec{d_i}$;

(b) for every $j \in \{1, \dots, n\}$, for every atom F in q, if $\theta_i(F)$ and $\theta_j(F)$ are key-equal, then $\theta_i(F) = \theta_j(F)$; and

(c) for every embedding θ of q in $\mathbf{db}$ such that $\theta(\vec{x}) = \vec{d_i}$, $\theta(r) \leq \theta_i(r)$.

It follows that there is a repair $\mathbf{r}$ of $\mathbf{db}$ such that for every $i \in \{1, \dots, n\}$, θ_i is an embedding of q in $\mathbf{r}$. Since $\mathbf{r} \models q'(\vec{d_i})$ for every $i \in \{1, \dots, n\}$, the repair $\mathbf{r}$ is a MAX-optimistic repair of $\mathbf{db}$ with respect to $q'(\vec{x})$. □

We now present the notion of MIN-*pessimistic repair.*

Definition 4.2.3 (MIN-Pessimistic repair)**.** Let $q(\vec{x})$ be a conjunctive query. Let r be a non-negative rational number or a numerical variable in $\mathsf{vars}(q)$. Let $\mathbf{db}$ be a database instance. We say that a repair $\mathbf{r}$ of $\mathbf{db}$ is a MIN-pessimistic repair with respect to $q(\vec{x})$ and r if

1. $\mathbf{r}$ is a pessimistic repair of $\mathbf{db}$ with respect to $q(\vec{x})$; and

2. for every $\vec{d} \in \mathsf{adom}(\mathbf{db})^{|\vec{x}|}$ such that $\mathbf{r} \models q(\vec{d})$, if M is the set of embeddings of q in $\mathbf{db}$ that map $\vec{x}$ to $\vec{d}$, then there is an embedding θ^* of q in $\mathbf{r}$ such that $\theta^*(\vec{x}) = \vec{d}$ and $\theta^*(r) = \min\{\theta(r) \mid \theta \in M\}$.

$\square$

We introduce some helping lemmas.

Lemma 4.2.3. *Let q be a query in* Cparsimony*, and let $\vec{x}$ be an id-set for q. Let $q'(\vec{x})$ be the query $\nexists\vec{x}\,[q]$. Let $\mathbf{db}$ be a database instance such that $\mathbf{db} \models_{\mathsf{cqa}} q$. Let $\{\vec{d_1}, \ldots, \vec{d_n}\}$ be a subset of $\mathsf{adom}(\mathbf{db})^{|\vec{x}|}$ such that $\mathbf{db} \models q'(\vec{d_i})$ for every $i \in \{1, \ldots, n\}$. Let r be a non-negative rational number or a numerical variable in $\mathsf{vars}(q)$. For each $i \in \{1, \ldots, n\}$, there exists an embedding θ_i of q in $\mathbf{db}$ satisfying the following conditions:*

(a) $\theta_i(\vec{x}) = \vec{d_i}$;

(b) for every $j \in \{1, \ldots, n\}$, for every atom F in q, if $\theta_i(F)$ and $\theta_j(F)$ are key-equal, then they are equal; and

(c) for every embedding θ of q in $\mathbf{db}$ such that $\theta(\vec{x}) = \vec{d_i}$, $\theta(r) \geq \theta_i(r)$.

The proof of Lemma 4.2.3 is symmetrical to the proof of Lemma 4.1.6.

Lemma 4.2.4. *Let q be a self-join free Boolean conjunctive query. Let $\mathbf{db}$ be a database instance. Let $\vec{x}$ be a vector of variables in q such that no variable in $\vec{x}$ is attacked in q. Let $q'(\vec{x}) = \nexists\vec{x}\,[q]$. Let f be the function with domain $2^{\mathbf{db}}$ that maps every subset $\mathbf{s}$ of $\mathbf{db}$ to the set $\{\vec{a} \in \mathsf{adom}(\mathbf{db})^{|\vec{x}|} \mid \mathbf{s} \models q'(\vec{a})\}$. Let $\mathbf{r}$ be a repair of $\mathbf{db}$. Let θ be an embedding of q in $\mathbf{r}$. Let F be an atom in q, and A a fact in $\mathbf{db}$ with the same relation name as the atom F such that $\theta(F)$ and A are key-equal. Then, $f((\mathbf{r} \setminus \{\theta(F)\}) \cup \{A\}) \subseteq f(\mathbf{r})$.*

The proof for Lemma 4.2.4 is provided in the Appendix of (Koutris & Wijsen, 2017). The following lemma gives a sufficient condition for the existence of `MIN`-pessimistic repairs.

Lemma 4.2.5. *Let $q \in$ Cparsimony, $\vec{x}$ an id-set for q, and $q'(\vec{x}) = \nexists\vec{x}\,[q]$. Let r be a non-negative rational number or a numerical variable in $\mathsf{vars}(q)$. Let $\mathbf{db}$ be a database instance such that $\mathbf{db} \models_{\mathsf{cqa}} q$. Then, $\mathbf{db}$ has a* `MIN`*-pessimistic repair with respect to $q'(\vec{x})$ and r.*

Proof. Let $\vec{d_1}, \ldots, \vec{d_n}$ enumerate all the elements $\vec{d}$ of $\mathsf{adom}(\mathbf{db})^{|\vec{x}|}$ such that $\mathbf{db} \models_{\mathsf{cqa}} q'(\vec{d})$ holds. By Lemma 4.2.3, for each $i \in \{1, \ldots, n\}$, there exists an embedding θ_i^* of q in $\mathbf{db}$ satisfying the following conditions:

(a) $\theta^*_i(\vec{x}) = \vec{d_i}$;

(b) for every $j \in \{1, \ldots, n\}$, for every atom F in q, if $\theta^*_i(F)$ and $\theta^*_j(F)$ are key-equal, then $\theta^*_i(F) = \theta^*_j(F)$; and

(c) for every embedding θ of q in $\mathbf{db}$ such that $\theta(\vec{x}) = \vec{d_i}$, $\theta(r) \geq \theta^*_i(r)$.

By Lemma 4.1.8, $\mathbf{db}$ has a pessimistic repair $\mathbf{r}$ with respect to $q'(\vec{x})$. By Definition 4.1.3, $\mathcal{K}(q) \models \vec{x} \to \mathsf{vars}(q)$. Let $\sigma = (F_1, \ldots, F_\ell)$ be a sequential proof for $\vec{x} \to \mathsf{vars}(q)$ in q. We will build, by induction on increasing i, a sequence of repairs $(\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_\ell)$ such that for every $i \in \{0, \ldots, \ell\}$:

1. $\mathbf{r}_i$ is a pessimistic repair of $\mathbf{db}$ with respect to $q'(\vec{x})$; and
2. for every $j \in \{1, \ldots, n\}$, $\theta^*_j(\{F_1, \ldots, F_i\}) \subseteq \mathbf{r}_i$.

Note that, if a variable $v \in \vec{x}$ does not appear in some atom of σ, then $\vec{x}$ is not minimal, a contradiction. Thus, $\mathsf{vars}(q) \subseteq \bigcup_{k=1}^{\ell} \mathsf{vars}(F_k)$.

$\boxed{\text{Basis } i = 0.}$ Trivial with $\mathbf{r}_0 = \mathbf{r}$.

$\boxed{\text{Step } i-1 \to i.}$ Assume, as induction hypothesis, that there is a repair $\mathbf{r}_{i-1}$ of $\mathbf{db}$ such that:

1. $\mathbf{r}_{i-1}$ is a pessimistic repair of $\mathbf{db}$ with respect to $q'(\vec{x})$; and
2. for every $j \in \{1, \ldots, n\}$, $\theta^*_j(\{F_1, \ldots, F_{i-1}\}) \subseteq \mathbf{r}_{i-1}$.

Let $\theta_1, \ldots, \theta_m$ be (all) the embeddings of q in $\mathbf{r}_{i-1}$. Since $\mathbf{r}_{i-1}$ is a pessimistic repair of $\mathbf{db}$ with respect to $q'(\vec{x})$, it follows that for every $j \in \{1, \ldots, m\}$, there is a $k \in \{1, \ldots, n\}$ such that $\theta_j(\vec{x}) = \vec{d_k}$. From Lemma 4.1.5, it follows that for every $j, k \in \{1, \ldots, m\}$, if $\theta_j(\vec{x}) = \theta_k(\vec{x})$, then $j = k$. Thus, we have that $m = n$. Assume, without loss of generality, that for every $j \in \{1, \ldots, n\}$, $\theta_j(\vec{x}) = \theta^*_j(\vec{x}) = \vec{d_j}$.

Now we argue that, for every $j \in \{1, \ldots, n\}$, $\theta_j(F_i)$ and $\theta^*_j(F_i)$ are key-equal. Let $j \in \{1, \ldots, n\}$. By definition of a sequential proof, it follows that $\mathsf{Key}(F_1) \subseteq \vec{x}$. Since $\theta_j(\vec{x}) = \theta^*_j(\vec{x})$ and $\theta^*_j(F_1) \in \mathbf{r}_{i-1}$ by induction hypothesis, it follows that $\theta_j(F_1) = \theta^*_j(F_1)$. By definition of a sequential proof, it follows that $\mathsf{Key}(F_2) \subseteq \vec{x} \cup \mathsf{vars}(F_1)$. Since $\theta_j(\vec{x}) = \theta^*_j(\vec{x})$, $\theta_j(F_1) = \theta^*_j(F_1)$ and $\theta^*_j(F_2) \in \mathbf{r}_{i-1}$ by induction hypothesis, it follows that $\theta_j(F_2) = \theta^*_j(F_2)$. And so on, until we reach $\theta_j(F_{i-1}) = \theta^*_j(F_{i-1})$. By definition of a sequential proof, it follows that $\mathsf{Key}(F_i) \subseteq \vec{x} \cup \bigcup_{k=1}^{i-1} \mathsf{vars}(F_k)$. Since $\theta_j(\vec{x}) = \theta^*_j(\vec{x})$ and, for every $v \in \bigcup_{k=1}^{i-1} \mathsf{vars}(F_k)$, $\theta_j(v) = \theta^*_j(v)$, it follows that $\theta_j(F_i)$ and $\theta^*_j(F_i)$ are key-equal.

If $\theta_j(F_i) = \theta^*_j(F_i)$ for every $j \in \{1, \dots, n\}$, then $\mathbf{r}_i = \mathbf{r}_{i-1}$ satisfies the wanted properties. Assume that there is $j \in \{1, \dots, n\}$ such that $\theta_j(F_i)$ and $\theta^*_j(F_i)$ are key-equal but different. Let $\mathbf{r}'_i = (\mathbf{r}_{i-1} \setminus \{\theta_j(F_i)\}) \cup \{\theta^*_j(F_i)\}$. By Lemma 4.2.4, $\mathbf{r}'_i$ is a pessimistic repair of $\mathbf{db}$ with respect to $q'(\vec{x})$. Note that all variables of an id-set are unattacked (by item (b) in Lemma B.1.3, provided in Section B.1 of Appendix B). By applying this modification for every $j \in \{1, \dots, n\}$ such that $\theta_j(F_i)$ and $\theta^*_j(F_i)$ are key-equal but different, we achieve a pessimistic repair $\mathbf{r}_i$ such that for every $j \in \{1, \dots, n\}$, $\theta^*_j(\{F_1, \dots, F_i\}) \subseteq \mathbf{r}_i$. Note that each of these modifications will never remove a fact $\theta^*_j(F_i)$ for some $j \in \{1, \dots, n\}$. Indeed, if it was the case, there are $j, k \in \{1, \dots, n\}$ such that $j \neq k$, $\theta^*_j(F_i)$ and $\theta^*_k(F_i)$ are key-equal but different, which is a contradiction. This concludes the induction step of the proof.

It follows that $\mathbf{r}_\ell$ is a pessimistic repair of $\mathbf{db}$ with respect to $q'(\vec{x})$ such that for every $j \in \{1, \dots, n\}$, $\theta^*_j(\{F_1, \dots, F_\ell\}) \subseteq \mathbf{r}_\ell$. Let $j \in \{1, \dots, n\}$. Since $\mathbf{r}_\ell$ is a pessimistic repair of $\mathbf{db}$ with respect to $q'(\vec{x})$, there is an embedding θ' of q in $\mathbf{r}_\ell$ such that $\theta'(\vec{x}) = \vec{d_j}$. By definition of a sequential proof, it follows that $\mathsf{Key}(F_1) \subseteq \vec{x}$. Since $\theta'(\vec{x}) = \theta^*_j(\vec{x})$ and $\theta^*_j(F_1) \in \mathbf{r}_\ell$, it follows that $\theta'(F_1) = \theta^*_j(F_1)$. By definition of a sequential proof, it follows that $\mathsf{Key}(F_2) \subseteq \vec{x} \cup \mathsf{vars}(F_1)$. Since $\theta'(\vec{x}) = \theta^*_j(\vec{x})$, $\theta'(F_1) = \theta^*_j(F_1)$ and $\theta^*_j(F_2) \in \mathbf{r}_\ell$, it follows that $\theta'(F_2) = \theta^*_j(F_2)$. And so on, until we reach $\theta'(F_\ell) = \theta^*_j(F_\ell)$. Since $\mathsf{vars}(q) \subseteq \bigcup_{k=1}^{\ell} \mathsf{vars}(F_k)$, it follows that $\theta' = \theta^*_j$. Thus, we have that for every $j \in \{1, \dots, n\}$, θ^*_j is an embedding of q in $\mathbf{r}_\ell$. From Lemma 4.1.5, since $\mathbf{r}_\ell$ is a pessimistic repair of $\mathbf{db}$ with respect to $q'(\vec{x})$, we obtain that $\theta^*_1, \dots, \theta^*_m$ are (all) the embeddings of q in $\mathbf{r}_\ell$. Since for every $j \in \{1, \dots, n\}$, for every embedding θ of q in $\mathbf{db}$ such that $\theta(\vec{x}) = \vec{d_j}$, we have $\theta(r) \geq \theta^*_j(r)$, it follows that $\mathbf{r}_\ell$ is a $\mathtt{MIN}$-pessimistic repair of $\mathbf{db}$. $\square$

The following example illustrates the preceding constructs and lemmas.

Example 4.2.1. Let $g() := \mathtt{SUM}(r) \leftarrow R(\underline{x}, y), S(\underline{x, y}, r)$. Let $q = \mathsf{BCQ}(g())$. Let $\mathbf{db}$ be the following database instance:

R	$\underline{x}$	y
	a_1	b_1
	a_1	b_2
	a_2	b_2

S	$\underline{x}$	$\underline{y}$	r
	a_1	b_1	2
	a_1	b_1	3
	a_2	b_2	2
	a_2	b_2	3

We have that $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}} = 2$ and $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}} = 6$. It can be easily verified that $q \in \mathsf{Cparsimony}$ with an id-set (x). We next compute $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}}$ and $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}}$ by means of parsimonious aggregation. To this end, let $q'(x) = \nexists x\,[q]$, and let $\varphi(x)$ be a consistent first-order rewriting for $q'(x)$. If we execute these queries on $\mathbf{db}$, we obtain:[2]

$$\begin{aligned} q'(\mathbf{db}) &= \{a_1, a_2\} \\ \varphi(\mathbf{db}) &= \{a_2\} \end{aligned}$$

For $i \in \{1, 2\}$, let M_i be the set of embeddings θ of q in $\mathbf{db}$ such that $\theta(x) = a_i$, $v^{(i)}_{\min} = \min\{\theta(r) \mid \theta \in M_i\}$ and $v^{(i)}_{\max} = \max\{\theta(r) \mid \theta \in M_i\}$. We have

$$v^{(1)}_{\min} = v^{(2)}_{\min} = 2$$

and

$$v^{(1)}_{\max} = v^{(2)}_{\max} = 3.$$

As stated in Theorem 4.2.1, we have

$$[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}} = \mathcal{F}_{\mathtt{SUM}}(\{\{v^{(2)}_{\min}\}\}) = \mathcal{F}_{\mathtt{SUM}}(2) = 2$$

and

$$[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}} = \mathcal{F}_{\mathtt{SUM}}(\{\{v^{(1)}_{\max}, v^{(2)}_{\max}\}\}) = \mathcal{F}_{\mathtt{SUM}}(3, 3) = 6.$$

It is important to understand that parsimonious aggregation obtains these bounds directly on $\mathbf{db}$, without computing any repair.

We elaborate this example further to illustrate the constructs of `MAX`-optimistic and `MIN`-pessimistic repairs. It can be easily verified that every repair of $\mathbf{db}$ containing the fact $R(a_1, b_1)$ is an optimistic repair of $\mathbf{db}$, and every repair of $\mathbf{db}$ containing the fact $R(a_1, b_2)$ is a pessimistic repair of $\mathbf{db}$. Let $\mathbf{r}_1$ be a repair that keeps the facts $R(a_1, b_2)$, and $S(a_2, b_2, 2)$. Let $\mathbf{r}_2$ be a repair that keeps the facts $R(a_1, b_1)$, $S(a_1, b_1, 3)$, and $S(a_2, b_2, 3)$. It is easily verifiable that $\mathbf{r}_1$ is a `MIN`-pessimistic repair of $\mathbf{db}$ with respect to $q'(\vec{x})$ and r, and $\mathbf{r}_2$ is a `MAX`-optimistic repair of $\mathbf{db}$ with respect to $q'(\vec{x})$ and r. We have:

$$\begin{aligned} [\![g()]\!]^{\mathbf{r}_1} &= \mathcal{F}_{\mathtt{SUM}}(\{\{v^{(2)}_{\min}\}\}) \\ [\![g()]\!]^{\mathbf{r}_2} &= \mathcal{F}_{\mathtt{SUM}}(\{\{v^{(1)}_{\max}, v^{(2)}_{\max}\}\}) \end{aligned}$$

$\lhd$

We finish this section with a proof of Theorem 4.2.1.

Proof of Theorem 4.2.1. Let q be a self-join-free Boolean conjunctive query in

[2]$\varphi(\mathbf{db})$ is a shorthand for the set of all tuples (c) such that $\mathbf{db} \models \varphi(c)$.

$\mathsf{Cparsimony}$. Let $\vec{x}$ be an id-set for q. Let r be a non-negative rational number or a numerical variable in $\mathsf{vars}(q)$. Let $g()$ be a numerical query with a head $\mathtt{AGG}(r)$ such that $\mathcal{F}_{\mathtt{AGG}}$ is a monotone aggregate operator and $\mathsf{BCQ}(g()) = q$. We have to prove that $g()$ admits parsimonious aggregation. It suffices to show that conditions (A), (B), and (C) in Definition 4.2.1 are satisfied for the id-set $\vec{x}$ of q. As in Definition 4.2.1, let $q'(\vec{x}) = \nexists\vec{x}\,[q]$.

Since q is in $\mathsf{Cparsimony}$, q has an acyclic attack graph. It follows from Theorem 2.0.1 that q has a consistent first-order rewriting. Thus, condition (A) in Definition 4.2.1 is satisfied. It is known (Koutris & Wijsen, 2017) that the attack graph of $q'(\vec{x})$ is a subgraph of the attack graph of q. Informally, no new attacks are introduced when bound variables are made free. It follows that $q'(\vec{x})$ has an acyclic attack graph, and therefore, by Theorem 2.0.1, a consistent first-order rewriting. Thus, condition (B) in Definition 4.2.1 is satisfied. In the remainder of the proof, we show that condition (C) in Definition 4.2.1 is satisfied. To this end, let $\mathbf{db}$ be an arbitrary database instance such that $\mathbf{db} \models_{\mathsf{cqa}} q$. By Lemma 4.2.5, there is a $\mathtt{MIN}$-pessimistic repair $\mathbf{r}_{\min}$ of $\mathbf{db}$ with respect to $q'(\vec{x})$ and r. By Lemma 4.2.2, there is a $\mathtt{MAX}$-optimistic repair $\mathbf{r}_{\max}$ of $\mathbf{db}$ with respect to $q'(\vec{x})$ and r. To show that condition (C) in Definition 4.2.1 is satisfied, it suffices to show that for every repair $\mathbf{r}$ of $\mathbf{db}$, we have

$$[\![g()]\!]^{\mathbf{r}_{\min}} \leq [\![g()]\!]^{\mathbf{r}} \leq [\![g()]\!]^{\mathbf{r}_{\max}}. \tag{4.6}$$

Note that the computations described in condition (b) correspond to $[\![g()]\!]^{\mathbf{r}_{\min}}$ and $[\![g()]\!]^{\mathbf{r}_{\max}}$ by definition of $\mathtt{MIN}$-pessimistic and $\mathtt{MAX}$-optimistic repairs.

Let $\mathbf{r}$ be a repair of $\mathbf{db}$. Let $\Theta = \{\theta_1, \dots, \theta_\ell\}$ be the set of embeddings of q in $\mathbf{r}$. Let $\Theta^+ = \{\theta_1^+, \dots, \theta_n^+\}$ be the set of embeddings of q in $\mathbf{r}_{\max}$. Let $\Theta^- = \{\theta_1^-, \dots, \theta_m^-\}$ be the set of embeddings of q in $\mathbf{r}_{\min}$. By Lemma 4.1.5, two distinct embeddings in Θ cannot agree on every variable of $\vec{x}$. This same property applies to Θ^+ and Θ^-.

By Definition 4.1.5, we have that for every $\theta^- \in \Theta^-$, there is a $\theta \in \Theta$ such that $\theta^-(\vec{x}) = \theta(\vec{x})$. It follows that $m \leq \ell$. Assume, without loss of generality, that for every $i \in \{1, \dots, m\}$, $\theta_i^-(\vec{x}) = \theta_i(\vec{x})$. It follows that

$$[\![g()]\!]^{\mathbf{r}_{\min}} = \mathcal{F}_{\mathtt{AGG}}(\{\{\theta_1^-(r), \dots, \theta_m^-(r)\}\}) \tag{4.7}$$

and

$$[\![g()]\!]^{\mathbf{r}} = \mathcal{F}_{\mathtt{AGG}}(\{\{\theta_1(r), \dots, \theta_m(r), \theta_{m+1}(r), \dots, \theta_\ell(r)\}\}). \tag{4.8}$$

By Definition 4.2.3, it follows that for every $i \in \{1, \dots, m\}$, $\theta_i^-(r) \leq \theta_i(r)$. By monotonicity of $\mathcal{F}_{\mathtt{AGG}}$, (4.7), and (4.8), it follows that $[\![g()]\!]^{\mathbf{r}_{\min}} \leq [\![g()]\!]^{\mathbf{r}}$.

By Definition 4.1.4, we have that for every $\theta \in \Theta$, there is a $\theta^+ \in \Theta^+$ such that $\theta(\vec{x}) = \theta^+(\vec{x})$. It follows that $\ell \leq n$. Assume, without loss of generality, that for every $i \in \{1, \dots, \ell\}$, $\theta_i(\vec{x}) = \theta_i^+(\vec{x})$. It follows that

$$[\![g()]\!]^{\mathbf{r}} = \mathcal{F}_{\mathtt{AGG}}(\{\{\theta_1(r), \dots, \theta_\ell(r)\}\}) \tag{4.9}$$

and

$$[\![g()]\!]^{\mathbf{r}_{\max}} = \mathcal{F}_{\mathtt{AGG}}(\{\{\theta_1^+(r), \dots, \theta_\ell^+(r), \theta_{\ell+1}^+(r), \dots, \theta_n^+(r)\}\}). \tag{4.10}$$

By Definition 4.2.2, it follows that for every $i \in \{1, \dots, \ell\}$, $\theta_i(r) \leq \theta_i^+(r)$. By monotonicity of $\mathcal{F}_{\mathtt{AGG}}$, (4.9) and (4.10), it follows that $[\![g()]\!]^{\mathbf{r}} \leq [\![g()]\!]^{\mathbf{r}_{\max}}$. We conclude that (4.6) holds and thus g admits parsimonious aggregation. □

4.2.2 Incompleteness of Cparsimony for MAX-queries

In this section, we show that the only-if-direction of Theorem 4.0.2 only applies for parsimonious aggregation with some monotone aggregate operators. First, we have that, if a SUM-query $g()$ admit parsimonious aggregation, then $\mathsf{BCQ}(g()) \in \mathsf{Cparsimony}$.

Theorem 4.2.6. *For every* SUM*-query* $g()$ *in* AGGR[sjfBCQ] *that admits parsimonious aggregation,* $\mathsf{BCQ}(g())$ *belongs to* Cparsimony.

Proof. The proof of Theorem 4.1.9 can be easily extended to SUM-queries and parsimonious aggregation. □

Now, we prove the existence of a MAX-query $g()$ such that $\mathsf{BCQ}(g()) \notin \mathsf{Cparsimony}$ and $g()$ admits parsimonious aggregation. Note that $\mathcal{F}_{\mathtt{MAX}}$ is a monotone aggregate operator.

Theorem 4.2.7. *There is a* MAX*-query* $g()$ *such that* $\mathsf{BCQ}(g()) \notin \mathsf{Cparsimony}$ *and* $g()$ *admits parsimonious aggregation.*

Proof. Consider the following numerical query

$$g() := \mathtt{MAX}(r) \leftarrow T(\underline{c}, r), R_1(\underline{x_1}, y), S_1(\underline{x_1}, y), R_2(\underline{x_2}, y), S_2(\underline{x_2}, y).$$

Let $q = \mathsf{BCQ}(g())$. The edge-set of q's attack graph is empty and no variable in $\mathsf{vars}(q)$ is frozen. Note that, since c is a constant, we have $\mathcal{K}(q) \models \emptyset \rightarrow r$.

We first show that $q \notin \mathsf{Cparsimony}$. According to condition (1) in Definition 4.1.3, every id-set (if any) must contain x_1. However, no id-set can contain x_1, because for the atom $R_2(\underline{x_2}, y)$, the edge $\{y, x_1\}$ in the Gaifman graph of q is a path between a variable in $\mathsf{notKey}(R_2)$ and x_1 that uses no variable in $\mathsf{Key}(R_2)$. We conclude that q is not in $\mathsf{Cparsimony}$.

Now, we show that q admits admits parsimonious aggregation on $\vec{x} = ()$. It suffices to show that conditions (A), (B), and (C) in Definition 4.2.1 are satisfied for $\vec{x}$. As in Definition 4.2.1, let $q' = \nexists\vec{x}\,[q] = q$.

It is easily verifiable that q has an acyclic attack graph. It follows from Theorem 2.0.1 that q has a consistent first-order rewriting. Thus, condition (A) in Definition 4.2.1 is satisfied. Since $q' = q$, it follows that q' has an acyclic attack graph and a consistent first-order rewriting. Thus, condition (B) in Definition 4.2.1 is satisfied. In the remainder of the proof, we show that condition (C) in Definition 4.2.1 is satisfied. To this end, let $\mathbf{db}$ be an arbitrary database instance such that $\mathbf{db} \models_{\mathsf{cqa}} q$.

Let $\mathbf{r}$ be a repair of $\mathbf{db}$. Let $n = |\mathbf{Emb}_q(\mathbf{r})|$. Since $\mathbf{db} \models_{\mathsf{cqa}} q$, we have $n \geq 1$. Since $\mathcal{K}(q) \models \emptyset \to r$, it follows that, for every $\theta_1, \theta_2 \in \mathbf{Emb}_q(\mathbf{r})$, $\theta_1(r) = \theta_2(r)$. Let $v_{\mathbf{r}}$ be the value such that for every $\theta \in \mathbf{Emb}_q(\mathbf{r})$, $\theta(r) = v_{\mathbf{r}}$. Thus, we have

$$[\![g()]\!]^{\mathbf{r}} = \mathcal{F}_{\mathtt{MAX}}(\{\{\overbrace{v_{\mathbf{r}}, \ldots, v_{\mathbf{r}}}^{n \text{ times}}\}\}) = v_{\mathbf{r}}. \tag{4.11}$$

From definition of $\mathsf{GLB\text{-}CQA}(g())$, we have

$$[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}} = \min\{v_{\mathbf{r}} \mid \mathbf{r} \in \mathsf{rset}(\mathbf{db})\}. \tag{4.12}$$

From definition of $\mathsf{LUB\text{-}CQA}(g())$, we have

$$[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}} = \max\{v_{\mathbf{r}} \mid \mathbf{r} \in \mathsf{rset}(\mathbf{db})\}. \tag{4.13}$$

Let $\theta_{\min} \in \mathbf{Emb}_q(\mathbf{db})$ such that for every $\theta \in \mathbf{Emb}_q(\mathbf{db})$, $\theta_{\min}(r) \leq \theta(r)$. Let $\mathbf{r}_{\min}$ be a repair of $\mathbf{db}$ such that $\theta_{\min}$ is an embedding of q in $\mathbf{r}_{\min}$. From (4.11), we have

$$[\![g()]\!]^{\mathbf{r}_{\min}} = v_{\mathbf{r}_{\min}} = \theta_{\min}(r).$$

It follows that, for every $\mathbf{r} \in \mathsf{rset}(\mathbf{db})$, $[\![g()]\!]^{\mathbf{r}_{\min}} \leq [\![g()]\!]^{\mathbf{r}}$. From (4.12), $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}} = \theta_{\min}(r)$. Let $\theta_{\max} \in \mathbf{Emb}_q(\mathbf{db})$ such that for every $\theta \in \mathbf{Emb}_q(\mathbf{db})$, $\theta(r) \leq \theta_{\max}(r)$. Let $\mathbf{r}_{\max}$ be a repair of $\mathbf{db}$ such that $\theta_{\max}$ is an embedding of q in $\mathbf{r}_{\max}$. From (4.11), we have

$$[\![g()]\!]^{\mathbf{r}_{\max}} = v_{\mathbf{r}_{\max}} = \theta_{\max}(r).$$

It follows that, for every $\mathbf{r} \in \mathsf{rset}(\mathbf{db})$, $[\![g()]\!]^{\mathbf{r}} \leq [\![g()]\!]^{\mathbf{r}_{\max}}$. From (4.13), $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}} = \theta_{\max}(r)$. We can conclude that condition (C) in Definition 4.2.1 is satisfied and, thus, $g()$ admits parsimonious aggregation. $\square$

4.2.3 Parsimonious Aggregation in AGGR[FOL]

Finally, we show that *parsimonious aggregation* can be captured in AGGR[FOL]; that is, the *Minimality* and *Maximality Conditions* from Definition 4.2.1 can be expressed within AGGR[FOL]. To this end, let $g()$ and $\vec{x}$ be as in Definition 4.2.1, and assume that $g()$ admits parsimonious aggregation on $\vec{x}$, with $\varphi(\vec{x})$ as given in the same definition. We illustrate the *Minimality Condition* for the case where r is a variable such that $r \notin \mathsf{vars}(\vec{x})$. The other cases are analogous. Define the following AGGR[FOL] formulas:

$$\begin{aligned}\psi_{\min}(\vec{x}, w) &:= \varphi(\vec{x}) \wedge w = \mathsf{Aggr}_{\mathcal{F}_{\mathtt{MIN}}} r \left[r, \nexists r \left[q'(\vec{x})\right]\right] ;\\ \phi_{\mathsf{glb}}() &:= \mathsf{Aggr}_{\mathcal{F}_{\mathsf{AGG}}} \vec{x}, w \left[w, \psi_{\min}(\vec{x}, w)\right].\end{aligned}$$

These formulas compute the value $\mathcal{F}_{\mathsf{AGG}}(\{\{v_{\min}^{(1)}, \dots, v_{\min}^{(k)}\}\})$ described in the *Minimality Condition* of Definition 4.2.1. It follows that for every database instance $\mathbf{db}$ such that $\mathbf{db} \models_{\mathsf{cqa}} \mathsf{BCQ}(g())$, we have

$$[\![\phi_{\mathsf{glb}}()]\!]^{\mathbf{db}} = [\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}}.$$

CHAPTER 5

From Repairs to Maximal Consistent Subsets

In this chapter, we introduce a new construct called the *maximal consistent subset* (MCS). This construct will serve as a useful tool in the following chapters for computing $\mathsf{GLB\text{-}CQA}(g())$ and $\mathsf{LUB\text{-}CQA}(g())$ for numerical queries $g()$ in $\mathsf{AGGR}[\mathsf{sjfBCQ}]$ with a monotone aggregate operator.

Let $g()$ be some numerical query in $\mathsf{AGGR}[\mathsf{sjfBCQ}]$ and $q = \mathsf{BCQ}(g())$. The notion that will be introduced below arises from the observation that, as defined in Section 3.2, the computation of $[\![g()]\!]^{\mathbf{db}}$ for some database instance $\mathbf{db}$ only takes into account the embeddings of q in $\mathbf{db}$. Every fact in $\mathbf{db}$ that is not used by some embedding of q in $\mathbf{db}$ can be omitted in the computation. Thus, to compute $\mathsf{GLB\text{-}CQA}(g())$ or $\mathsf{LUB\text{-}CQA}(g())$, no repair $\mathbf{r}$ of $\mathbf{db}$ needs to be computed as long as we can obtain the embeddings of q in $\mathbf{r}$. Consider the following construct:

Definition 5.0.1. Let q be a query in sjfBCQ, and $\mathbf{db}$ be a database instance. Let M be a set of embeddings of q in $\mathbf{db}$. A *maximal consistent subset (MCS) of M* is a $\subseteq$-maximal subset N of M such that $N \models \mathcal{K}(q)$. We write $\mathsf{MCS}_q(M)$ for the subset of 2^M that contains all (and only) MCSs of M. □

The following lemmas describe the relation between repairs and maximal consistent subsets.

Lemma 5.0.1. *Let q be a query in sjfBCQ, and $\mathbf{db}$ be a database instance. Let $\mathbf{r}$ be a repair of $\mathbf{db}$. There is an MCS N of $\mathbf{Emb}_q(\mathbf{db})$ such that $\mathbf{Emb}_q(\mathbf{r}) \subseteq N$.*

Proof. Trivial. □

Lemma 5.0.2. *Let q be a query in* sjfBCQ*, and* **db** *be a database instance. Let N be an MCS of* $\mathbf{Emb}_q(\mathbf{db})$*. There is a repair* **r** *of* **db** *such that* $\mathbf{Emb}_q(\mathbf{r}) = N$*.*

Proof. Consider the repair **r** of **db** built as follows: for every block in **db**, if there is an embedding in N that uses a fact in that block, keep that fact. Otherwise, keep any fact on the block. Note that, by definition of an MCS, $N \models \mathcal{K}(q)$ and, thus, two embeddings in N can not use different facts in a same block. Clearly, by construction, $N \subseteq \mathbf{Emb}_q(\mathbf{r})$. It remains to be shown that $\mathbf{Emb}_q(\mathbf{r}) \subseteq N$. Assume, for the sake of contradiction, that there is an embedding $\theta \in \mathbf{Emb}_q(\mathbf{r})$ such that $\theta \notin N$. Clearly, $\theta \in \mathbf{Emb}_q(\mathbf{db})$. By definition of a repair, it follows that $N \cup \{\theta\} \models \mathcal{K}(q)$ and, thus, N is not an MCS of $\mathbf{Emb}_q(\mathbf{db})$; contradiction. We conclude by contradiction that $\mathbf{Emb}_q(\mathbf{r}) \subseteq N$. □

Informally, for every maximal consistent subset N of $M_{\mathbf{db}}$, there is a repair of **db** with exactly the embeddings in N, but the reverse is not necessarily true.

Example 5.0.1. Let $q = \exists x(R(\underline{x}, c))$. Let **db** be the following database instance:

R	$\underline{x}$	c
	a	c
	a	d

Let $\mathbf{r} = \mathbf{db} \setminus \{R(a,c)\}$. Clearly, **r** is a repair of **db** and $\mathbf{Emb}_q(\mathbf{r}) = \emptyset$. However, the empty set is not an MCS of $\mathbf{Emb}_q(\mathbf{db})$ since $\mathbf{Emb}_q(\mathbf{db})$ contains an embedding θ of q in **db** mapping x to a. ◁

If $\mathcal{F}_{\texttt{AGG}}$ is monotone, we obtain the following lemma.

Lemma 5.0.3. *Let $g()$ be a numerical query in* AGGR[sjfBCQ]*, with a head* AGG(r)*,* $q = \mathsf{BCQ}(g())$ *and such that* $\mathcal{F}_{\texttt{AGG}}$ *is monotone. Let* **db** *be a database instance. For every couple* **r***,* **s** *of repairs of* **db***, if* $\mathbf{Emb}_q(\mathbf{r}) \subseteq \mathbf{Emb}_q(\mathbf{s})$*, then* $[\![g()]\!]^{\mathbf{r}} \leq [\![g()]\!]^{\mathbf{s}}$*.*

Proof. Straightforward from monotonicity of $\mathcal{F}_{\texttt{AGG}}$. □

The following theorem expresses LUB-CQA$(g())$ using maximal consistent subsets.

Theorem 5.0.4. *Let $g()$ be a numerical query in* $\mathsf{AGGR[FOL]}$*, with a head* $\mathtt{AGG}(r)$ *and* $q = \mathsf{BCQ}(g())$*, such that* $\mathcal{F}_{\mathtt{AGG}}$ *is monotone. Let* **db** *be a database instance. Then,*

- *if some repair of* **db** *falsifies q and $\mathcal{F}_{\mathtt{AGG}}(\emptyset)$ is undefined, then*

$$[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}} = \bot; \tag{5.1}$$

- *otherwise,*

$$[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}} = \max_{N \in \mathsf{MCS}_q(\mathbf{Emb}_q(\mathbf{db}))} \mathcal{F}_{\mathtt{AGG}}(\{\{\theta(r) \mid \theta \in N\}\}). \tag{5.2}$$

Proof. Straightforward from Lemma 5.0.1, Lemma 5.0.2 and Lemma 5.0.3. □

In Chapter 6, we will prove that, under some constraints, a similar result can be achieved for $\mathsf{GLB\text{-}CQA}(g())$ by replacing max with min and restricting $\mathbf{Emb}_q(\mathbf{db})$ to a subset that can be computed in FOL.

Now, consider the following notion.

Definition 5.0.2 ($\mathcal{F}_{\mathtt{AGG}}$-maximal value for a partial valuation)**.** Let $g()$ be a numerical query in $\mathsf{AGGR[sjfBCQ]}$ with a head $\mathtt{AGG}(r)$ and $q = \mathsf{BCQ}(g())$. Let **db** be a database instance. Let θ be a valuation over some subset of $\mathsf{vars}(q)$ that can be extended to an embedding of q in **db**. We write $\mathsf{Ext}(\theta \mid q, \mathbf{db})$ (or simply $\mathsf{Ext}(\theta)$ if q and **db** are clear from the context) for the set of embeddings of q in **db** that extend θ.

The rational number v defined as

$$v := \max_{N \in \mathsf{MCS}_q(\mathsf{Ext}(\theta))} \left\{\mathcal{F}_{\mathtt{AGG}}\left(\{\{\theta'(r) \mid \theta' \in N\}\}\right)\right\}$$

is called the *$\mathcal{F}_{\mathtt{AGG}}$-maximal value for θ in* **db**; and an MCS N^* of $\mathsf{Ext}(\theta)$ is called *$\mathcal{F}_{\mathtt{AGG}}$-maximal* if it satisfies $\mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N^*\}\}\right) = v$. We say that an MCS N^* of $\mathsf{Ext}(\theta)$ is *$\mathcal{F}_{\mathtt{COUNT}}$-maximal* if for every MCS N of M, we have $|N^*| \leq |N|$.

The notions of a *$\mathcal{F}_{\mathtt{AGG}}$-minimal value for θ in* **db** and a *$\mathcal{F}_{\mathtt{AGG}}$-minimal MCS* of $\mathsf{Ext}(\theta)$ are defined symmetrically by replacing max with min. A *$\mathcal{F}_{\mathtt{COUNT}}$-minimal MCS* is also defined symmetrically by replacing $\leq$ with $\geq$. □

Informally, Theorem 5.0.4 implies that we can compute $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}}$, for some database instance **db**, by computing the $\mathcal{F}_{\mathtt{AGG}}$-maximal value for the empty valuation in **db**, which requires computing maximal consistent subsets rather than repairs. Moreover, we will see in Chapter 6 that, under some constraints, a similar result can be obtained for $\mathsf{GLB\text{-}CQA}(g())$.

CHAPTER 6

GLB-CQA with Monotone, Associative Aggregate Operators

In this chapter, we study the expressibility of $\mathsf{GLB\text{-}CQA}(g())$ in $\mathsf{AGGR[FOL]}$ for numerical queries $g()$ in $\mathsf{AGGR[sjfBCQ]}$ with aggregate operators that are both monotone and associative. `SUM` and `MAX` are among the most common aggregate operators with these properties, and are therefore covered by Theorem 6.0.1, the main result of this chapter. Note that counting queries are also covered, since they take the form $\texttt{SUM}(1) \leftarrow q(\vec{y})$.

Theorem 6.0.1 (Separation Theorem for $\mathsf{GLB\text{-}CQA}$)**.** *The following decision problem is decidable in quadratic time (in the size of the input): Given as input a numerical query $g()$ in* $\mathsf{AGGR[sjfBCQ]}$ *whose aggregate operator is both monotone and associative, is* $\mathsf{GLB\text{-}CQA}(g())$ *expressible in* $\mathsf{AGGR[FOL]}$*? Moreover, if the answer is "yes," then it is possible to effectively construct, also in quadratic time, a formula in* $\mathsf{AGGR[FOL]}$ *that solves* $\mathsf{GLB\text{-}CQA}(g())$*.*

The formal proof of Theorem 6.0.1 will be provided in Section 6.4. In this chapter, we will not discuss numerical queries $g()$ such that the attack graph of $\mathsf{BCQ}(g())$ is cyclic, as they are already covered by Theorem 3.5.6. Thus, most of this chapter will be dedicated to prove the expressibility of $\mathsf{GLB\text{-}CQA}(g())$ in $\mathsf{AGGR[FOL]}$ for numerical queries $g()$ where the attack graph of $\mathsf{BCQ}(g())$ is acyclic, which is stated by the following theorem.

Theorem 6.0.2. *Let $g()$ be a numerical query in* $\mathsf{AGGR[sjfBCQ]}$ *using an aggregate operator $\mathcal{F}_{\texttt{AGG}}$ that is monotone and associative, and such that the attack graph of* $\mathsf{BCQ}(g())$ *is acyclic. Then,* $\mathsf{GLB\text{-}CQA}(g())$ *is expressible in* $\mathsf{AGGR[FOL]}$*.*

Section 6.2 will present the main ideas of the proof of Theorem 6.0.2, using some new constructs previously introduced in Section 6.1. Then, the formal proof of Theorem 6.0.2 will be provided in Section 6.3. Section 6.4 will provide the formal proof of Theorem 6.0.1. Finally, Section 6.5 discusses how our results relate to those of a recent study by Kolaitis et al. (2025).

6.1. ∀Embeddings and Superfrugal Repairs

In (Koutris & Wijsen, 2017), a construct called "frugal repair" was introduced. In the current section, we introduce a related but more stringent construct, called *superfrugal repair.*

∀embeddings Whenever $q(\vec{u})$ is a conjunction of atoms, we write q to denote the closed formula $\exists\vec{u}(q(\vec{u}))$. Let $q(\vec{u})$ now be a self-join-free conjunction of n atoms such that the attack graph of q is acyclic. The following definitions are relative to a fixed topological sort $(F_1, \dots, F_n)$ of q's attack graph and a fixed database instance **db**. We define the following sequences of variables for $\ell \in \{1, \dots, n\}$:

- $\vec{u}_\ell$ contains all (and only) variables of $\bigcup_{i=1}^{\ell} \mathsf{vars}(F_i)$. Thus, $\vec{u}_n = \vec{u}$;
- $\vec{x}_\ell$ contains the variables of $\mathsf{Key}(F_\ell)$ that do not occur in $\bigcup_{i=1}^{\ell-1} \mathsf{vars}(F_i)$; and
- $\vec{y}_\ell$ contains the variables of $\mathsf{notKey}(F_\ell)$ that do not already occur in $\bigcup_{i=1}^{\ell-1} \mathsf{vars}(F_i)$.

Moreover, we define $\vec{u}_0 = ()$, the empty sequence. With this notation, we have that for every $\ell \in \{1, \dots, n\}$,

$$\vec{u}_\ell = (\vec{u}_{\ell-1}, \vec{x}_\ell, \vec{y}_\ell). \tag{6.1}$$

The following definition is by induction. An ℓ-embedding θ is called an *ℓ-∀embedding (of q in* **db***)* if one of the following holds true:

Basis: $\ell = 0$ and every repair of **db** satisfies q; or

Step: $\ell \geq 1$ and both the following hold true:

- $(\mathbf{db}, \theta\restriction_{\vec{u}_{\ell-1}\vec{x}_\ell}) \models_{\mathsf{cqa}} F_\ell \land F_{\ell+1} \land \cdots \land F_n$; and
- the $(\ell-1)$-embedding contained in θ is an $(\ell-1)$-∀embedding.

In simple terms, the first bullet states that every repair must satisfy the query whose atoms are obtained from $F_\ell, F_{\ell+1}, \ldots, F_n$ by replacing x with $\theta(x)$ whenever x is a variable that occurs in the primary key of F_ℓ or in an atom that precedes F_ℓ. The second bullet implies that the same condition must hold for $\ell-1, \ell-2, \ldots, 2, 1$, and eventually 0. An ℓ-$\forall$embedding with $\ell = n$ is also called a $\forall$*embedding* for short. We write $\forall\mathbf{Emb}_q(\mathbf{db})$ for the set containing every $\forall$embedding of q in $\mathbf{db}$. We say that a ℓ-key-embedding γ is a ℓ-$\forall$*key-embedding* if $\gamma = \theta \restriction_{\vec{u}_{\ell-1}\vec{x}_\ell}$ for some ℓ-$\forall$embedding θ.

Example 6.1.1. The query $q_0 = \exists t \exists p(\mathit{Dealers}(\underline{\text{“James”}}, t) \land \mathit{Stock}(\underline{p, t}, 35))$ checks if there is any product stored in a quantity of 35 in the town where James is a dealer. It holds true in every repair of the database instance $\mathbf{db}_{\mathsf{Stock}}$ of Fig. 1.1. The embedding $\{t \mapsto \text{“Boston”}, p \mapsto \text{“Tesla Y”}\}$ is a $\forall$embedding. On the other hand, the embedding $\theta := \{t \mapsto \text{“Boston”}, p \mapsto \text{“Tesla X”}\}$ is not a $\forall$embedding, because $\theta \restriction_{\{t,p\}} = \theta$ and $(\mathbf{db}_{\mathsf{Stock}}, \theta) \not\models_{\mathsf{cqa}} \mathit{Stock}(\underline{p, t}, 35)$. Indeed, if $\mathbf{r}$ is a repair that contains $\mathit{Stock}(\underline{\text{“Tesla X”}, \text{“Boston”}}, 40)$, then $(\mathbf{r}, \theta) \not\models \mathit{Stock}(\underline{p, t}, 35)$. $\lhd$

We now state two important helping lemmas. The first one states that all topological sorts of an acyclic attack graph yield the same $\forall$embeddings. The second lemma establishes that $\forall$embeddings can be computed in FOL.

Lemma 6.1.1. *Let q a query in* sjfBCQ *with an acyclic attack graph, and* $\mathbf{db}$ *be a database instance. Let n be the number of atoms in q. Let $\prec_1$ and $\prec_2$ be two topological sorts of q's attack graph. Every n-$\forall$embedding relative to $\prec_1$ is an n-$\forall$embedding relative to $\prec_2$.*

The proof for Lemma 6.1.1 is provided in Section C.2 of Appendix C.

Lemma 6.1.2. *Let $q := \exists\vec{u}(q(\vec{u}))$ be a query in* sjfBCQ *with an acyclic attack graph. It is possible to construct, in quadratic time in the size of q, a* FOL *formula $\varphi(\vec{u})$ such that for every database instance* $\mathbf{db}$*, for every valuation θ over $\vec{u}$, $(\mathbf{db}, \theta) \models \varphi(\vec{u})$ if and only if θ is a $\forall$embedding of q in* $\mathbf{db}$.

The proof for Lemma 6.1.2 is provided in Section C.3 of Appendix C. Note that since the formula $\varphi(\vec{u})$ in Lemma 6.1.2 can be constructed in quadratic time, its length is at most quadratic (in the size of q).

Superfrugal repairs A repair $\mathbf{r}$ of a database instance $\mathbf{db}$ is *superfrugal* relative to a query q in sjfBCQ if every embedding of q in $\mathbf{r}$ is a $\forall$embedding of q in $\mathbf{db}$. Informally, superfrugal repairs are repairs with $\subseteq$-minimal sets of embeddings, which is expressed by Lemma 6.1.3.

Example 6.1.2. Continuation of Example 6.1.1. Let $\mathbf{r}$ be the repair of $\mathbf{db}_{\mathsf{Stock}}$ that contains all (and only) tuples preceded by † in Fig. 1.1. Then, $\mathbf{r}$ is not superfrugal relative to $q_0 := \exists t \exists p(\mathit{Dealers}(\underline{\text{“James”}}, t) \wedge \mathit{Stock}(\underline{p, t}, 35))$. Indeed, $\theta := \{t \mapsto \text{“Boston”}, p \mapsto \text{“Tesla X”}\}$ is an embedding of q_0 in $\mathbf{r}$, but as discussed in Example 6.1.1, θ is not a ∀embedding of q_0 in $\mathbf{db}$. ◁

Lemma 6.1.3. *Let* $\mathbf{db}$ *be a database instance, and* q *a query in* sjfBCQ *with an acyclic attack graph. For every repair* $\mathbf{r}$ *of* $\mathbf{db}$*, there exists a superfrugal repair* $\mathbf{r}^*$ *of* $\mathbf{db}$ *such that every embedding of* q *in* $\mathbf{r}^*$ *is also an embedding of* q *in* $\mathbf{r}$.

The proof of Lemma 6.1.3 is provided in Section C.4 of Appendix C. Lemma C.5.1 in Appendix C.5 shows that for queries $q \in$ sjfBCQ with an acyclic attack graph, superfrugal repairs are identical to the n-minimal repairs defined in (Figueira et al., 2023) and the $\preceq_q^X$-frugal repairs introduced in (Koutris & Wijsen, 2017), where n denotes the number of atoms and $X = \mathsf{vars}(q)$. In the current work, we opted to define superfrugal repairs in terms of ∀embeddings; an alternative approach would be to take n-minimal repairs (or, equivalently, $\preceq_q^X$-frugal repairs) as a starting point and show that all embeddings in them are ∀embeddings. This alternative approach would also arrive at the conclusion of Lemma 6.1.2 regarding the computability of ∀embeddings in FOL. Specifically, Lemma 8 and Remark 11 in (Figueira et al., 2023, Section 4) entail that one can compute in FOL a set E that contains, for every $\ell \in \{0, 1, 2, \ldots, n\}$, each ℓ-∀embedding. The goal in Figueira et al. (2023) is to check whether E contains a 0-∀embedding (or, equivalently, the empty set), which indicates that q holds true in every repair. For that purpose, it is sufficient for the set E to be a superset, rather than an exact match, of the set of all ℓ-∀embeddings, for $0 \leq \ell \leq n$. Lemma 6.1.2 differs in that it refers to a set that contains exactly all n-∀embeddings and nothing else. On the other hand, we will argue in Chapter 9 that the more general technical development in (Figueira et al., 2023), which also handles cyclic attack graphs, could be particularly valuable for exploring the polynomial-time computability of GLB-CQA($g()$), an intriguing open problem that is not the focus of our current work.

It follows from Theorems 2.0.1 and 6.0.2 that for a numerical query $g()$ in AGGR[sjfBCQ] whose aggregate operator is both monotone and associative, if CERTAINTY(BCQ($g()$)) is expressible in FOL, then GLB-CQA($g()$) is expressible in AGGR[FOL]. However, it will soon become apparent that transitioning from CERTAINTY(BCQ($g()$)) to GLB-CQA($g()$) introduces new challenges that require novel techniques. Briefly, in the former problem, it is sufficient to determine the existence (or non-existence) of a 0-∀embedding. In contrast, the latter

R	$\underline{x}$	y
	a_1	b_1
	a_1	b_2
	a_2	b_2
	a_2	b_3
	a_3	b_4

S	$\underline{y}$	$\underline{z}$	d	r
	b_1	c_1	d	1
	b_1	c_1	d	2
	b_1	c_2	d	3
	b_2	c_3	d	5
	b_2	c_3	d	6
	b_3	c_4	d	5
	b_4	c_5	d	7
	b_4	c_5	e	8

$M_0 := [\![\phi_0]\!]^{\mathbf{db}_0}$	x	y	z	r
	a_1	b_1	c_1	1
	a_1	b_1	c_1	2
	a_1	b_1	c_2	3
	a_1	b_2	c_3	5
	a_1	b_2	c_3	6
	a_2	b_2	c_3	5
	a_2	b_2	c_3	6
	a_2	b_3	c_4	5

Figure 6.1: Example database instance $\mathbf{db}_0$, and the set M_0 of all ∀embeddings of q_0 into $\mathbf{db}_0$.

$[\![\phi_1]\!]^{M_0}$	$xyz \rightarrow r$			
	$\underline{x}$	$\underline{y}$	$\underline{z}$	r
	a_1	b_1	c_1	1
	~~a_1~~	~~b_1~~	~~c_1~~	~~2~~
	a_1	b_1	c_2	3
	a_1	b_2	c_3	5
	~~a_1~~	~~b_2~~	~~c_3~~	~~6~~
	a_2	b_2	c_3	5
	~~a_2~~	~~b_2~~	~~c_3~~	~~6~~
	a_2	b_3	c_4	5

$[\![\phi_2]\!]^{M_0}$	$x \rightarrow y$			
	$\underline{x}$	y	z	r
	a_1	b_1	c_1	1
	a_1	b_1	c_2	3
	~~a_1~~	~~b_2~~	~~c_3~~	~~5~~
	a_2	b_2	c_3	5
	~~a_2~~	~~b_3~~	~~c_4~~	~~5~~

Figure 6.2: Computation of an $\mathcal{F}_{\texttt{SUM}}$-minimal MCS relative to $\{x \rightarrow y, yz \rightarrow r\}$.

problem, if a 0-∀embedding exists, additionally requires finding a ⊆-maximal consistent set of ∀embeddings whose aggregated value is minimal.

6.2. Main Ideas for Proving Theorem 6.0.2

In this section, we use a concrete example to introduce the main ingredients of the proof for Theorem 6.0.2. Our example uses the database instance $\mathbf{db}_0$ of Fig. 6.1 and the following `SUM`-query $g_0()$, in which d is a constant:

$$\texttt{SUM}(r) \leftarrow \overbrace{R(\underline{x}, y), S(\underline{y, z}, d, r)}^{q_0(x,y,z,r)}. \qquad (g_0())$$

$$\begin{aligned}
\sigma(x,y,z,r) &:= R(\underline{x},y) \wedge S(\underline{y,z},d,r) \wedge \forall v \forall r' \left(S(\underline{y,z},v,r') \rightarrow v = d\right)\\
\phi_0(x,y,z,r) &:= \sigma(x,y,z,r) \wedge \forall y' \left(R(\underline{x},y') \rightarrow \exists z' \exists r' \left(\sigma(x,y',z',r')\right)\right)\\
\phi_1(x,y,z,r) &:= \phi_0(x,y,z,r) \wedge \forall r' \left(\phi_0(x,y,z,r') \rightarrow r \leq r'\right)\\
t(x,y) &:= \mathsf{Aggr}_{\mathcal{F}_{\mathtt{SUM}}}(z,r)\,[r, \phi_1(x,y,z,r)]\\
\phi_2(x,y,z,r) &:= \ \phi_1(x,y,z,r) \wedge\\
&\quad \forall y' \forall z' \forall r' \left(\phi_1(x,y',z',r') \rightarrow \left(\begin{array}{l} t(x,y) \leq t(x,y') \wedge \\ (y' \prec y \rightarrow t(x,y) < t(x,y')) \end{array}\right)\right)
\end{aligned}$$

$$\begin{aligned}
\psi_2(x,v) &:= \exists y \exists z \exists r \left(\begin{array}{l} \phi_1(x,y,z,r) \wedge (v = t(x,y)) \wedge \\ \forall y' \forall z' \forall r' \left(\phi_1(x,y',z',r') \rightarrow (t(x,y) \leq t(x,y'))\right) \end{array}\right)\\
\mathsf{GLB\text{-}CQA}(g_0()) &:= \mathsf{Aggr}_{\mathcal{F}_{\mathtt{SUM}}}(x,v)\,[v, \psi_2(x,v)]
\end{aligned}$$

Figure 6.3: Calculation of both an $\mathcal{F}_{\mathtt{SUM}}$-minimal MCS (formula ϕ_2) and $\mathsf{GLB\text{-}CQA}(g_0())$ for $\mathtt{SUM}(r) \leftarrow R(\underline{x},y), S(\underline{y,z},d,r)$.

The attack graph of the underlying Boolean conjunctive query has a single attack, from the R-atom to the S-atom. Fig. 6.1 shows the set M_0 of all $\forall$embeddings of q_0 in $\mathbf{db}_0$, which can be calculated by the FOL formula $\phi_0(x,y,z,r)$ in Fig. 6.3. Note incidentally that the embedding (in $\mathbf{db}_0$) which maps (x,y,z,r) to $(a_3,b_4,c_5,d,7)$ is not a $\forall$embedding, because of the value e ($e \neq d$) in the last row of the S-relation. We have $\mathcal{K}(q_0) = \{x \rightarrow y, yz \rightarrow r\}$, and from Fig. 6.1, it is clear that $M_0 \not\models \mathcal{K}(q_0)$. We now introduce a crucial lemma about maximal consistent subsets of M_0.

Lemma 6.2.1. *Let q be a query in* sjfBCQ *with an acyclic attack graph. Let* $\mathbf{db}$ *be a database instance. Then,*

1. *for every superfrugal repair* $\mathbf{r}$ *of* $\mathbf{db}$*,* $\mathbf{Emb}_q(\mathbf{r})$ *is an MCS of* $\forall\mathbf{Emb}_q(\mathbf{db})$*; and*

2. *whenever N is an MCS of* $\forall\mathbf{Emb}_q(\mathbf{db})$*, there is a superfrugal repair* $\mathbf{r}$ *of* $\mathbf{db}$ *such that* $\mathbf{Emb}_q(\mathbf{r}) = N$*.*

The proof of Lemma 6.2.1 is provided in Section C.6 of Appendix 6. The following Corollary 6.2.1.1, which expresses $\mathsf{GLB\text{-}CQA}(g())$ in terms of the construct of MCS, will be very helpful. It requires monotonicity, but not associativity, of aggregate operators.

Corollary 6.2.1.1. *Let $g()$ be a numerical query in* $\mathsf{AGGR[FOL]}$*, with a head* $\mathtt{AGG}(r)$ *and* $q = \mathsf{BCQ}(g())$*, such that the attack graph of q is acyclic. Let* $\mathbf{db}$ *be a database instance. If* $\mathcal{F}_{\mathtt{AGG}}$ *is monotone, then*

- *if* $\forall\mathbf{Emb}_q(\mathbf{db}) = \emptyset$ *and* $\mathcal{F}_{\mathtt{AGG}}(\emptyset)$ *is undefined, then*
$$[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}} = \bot; \tag{6.2}$$
- *otherwise,*
$$[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}} = \min_{N \in \mathsf{MCS}_q(\forall\mathbf{Emb}_q(\mathbf{db}))} \mathcal{F}_{\mathtt{AGG}}(\{\{\theta(r) \mid \theta \in N\}\}). \tag{6.3}$$

Proof. Immediate corollary of Lemma 6.2.1. Note that $\forall\mathbf{Emb}_q(\mathbf{db}) = \emptyset$ implies that some repair of $\mathbf{db}$ falsifies q. □

The proof of Theorem 6.0.1 constructs an $\mathsf{AGGR[FOL]}$-formula which correctly calculates the right-hand side of equation (6.3), provided that $\mathcal{F}_{\mathtt{AGG}}$ is not only monotone, but also associative. In the remainder, we illustrate the construction by our running example, which uses q_0 and M_0.

First, we focus on calculating an MCS $N^* \in \mathsf{MCS}_{q_0}(M_0)$ at which the minimum of (6.3) is reached, that is,

$$\mathcal{F}_{\mathtt{SUM}}(\{\{\theta(r) \mid \theta \in N^*\}\}) = \min_{N \in \mathsf{MCS}_{q_0}(M_0)} \mathcal{F}_{\mathtt{SUM}}(\{\{\theta(r) \mid \theta \in N\}\}).$$

Informally, to obtain such an MCS N^*, we must delete a $\subseteq$-minimal set of tuples from M_0 in order to satisfy $\{x \to y, yz \to r\}$, in a way that minimizes the $\mathtt{SUM}$ over the remaining r-values. Such deletions are represented in Fig. 6.2 by struck-through tuples.

The left-hand table of Fig. 6.2 shows how, in a first step, we delete tuples from M_0 in order to satisfy $xyz \to r$ (which is logically implied by $yz \to r$), in a way that minimizes the $\mathtt{SUM}$ over the remaining r-values: within each set of tuples that agree on xyz, we pick the one with the smallest r-value. The rationale for this step relies on the monotonicity of $\mathtt{SUM}$: smaller arguments will result in a smaller $\mathtt{SUM}$. This computation is readily expressed by the formula $\phi_1(x, y, z, r)$ in Fig. 6.3, where for readability we assume a vocabulary with $\leq$. In $\mathsf{AGGR[FOL]}$, $t_1(\vec{x}) \leq t_2(\vec{x})$ can be expressed as $t_1(\vec{x}) = \mathsf{Aggr}_{\mathcal{F}_{\mathtt{MIN}}} v\, [v, v = t_1(\vec{x}) \lor v = t_2(\vec{x})]$. In this simple example, it is evident that the remaining tuples will satisfy $yz \to r$, as desired, since the minimal value of r within each xyz-group is completely determined by yz, regardless of x. Proving that such independences hold in general is a major challenge in the general proof of Theorem 6.0.1.

The next step is to delete more tuples in order to also satisfy $x \to y$, which is illustrated by the right-hand table of Fig. 6.2. Within each group of tuples that agree on x, we pick the y-value that results in the smallest SUM. In the example, since $1+3 < 5$, the tuple with r-value 5 is deleted, indicated by a blue strike-through. There is a tie among the tuples where the x-value equals a_2. To break this tie, we opt for the smallest y-value according to lexicographical order. Specifically, b_2 is chosen over b_3, indicated by a red strike-through. This computation is expressed by the formula $\phi_2(x, y, z, r)$ in Fig. 6.3, where $\preceq$ is used for lexicographical order.

To conclude this example, we note that to calculate the value at the right-hand side of (6.3), there is actually no need to entirely compute the right-hand table of Fig. 6.2. We only need to know, for every set of tuples that share the same x-value, the total sum of their r-values. This is achieved by the formula ψ_2 in Fig. 6.3, which does not rely on lexicographical order. In our example, $\psi_2(a_1, 4)$ $(4 = 3+1)$ and $\psi_2(a_2, 5)$ hold true. Finally, $\mathsf{GLB\text{-}CQA}(g_0())$ is obtained by applying SUM over the second column of ψ_2, which yields 9 in our example. Note that associativity is needed to enable incremental aggregation.

6.3. Proof of Theorem 6.0.2

In this section we will provide a formal proof for Theorem 6.0.2. First, we introduce two helping Lemmas.

Lemma 6.3.1 (Decomposition Lemma)**.** *Let $g()$ be a numerical query in* $\mathsf{AGGR[sjfBCQ]}$*, with a head* $\mathtt{AGG}(r)$ *and* $q = \mathsf{BCQ}(g())$*, such that* $\mathcal{F}_{\mathtt{AGG}}$ *is monotone and associative, and the attack graph of q is acyclic. Let $(F_1, \dots, F_n)$ be a topological sort of q's attack graph. Let $\ell \in \{0, 1, \dots, n-1\}$. Let* **db** *be a database instance. Let θ be an ℓ-$\forall$embedding of q in* **db**. *Let m be the* $\mathcal{F}_{\mathtt{AGG}}$*-minimal value for θ in* **db** *as defined by Definition 5.0.2. Let $\gamma_1, \dots, \gamma_k$ enumerate all extensions of θ that are $(\ell{+}1)$-$\forall$key-embeddings of q in* **db**. *For $i \in \{1, \dots, k\}$, let N_i be an* $\mathcal{F}_{\mathtt{AGG}}$*-minimal MCS of* $\mathsf{Ext}(\gamma_i)$*, and define* $v_i := \mathcal{F}_{\mathtt{AGG}}(\{\{\mu(r) \mid \mu \in N_i\}\})$*. If* $\bigcup_{i=1}^{k} N_i \models \mathcal{K}(q)$*, then* $m = \mathcal{F}_{\mathtt{AGG}}(\{\{v_1, v_2, \dots, v_k\}\})$.

The proof for Lemma 6.3.1 is provided in Section C.7 of Appendix C. Let v be the $\mathcal{F}_{\mathtt{AGG}}$-minimal value for some ℓ-$\forall$embedding θ. Let $\gamma_1, \dots, \gamma_k$ enumerate all $(\ell{+}1)$-$\forall$key-embeddings of q in **db** such that each γ_i extends θ. For each $i \in \{1, 2, \dots, k\}$, let v_i be the $\mathcal{F}_{\mathtt{AGG}}$-minimal value for γ_i. Lemma 6.3.1 establishes that under some consistency hypothesis, we have $v = \mathcal{F}_{\mathtt{AGG}}(\{\{v_1, v_2, \dots, v_k\}\})$. Informally, this consistency hypothesis expresses that there is a *single* frugal repair **r** in which each v_i is attained, that is, for each $i \in \{1, 2, \dots, k\}$, $v_i =$

$\mathcal{F}_{\texttt{AGG}}\left(\{\{\mu(r) \mid \mu \text{ is an embedding extending } \gamma_i \text{ such that } \mu(q) \subseteq \mathbf{r}\}\}\right)$. The Consistent Extension Lemma (Lemma 6.3.2) will demonstrate that this consistency hypothesis can always be satisfied.

Lemma 6.3.2 (Consistent Extension Lemma)**.** *Let $g()$ be a numerical query in $\mathsf{AGGR[sjfBCQ]}$, with a head $\texttt{AGG}(r)$ and $q = \mathsf{BCQ}(g())$, such that $\mathcal{F}_{\texttt{AGG}}$ is monotone and associative, and the attack graph of q is acyclic. Let $(F_1, \dots, F_n)$ be a topological sort of q's attack graph. Let $\mathbf{db}$ be a database instance. Let $\ell \in \{1, 2, \dots, n\}$. Let $\gamma_1, \dots, \gamma_k$ be a sequence of ℓ-$\forall$key-embeddings of q in $\mathbf{db}$ such that $\{\gamma_1, \dots, \gamma_k\} \models \mathcal{K}(\{F_1, \dots, F_{\ell-1}\})$. For every $i \in \{1, \dots, k\}$, there is an $\mathcal{F}_{\texttt{AGG}}$-minimal MCS N_i of $\mathsf{Ext}(\gamma_i)$ such that $\bigcup_{i=1}^{k} N_i \models \mathcal{K}(q)$.*

The proof for Lemma 6.3.2 is provided in Section C.8 of Appedinx C. With these helping lemmas in place, we can now proceed with the proof of Theorem 6.0.2.

Proof of Theorem 6.0.2. Let $g()$ be a numerical query in $\mathsf{AGGR[sjfBCQ]}$, with a head $\texttt{AGG}(r)$ and $q = \mathsf{BCQ}(g())$, such that $\mathcal{F}_{\texttt{AGG}}$ is monotone and associative, and the attack graph of q is acyclic. We need to show that $\mathsf{GLB\text{-}CQA}(g())$ is expressible in $\mathsf{AGGR[FOL]}$. Let $\mathbf{db}$ be a database instance. By Lemma 6.1.2, $\forall\mathbf{Emb}_q(\mathbf{db})$ can be calculated in FOL. If $\forall\mathbf{Emb}_q(\mathbf{db}) = \emptyset$ and $\mathcal{F}_{\texttt{AGG}}(\emptyset)$ is undefined, by Corollary 6.2.1.1, we have $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}} = \bot$. Assume from here on that $\forall\mathbf{Emb}_q(\mathbf{db}) \neq \emptyset$ or $\mathcal{F}_{\texttt{AGG}}(\emptyset)$ is defined. By Corollary 6.2.1.1,

$$[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}} = \min_{N \in \mathsf{MCS}_q(\forall\mathbf{Emb}_q(\mathbf{db}))} \mathcal{F}_{\texttt{AGG}}\left(\{\{\theta(r) \mid \theta \in N\}\}\right). \tag{6.4}$$

The proof of Theorem 6.0.2 proceeds by showing that the right-hand expression of (6.4) can be expressed in $\mathsf{AGGR[FOL]}$. Let $(F_1, \dots, F_n)$ be a topological sort of q's attack graph. Let θ be an ℓ-$\forall$embedding of q in $\mathbf{db}$. We will show, by induction on decreasing $\ell = n, n-1, \dots, 0$, that the $\mathcal{F}_{\texttt{AGG}}$-minimal value for θ in $\mathbf{db}$ can be computed in $\mathsf{AGGR[FOL]}$. Note that for $\ell = 0$, we have $\theta = \varnothing$ and, by Corollary 6.2.1.1, the expression of the $\mathcal{F}_{\texttt{AGG}}$-minimal value for $\varnothing$ in $\mathbf{db}$ calculates $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}}$.

For the induction basis ($\ell = n$), we have $\mathsf{Ext}(\theta) = \{\theta\} = \mathsf{MCS}_q(\mathsf{Ext}(\theta))$. It follows that the $\mathcal{F}_{\texttt{AGG}}$-minimal value for θ in $\mathbf{db}$ is $\mathcal{F}_{\texttt{AGG}}\left(\{\{\theta(r)\}\}\right)$, which can obviously be computed in $\mathsf{AGGR[FOL]}$.

For the induction step ($\ell + 1 \to \ell$), the induction hypothesis is that for every $i \in \{\ell+1, \ell+2, \dots, n\}$, for every i-$\forall$embedding θ' of q in $\mathbf{db}$, the $\mathcal{F}_{\texttt{AGG}}$-minimal value for θ' in $\mathbf{db}$ can be computed in $\mathsf{AGGR[FOL]}$. Let $\gamma_1, \dots, \gamma_k$ enumerate all $(\ell+1)$-$\forall$key-embeddings of q in $\mathbf{db}$ that extend θ. By Lemma 6.3.2, for each $i \in \{1, 2, \dots, k\}$, we can assume an $\mathcal{F}_{\texttt{AGG}}$-minimal

MCS N_i of $\mathsf{Ext}(\gamma_i)$ such that $\bigcup_{i=1}^{k} N_i \models \mathcal{K}(q)$. For every $i \in \{1, \dots, k\}$, let $v_i := \mathcal{F}_{\mathtt{AGG}} (\{\{\mu(r) \mid \mu \in N_i\}\})$, that is, v_i is the $\mathcal{F}_{\mathtt{AGG}}$-minimal value for γ_i in **db**. By Lemma 6.3.1, $\mathcal{F}_{\mathtt{AGG}} (\{\{v_1, \dots, v_k\}\})$ is the $\mathcal{F}_{\mathtt{AGG}}$-minimal value for θ in **db**. We show in the next paragraph that for every $i \in \{1, \dots, k\}$, v_i can be computed in AGGR[FOL]. This suffices to show the theorem, as $\mathcal{F}_{\mathtt{AGG}}$ can be expressed in AGGR[FOL].

Let $i \in \{1, \dots, k\}$. Let $\theta_1^+, \theta_2^+, \dots, \theta_p^+$ enumerate all $(\ell + 1)$-$\forall$embeddings of q in **db** that extend γ_i. For every $j \in \{1, \dots, p\}$, let v_j^+ be the $\mathcal{F}_{\mathtt{AGG}}$-minimal MCS of θ_j^+. By the induction hypothesis, each v_j^+ can be computed in AGGR[FOL]. Since $\mathcal{K}(q) \models \mathsf{Key}(F_{\ell+1}) \rightarrow \mathsf{vars}(F_{\ell+1})$, it is clear that $\mathcal{F}_{\mathtt{MIN}}(\{\{v_1^+, v_2^+, \dots, v_p^+\}\})$ is the $\mathcal{F}_{\mathtt{AGG}}$-minimal value for γ_i in **db**. Since $\mathcal{F}_{\mathtt{MIN}}$ can be expressed in AGGR[FOL], it is correct to conclude that the $\mathcal{F}_{\mathtt{AGG}}$-minimal value for θ can be computed in AGGR[FOL]. $\square$

6.4. Proof of Theorem 6.0.1

We can finally provide the proof of Theorem 6.0.1.

Proof of Proof of Theorem 6.0.1. Let $g()$ be a numerical query in AGGR[sjfBCQ] with a head $\mathtt{AGG}(r)$ and $q = \mathsf{BCQ}(g())$ and such that $\mathcal{F}_{\mathtt{AGG}}$ is monotone and associative. If the attack graph of q is cyclic, then GLB-CQA$(g())$ is not in AGGR[FOL] by Theorem 3.5.6. If the attack graph of q is acyclic, then GLB-CQA$(g())$ is expressible in AGGR[FOL] by Theorem 6.0.2.

It remains to establish the upper bounds on the time complexities stated in Theorem 6.0.1. Acyclicity of the attack graph of q can be tested using the QuadAttack algorithm described in (Wijsen, 2012, Section 9), which runs in quadratic time. Assuming that the the attack graph of q is acyclic, the proof of Theorem 6.0.2 constructs the expression for GLB-CQA$(g())$ in two steps: first, a FOL formula for the set of $\forall$embeddings, followed by a AGGR[FOL] formula. The construction of the former formula is in quadratic time by Lemma 6.1.2, and the proof of Theorem 6.0.2 shows that the latter formula can be constructed in linear time (with respect to the length of $g()$). $\square$

6.5. Databases Annotated with Semiring Values

In Kolaitis et al. (2025), all facts in a database instance are annotated with elements of a *naturally ordered positive semiring* $\mathbb{K} = (K, +, \times, 0, 1)$. The authors call $\mathbb{K}$ positive if $a + b = 0$ implies $a = 0$ and $b = 0$, and $a \times b = 0$

implies $a = 0$ or $b = 0$. In our setting, annotations can be captured by defining an *annotated database instance* as a pair $(\mathbf{db}, \alpha)$ where α maps each fact of the database instance $\mathbf{db}$ to a value in $K \setminus \{0\}$. We extend this *annotation function* to all facts by mapping facts not in $\mathbf{db}$ to 0. We now briefly explain the problem studied in Kolaitis et al. (2025), and compare one of their main findings with ours.

Let q be a self-join-free Boolean conjunctive query whose set of atoms is $\{F_1, F_2, \dots, F_n\}$. The function α gives rise to a mapping $\mathsf{evalVal}_{q,\alpha}$ from valuations over $\mathsf{vars}(q)$ to K, and a mapping $\mathsf{eval}_{q,\alpha}$ from database instances to K as follows. For every valuation θ over $\mathsf{vars}(q)$, we define

$$\mathsf{evalVal}_{q,\alpha}(\theta) := \prod_{i=1}^{n} \alpha(\theta(F_i)), \tag{6.5}$$

where $\prod$ refers to multiplication in the semiring. For every subset $\mathbf{s}$ of $\mathbf{db}$, we define

$$\mathsf{eval}_{q,\alpha}(\mathbf{s}) := \sum_{\theta \in \mathbf{Emb}_q(\mathbf{s})} \mathsf{evalVal}_{q,\alpha}(\theta), \tag{6.6}$$

where $\sum$ refers to addition in the semiring. Significantly, the summation in (6.6) ranges over $\mathbf{Emb}_q(\mathbf{s})$. Note that if $\theta \in \mathbf{Emb}_q(\mathbf{s})$ and $\mathbf{s} \subseteq \mathbf{db}$, then $\theta \in \mathbf{Emb}_q(\mathbf{db})$, hence $\alpha(\theta(F_i)) \neq 0$ for every i, and therefore $\mathsf{evalVal}_{q,\alpha}(\theta) \neq 0$. The aim in Kolaitis et al. (2025) is to determine the value

$$\min_{\mathbf{r} \in \mathsf{rset}(\mathbf{db})} \mathsf{eval}_{q,\alpha}(\mathbf{r}). \tag{6.7}$$

Their main result (Kolaitis et al., 2025, Theorem 4.9) shows that acyclicity of q's attack graph is a necessary and sufficient condition for (6.7) to be computable in a logic called $\mathcal{L}_{\mathbb{K}}$, which is reminiscent of what we have proved in this chapter. We next sketch why this is not merely a superficial similarity.

Let $\vec{u}$ be a shortest sequence containing every variable of $\mathsf{vars}(q)$. Consider a logic in which we can express a predicate $\nu(\vec{u}, r)$, where r is a variable ranging over K, with the following interpretation relative to an annotation function α: for every sequence $\vec{c}$ of constants of the same length as $\vec{u}$, $\nu(\vec{c}, k)$ holds if and only if $k = \mathsf{evalVal}_{q,\alpha}(\theta)$, where θ is the valuation satisfying $\theta(\vec{u}) = \vec{c}$. As argued before, we can also construct a FOL formula $\psi(\vec{u})$ such that $\mathbf{db} \models \psi(\vec{c})$ if and only if the valuation mapping $\vec{u}$ to $\vec{c}$ is a $\forall$embedding of q in $\mathbf{db}$. Finally, define

$$\phi_0(\vec{u}, r) := \psi(\vec{u}) \wedge \nu(\vec{u}, r).$$

Informally, ϕ_0 associates each $\forall$embedding with its semiring value defined by (6.5). Then, the AGGR[FOL] rewriting established in the proof of Theorem 6.0.1 (illustrated in Fig. 6.1) can be applied. Here, `SUM` is interpreted relative to the additive operation of the semiring, yielding a monotone and associative aggregate operator, while `MAX` is interpreted relative to the natural order of the semiring. If $(F_1, \dots, F_n)$ is a topological sort of q's attack graph, then this rewriting yields precisely (6.7). On the other hand, if the attack graph of q has a cycle, then (6.7) cannot be computed in any Hanf-local logic.

CHAPTER 7

LUB-CQA with Monotone, Associative Aggregate Operators

A natural question is whether the results provided in Chapter 6 can also be applied to LUB-CQA($g()$) for numerical queries $g()$ using aggregate operators that are both monotone and associative. We will see in this chapter that Theorem 6.0.2 fails if GLB-CQA($g()$) is replaced with LUB-CQA($g()$), while keeping all other conditions the same. That is, acyclicity of the attack graph is not a sufficient condition for LUB-CQA($g()$) to be expressible in AGGR[FOL]. To address this, we will introduce *κ-acyclicity*, a polynomial-time decidable syntactic property of sjfBCQ queries. Our main result can then be stated as follows:

Theorem 7.0.1 (Separation Theorem for SUM)**.** *Let $g()$ be a numerical query in* AGGR[sjfBCQ]*, with a head* SUM(r) *and $q =$* BCQ($g()$)*. Then,* LUB-CQA($g()$) *is expressible in* AGGR[FOL] *if and only if q is κ-acyclic.*

Our second main result is that the right-to-left implication in Theorem 7.0.1 extends from SUM-queries to all AGG-queries in AGGR[sjfBCQ] whose aggregate operator is monotone and associative.

Theorem 7.0.2. *Let $g()$ be a numerical query in* AGGR[sjfBCQ]*, with a head* AGG(r) *and $q =$* BCQ($g()$)*, such that $\mathcal{F}_{\texttt{AGG}}$ is monotone and associative. If q is κ-acyclic, then* LUB-CQA($g()$) *can be expressed by a query in* AGGR[FOL] *(and such a query can be effectively constructed in polynomial time in the size of q).*

The inverse of Theorem 7.0.2 does not generally hold. For instance, the problem LUB-CQA($g()$) is expressible in AGGR[FOL] for all MAX-queries in

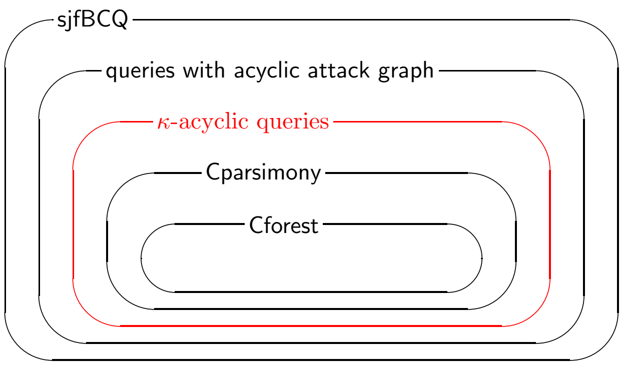

Figure 7.1: Subclasses of self-join-free Boolean conjunctive queries. The class of κ-acyclic queries, introduced in the current chapter, is the largest class that allows lub rewriting for SUM-queries. The class of queries with an acyclic attack graph is the largest class that allows glb rewriting.

AGGR[sjfBCQ], with $\mathcal{F}_{\texttt{MAX}}$ being monotone and associative. This is because LUB-CQA($g()$) is trivial if $g()$ is a MAX-query: the maximal result of a MAX-query across all repairs of a database is equal to the result of the MAX-query on the original database.

Our results add κ-acyclic queries to the query classes in Fig. 7.1 and prove the correctness of the inclusions shown. In Chapter 4, we saw that lub rewriting of SUM-queries is possible for the classes Cforest and Cparsimony over some database instances. The notion of κ-acyclicity is new and identifies the largest class of queries that allow lub rewriting of SUM-queries. Interestingly, Theorem 6.0.1 and our results show that there are SUM-queries that allow glb rewriting but not lub rewriting, specifically those that are not κ-acyclic but have an acyclic attack graph.

This chapter is organized as follows. In Section 7.1, we introduce a key auxiliary result, the *Reduction Lemma*. Section 7.2 introduces the new notion of κ-acyclicity and establishes some of its properties. The expressibility result of Theorem 7.0.2 is discussed in Section 7.3. Section 7.4 establishes the inexpressibility direction of Theorem 7.0.1, showing that lub rewriting in AGGR[FOL] is impossible for SUM-queries that lack κ-acyclicity. Finally, Section 7.5 shows that all queries in Cparsimony are κ-acyclic.

7.1. The Reduction Lemma

In this section we present an important result, called the *Reduction Lemma*, which is stated next.

Lemma 7.1.1 (Reduction Lemma). *Let $g_1()$ and $g_2()$ be two $\mathtt{AGG}$-queries in $\mathsf{AGGR[sjfBCQ]}$ with the same head. Assume that $\mathcal{F}_{\mathtt{AGG}}$ is monotone, and that $\mathcal{F}_{\mathtt{AGG}}(\emptyset)$ is defined. Let $q_1 = \mathsf{BCQ}(g_1())$, and $q_2 = \mathsf{BCQ}(g_2())$. If $q_1 \preccurlyeq_{\mathsf{g}} q_2$ and $\mathcal{K}(q_1) \equiv \mathcal{K}(q_2)$, then $\mathsf{LUB\text{-}CQA}(g_1()) \leq_{\mathsf{nr\text{-}datalog}} \mathsf{LUB\text{-}CQA}(g_2())$.*

Before proving the Reduction Lemma, we illustrate it with an example and show that it does not hold for greatest lower bounds.

Example 7.1.1. Let $g_1 := \mathtt{SUM}(1) \leftarrow q_0$ and $g_2 := \mathtt{SUM}(1) \leftarrow q_0'$ with q_0 and q_0' as shown in Fig. 2.1. Clearly, $q_0 \sim_{\mathsf{g}} q_0'$. Both queries have the same Gaifman graph (specifically, a triangle involving x, y, and z), and the same set of induced functional dependencies (notably, $\{xy \to z, z \to x\}$). The Reduction Lemma implies that $\mathsf{LUB\text{-}CQA}(g_1())$ and $\mathsf{LUB\text{-}CQA}(g_2())$ are equivalent under first-order reductions. However, this equivalence does not hold for glb: indeed, by Theorem 3.5.6 and Theorem 6.0.2, $\mathsf{GLB\text{-}CQA}(g_1())$ is in $\mathsf{AGGR[FOL]}$ but $\mathsf{GLB\text{-}CQA}(g_2())$ is not. ◁

Proof of Lemma 7.1.1. Assume $q_1 \preccurlyeq_{\mathsf{g}} q_2$ and $\mathcal{K}(q_1) \equiv \mathcal{K}(q_2)$. Since $q_1 \preccurlyeq_{\mathsf{g}} q_2$, we have $\mathsf{vars}(q_1) \subseteq \mathsf{vars}(q_2)$. For simplicity, we first show the lemma under the hypothesis $\mathsf{vars}(q_1) = \mathsf{vars}(q_2)$, and then indicate how the proof extends to the case $\mathsf{vars}(q_1) \subsetneq \mathsf{vars}(q_2)$. Let $\mathbf{db}$ be a database instance. Let

$$\mathbf{db}' := \{\theta(G) \mid \theta \in \mathbf{Emb}_{q_1}(\mathbf{db}), G \text{ atom in } q_2\}. \tag{7.1}$$

Clearly, we have that $\mathbf{db}'$ is computable in $\mathsf{nr\text{-}datalog}$. We need to show $[\![\mathsf{LUB\text{-}CQA}(g_1())]\!]^{\mathbf{db}} = [\![\mathsf{LUB\text{-}CQA}(g_2())]\!]^{\mathbf{db}'}$.

Claim 1. $\mathbf{Emb}_{q_1}(\mathbf{db}) = \mathbf{Emb}_{q_2}(\mathbf{db}')$

Proof of Claim 1. The inclusion $\mathbf{Emb}_{q_1}(\mathbf{db}) \subseteq \mathbf{Emb}_{q_2}(\mathbf{db}')$ holds straightforwardly true by construction. We show $\mathbf{Emb}_{q_2}(\mathbf{db}') \subseteq \mathbf{Emb}_{q_1}(\mathbf{db})$ in the remainder of the proof. To this end, let $\mu \in \mathbf{Emb}_{q_2}(\mathbf{db}')$. Assume, for the sake of contradiction, that $\mu \notin \mathbf{Emb}_{q_1}(\mathbf{db})$. Then there exists an atom F in q_1 such that $\mu(F) \notin \mathbf{db}$. Since $q_1 \preccurlyeq_{\mathsf{g}} q_2$, there exists an atom G in q_2 with $\mathsf{vars}(F) \subseteq \mathsf{vars}(G)$. Because $\mu \in \mathbf{Emb}_{q_2}(\mathbf{db}')$, we have $\mu(G) \in \mathbf{db}'$. By the construction of $\mathbf{db}'$, there exists an embedding $\theta \in \mathbf{Emb}_{q_1}(\mathbf{db})$ such that $\theta(G) = \mu(G)$. Now, $\mathsf{vars}(F) \subseteq \mathsf{vars}(G)$ implies $\mu(F) = \theta(F)$, and since

$\theta \in \mathbf{Emb}_{q_1}(\mathbf{db})$, we have $\theta(F) \in \mathbf{db}$. This contradicts our assumption that $\mu(F) \notin \mathbf{db}$. Hence, we conclude that $\mu \in \mathbf{Emb}_{q_1}(\mathbf{db})$, as desired. This concludes the proof of Claim 1. $\square$

Since $\mathcal{K}(q_1) \equiv \mathcal{K}(q_2)$, every MCS of $\mathbf{Emb}_{q_1}(\mathbf{db})$ with respect to $\mathcal{K}(q_1)$ is also an MCS of $\mathbf{Emb}_{q_2}(\mathbf{db}')$ with respect to $\mathcal{K}(q_2)$. By Theorem 5.0.4, $[\![\mathsf{LUB\text{-}CQA}(g_1())]\!]^{\mathbf{db}} = [\![\mathsf{LUB\text{-}CQA}(g_2())]\!]^{\mathbf{db}'}$.

The preceding proof can be easily adapted to the case where $\mathsf{vars}(q_1) \subsetneq \mathsf{vars}(q_2)$ by mapping every variable $v \in \mathsf{vars}(q_2) \setminus \mathsf{vars}(q_1)$ to the same constant c in the construction of $\mathbf{db}'$. This concludes the proof of Lemma 7.1.1. $\square$

7.2. Kernels and κ-Acyclicity

We assume that the reader is familiar with the notion of a *minimal cover* of a set of functional dependencies (FDs), which can be computed in polynomial time (Maier, 1980)(Abiteboul et al., 1995, p. 257). To recall briefly, a set Σ of FDs is a *minimal cover* (a.k.a. *irreducible*) if each functional dependency σ in Σ has the form $X \to A$, where A is an attribute, and σ is *left-reduced* and *non-redundant*. A set Σ of FDs is a minimal cover of another set Σ' of FDs if Σ is a minimal cover and $\Sigma \equiv \Sigma'$. We now extend this notion of irreducibility to sjfBCQ queries, introduce the notion of a *kernel*, and present some auxiliary lemmas.

Definition 7.2.1 (Irreducible query and Kernel)**.** Let q be query in sjfBCQ, and let $\widehat{q}$ denote the set of atoms in q that are not full-key. We say that q is *irreducible* if the following two conditions hold:

1. for every atom F of $\widehat{q}$, we have $\mathcal{K}(\widehat{q} \setminus \{F\}) \not\equiv \mathcal{K}(\widehat{q})$; and
2. the set $\{\mathsf{Key}(F) \to \mathsf{notKey}(F) \mid F \in \widehat{q}\}$ is irreducible .

A *kernel of* q is an irreducible query q' in sjfBCQ such that $\mathcal{K}(q') \equiv \mathcal{K}(q)$ and $q' \sim_{\mathsf{g}} q$.

$\square$

Example 7.2.1. Consider the queries $q_0 = \exists x \exists y \exists z (R(\underline{x,y},z), S(\underline{z},x), T(\underline{z},x))$ and $q_0' = \exists x \exists y \exists z (R(\underline{x,y},z),\ S(\underline{z},x),\ T'(\underline{z,x}))$ in Fig. 2.1. The query q_0 is not irreducible, because it does not satisfy the first condition in Definition 7.2.1. The query q_0' is irreducible. Since $\mathcal{K}(q_0) \equiv \mathcal{K}(q_0') \equiv \{xy \to z, z \to x\}$ and $q_0 \sim_{\mathsf{g}} q_0'$, it follows that q_0' is a kernel of q_0. Notably, the attack graph of q_0 is acyclic, whereas the attack graph of its kernel contains a cycle. $\triangleleft$

Lemma 7.2.1. *Let q be a query in* sjfBCQ *that is irreducible. Let G be an atom in q that is not full-key. Then, there is a variable x such that* $\mathsf{notKey}(G) = \{x\}$ *and* $G \overset{q}{\rightsquigarrow} x$.

Proof. By the second item in Definition 7.2.1, $|\mathsf{notKey}(G)| = 1$. Hence, we can assume x such that $\mathsf{notKey}(G) = \{x\}$. Let $\widehat{q}$ denote the set of atoms in q that are not full-key. By the first item in Definition 7.2.1, $\mathcal{K}(\widehat{q} \setminus \{G\}) \not\equiv \mathcal{K}(\widehat{q})$. It must obviously be the case that $\mathcal{K}(\widehat{q} \setminus \{G\}) \not\models \mathsf{Key}(G) \to x$, which implies $G \overset{q}{\rightsquigarrow} x$. □

Lemma 7.2.2. *Let q be a query in* sjfBCQ. *A kernel of q can be constructed in polynomial time in the size of q.*

Proof. Let Σ be a minimal cover of $\mathcal{K}(q)$ such that for every $X \to y$ in Σ, there is an atom F in q such that $X \cup \{y\} \subseteq \mathsf{vars}(F)$. It is easily verified that such a Σ can be computed in polynomial time by a standard algorithm for minimal covers; see (Maier, 1983, Chapter 5). Let q' be a query in sjfBCQ constructed as follows:

(a) q' contains every full-key atom of q;

(b) for every atom $R(\underline{\vec{x}}, \vec{y})$ in q that is not full-key, q' contains a full-key atom $R'(\underline{\vec{x}, \vec{y}})$;

(c) for every $X \to \{y\}$ in Σ, q' contains an atom $S(\underline{\vec{x}}, y)$ such that $\mathsf{vars}(\vec{x}) = X$, where S is a fresh relation name. If $X = \emptyset$, then $\vec{x} = c$, for some constant c; and

(d) q' contains no atoms other than those specified in ((a)), ((b)), and ((c)).

It is easy to verify that this construction runs in polynomial time and ensures that q' is irreducible, $q' \sim_{\mathsf{g}} q$, and $\mathcal{K}(q') \equiv \mathcal{K}(q)$. Hence, q' is a kernel of q. □

Example 7.2.1 illustrates that an sjfBCQ query with an acyclic attack graph can have a kernel whose attack graph contains a cycle. Informally, this means that cycles in attack graphs can emerge during the construction of a kernel but, as stated in Proposition 7.2.4, they cannot disappear. Moreover, the following lemma states that all kernels of the same query either all have an acyclic attack graph or all have a cyclic attack graph.

Lemma 7.2.3. *Let q be a query in* sjfBCQ. *If some kernel of q has an acyclic attack graph, then every kernel of q has an acyclic attack graph.*

The proof of Lemma 7.2.3 in Section D.1 of Appendix D also establishes the following result: if some kernel of q has an acyclic attack graph, then all kernels agree on their atoms that are not full-key, up to the choice of relation names. Lemma 7.2.3 allows the following definition.

Definition 7.2.2 (κ-acyclic)**.** A query q in sjfBCQ is κ-acyclic if it has a kernel with an acyclic attack graph (and hence, by Lemma 7.2.3, all of its kernels have acyclic attack graphs). □

As illustrated in the Venn diagram in Fig. 7.1, κ-acyclic queries have acyclic attack graphs:

Proposition 7.2.4. *Every κ-acyclic query in* sjfBCQ *has an acyclic attack graph.*

Two different proofs for Proposition 7.2.4 are provided in Section D.2 of Appendix D.

7.3. Expressibility Result (Theorem 7.0.2)

Theorem 7.0.2 uses AGGR[FOL] as the target language for solving the problem LUB-CQA($g()$). The following lemma uses a weaker logic that yields more readable rewritings, namely AGGR[nr-datalog]. The language nr-datalog, unlike FOL, does not include negation. As it turns out, if LUB-CQA($g()$) is expressible in AGGR[FOL] according to Theorem 7.0.2, and moreover $\mathcal{F}_{\texttt{AGG}}(\emptyset)$ is defined, then LUB-CQA($g()$) is also expressible in AGGR[nr-datalog]. However, if $\mathcal{F}_{\texttt{AGG}}(\emptyset)$ is not defined, then LUB-CQA($g()$) is defined to return $\perp$ on "no"-instances of CERTAINTY(q), where q = BCQ($g()$). Distinguishing between "yes"-instances and "no"-instances generally requires negation, because CERTAINTY(q) is typically non-monotonic: a "yes"-instance may have an extension that is a "no"-instance.

Lemma 7.3.1. *Let $g()$ be a numerical query in* AGGR[sjfBCQ]*, with a head* AGG(r) *and q =* BCQ($g()$)*. Assume that $\mathcal{F}_{\texttt{AGG}}$ is monotone, associative, and that $\mathcal{F}_{\texttt{AGG}}(\emptyset)$ is defined. If q is irreducible and has an acyclic attack graph, then an* AGGR[nr-datalog] *program solving* LUB-CQA($g()$) *can be constructed in polynomial time in the size of q.*

The proof of Lemma 7.3.1 is provided in Section 7.3.1. Then, in Section 7.3.2, we provide the proof for Theorem 7.0.2. Finally, in Section 7.3.3, we illustrate the rewriting in AGGR[nr-datalog] using a concrete example and show that there is potential for making the rewriting more efficient. However, in this thesis, our focus is solely on rewritability in $\mathsf{AGGR}[\mathsf{nr\text{-}datalog}]$, without addressing the optimization of such rewritings.

7.3.1 Proof of Lemma 7.3.1

First, we introduce two helping Lemmas which are variations of the Decomposition Lemma and Consistent Extension Lemma introduced in Section 6.3 of Chapter 6. The main variation with respect to the original lemmas are that we consider irreducible queries (rather than queries with an acyclic attack graph) and ℓ-embeddings (rather than ℓ-$\forall$embeddings).

Lemma 7.3.2 (Decomposition Lemma for $\mathsf{LUB\text{-}CQA}$)**.** *Let $\exists\vec{u}(q(\vec{u}))$ be an irreducible* sjfBCQ *query with an acyclic attack graph, topologically sorted as $(F_1, \ldots, F_n)$. Let $g()$ be the* $\mathsf{AGGR}[\mathsf{sjfBCQ}]$ *query defined as $g() := \mathtt{AGG}(r) \leftarrow q(\vec{u})$, where $\mathcal{F}_{\mathtt{AGG}}$ is monotone and associative. Let* **db** *be a database instance. Let $\ell \in \{0, \ldots, n-1\}$. Let θ be an ℓ-embedding of q in* **db**, *and let m be the $\mathcal{F}_{\mathtt{AGG}}$-maximal value for θ in* **db**. *Let $\gamma_1, \ldots, \gamma_k$ enumerate all extensions of θ that are (ℓ+1)-key-embeddings of q in* **db**. *For $i \in \{1, \ldots, k\}$, let N_i be an $\mathcal{F}_{\mathtt{AGG}}$-maximal MCS of $\mathsf{Ext}(\gamma_i)$, and define $v_i := \mathcal{F}_{\mathtt{AGG}}(\{\{\mu(r) \mid \mu \in N_i\}\})$. If $\bigcup_{i=1}^{k} N_i \models \mathcal{K}(q)$, then $m = \mathcal{F}_{\mathtt{AGG}}(\{\{v_1, v_2, \ldots, v_k\}\})$.*

Lemma 7.3.3 (Consistent Extension Lemma for $\mathsf{LUB\text{-}CQA}$)**.** *Let $\exists\vec{u}(q(\vec{u}))$ be an irreducible* sjfBCQ *query with an acyclic attack graph, topologically sorted as $(F_1, \ldots, F_n)$. Let $g()$ be the* $\mathsf{AGGR}[\mathsf{sjfBCQ}]$ *query defined as $g() := \mathtt{AGG}(r) \leftarrow q(\vec{u})$, where $\mathcal{F}_{\mathtt{AGG}}$ is monotone and associative. Let* **db** *be a database instance. Let $\ell \in \{1, 2, \ldots, n\}$. Let $\gamma_1, \ldots, \gamma_k$ be a sequence of ℓ-key-embeddings of q in* **db** *such that $\{\gamma_1, \ldots, \gamma_k\} \models \mathcal{K}(\{F_1, \ldots, F_{\ell-1}\})$. For every $i \in \{1, \ldots, k\}$, there is an $\mathcal{F}_{\mathtt{AGG}}$-maximal MCS N_i of $\mathsf{Ext}(\gamma_i)$ such that $\bigcup_{i=1}^{k} N_i \models \mathcal{K}(q)$.*

The proof for both lemmas are provided in Section D.3 in Appendix D. We can now proceed with the proof of Lemma 7.3.1.

Proof of Lemma 7.3.1. Let $g()$ be a numerical query in $\mathsf{AGGR}[\mathsf{sjfBCQ}]$, with a head $\mathtt{AGG}(r)$ and $q = \mathsf{BCQ}(g())$. Assume that q is irreducible and has an acyclic attack graph. We need to show that $\mathsf{LUB\text{-}CQA}(g())$ is expressible in $\mathsf{AGGR}[\mathsf{nr\text{-}datalog}]$. Let **db** be a database instance. By Theorem 5.0.4,

$$[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}} = \max_{N \in \mathsf{MCS}_q(\mathbf{Emb}_q(\mathbf{db}))} \mathcal{F}_{\mathtt{AGG}}(\{\{\theta(r) \mid \theta \in N\}\}). \tag{7.2}$$

The proof of Lemma 7.3.1 proceeds by showing that the right-hand expression of (7.2) can be expressed in $\mathsf{AGGR}[\mathsf{nr\text{-}datalog}]$. The technical treatment follows the outline of the proof of Theorem 6.0.2, but requires novel theoretical notions due to the difference between glb and lub.

Let $(F_1, \dots, F_n)$ be a topological sort of q's attack graph. For every $\ell \in \{1, \dots, n\}$, we use the notation $\vec{u}_\ell = (\vec{u}_{\ell-1}, \vec{x}_\ell, \vec{y}_\ell)$ defined in Chapter 2. Let $q(\vec{u})$ be the self-join-free conjunction of atoms in the body of $g()$. Note that $\vec{u}_n = \vec{u}$. Consider the $\mathsf{AGGR}[\textsf{nr-datalog}]$ program built as follows. The program uses a fresh IDB predicate E_0 of arity 2, and for every $\ell \in \{1, 2, \dots, n\}$, it uses two fresh IDB predicates: E_ℓ of arity $|\vec{u}_\ell| + 1$, and $KeyE\_\ell$ of arity $|\vec{u}_{\ell-1}| + |\vec{x}_\ell| + 1$. The IDB predicate E_n is defined as follows:

$$E\_n(\vec{u}, \texttt{AGG}(r_n)) \leftarrow q(\vec{u}), r_n = r \tag{7.3}$$

For every $\ell \in \{1, \dots, n\}$, the IDB predicates $KeyE\_\ell$ and $E\_(\ell - 1)$ are defined as follows:

$$KeyE\_\ell(\vec{u}_{\ell-1}, \vec{x}_\ell, \texttt{MAX}(r_\ell)) \leftarrow E\_\ell(\vec{u}_\ell, r_\ell) \tag{7.4}$$

$$E\_(\ell - 1)(\vec{u}_{\ell-1}, \texttt{AGG}(m_\ell)) \leftarrow KeyE\_\ell(\vec{u}_{\ell-1}, \vec{x}_\ell, m_\ell) \tag{7.5}$$

By inspecting the program, one can see that for every ℓ-embedding θ of q in $\mathbf{db}$, there exists c such that $\mathbf{db} \models E\_\ell(\theta(\vec{u}_\ell), c)$. We will show, by induction on decreasing $\ell = n, n-1, \dots, 0$, that the following holds for every ℓ:

for every ℓ-embedding θ of q in $\mathbf{db}$,

$$\mathbf{db} \models E\_\ell(\theta(\vec{u}_\ell), c) \text{ iff } c \text{ is the } \mathcal{F}_{\texttt{AGG}}\text{-maximal value for } \theta \text{ in } \mathbf{db}. \tag{7.6}$$

For $\ell = 0$, we obtain that $\mathbf{db} \models E\_0((), c_0)$ if and only if c_0 is the $\mathcal{F}_{\texttt{AGG}}$-maximal value for $\varnothing$ in $\mathbf{db}$, and hence, by Theorem 5.0.4, we have $c_0 = [\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}}$.

To show (7.6), let θ be an arbitrary ℓ-embedding of q in $\mathbf{db}$. For the induction basis, $\ell = n$, we have $\mathsf{Ext}(\theta) = \{\theta\} = \mathsf{MCS}_q(\mathsf{Ext}(\theta))$. It follows that the $\mathcal{F}_{\texttt{AGG}}$-maximal value for θ in $\mathbf{db}$ is $\mathcal{F}_{\texttt{AGG}}(\{\{\theta(r)\}\})$. The program rule (7.3) ensures that $\mathbf{db} \models E\_n(\theta(\vec{u}), c)$ if and only if $c = \mathcal{F}_{\texttt{AGG}}(\{\{\theta(r)\}\}))$.

For the induction step, $\ell + 1 \to \ell$, let $\gamma_1, \dots, \gamma_k$ enumerate all $(\ell+1)$-key-embeddings of q in $\mathbf{db}$ that extend θ. By Lemma 7.3.3, for each $i \in \{1, 2, \dots, k\}$, we can assume an $\mathcal{F}_{\texttt{AGG}}$-maximal MCS N_i of $\mathsf{Ext}(\gamma_i)$ such that $\bigcup_{i=1}^{k} N_i \models \mathcal{K}(q)$. For every $i \in \{1, \dots, k\}$, let $v_i := \mathcal{F}_{\texttt{AGG}}(\{\{\mu(r) \mid \mu \in N_i\}\})$, that is, v_i is the $\mathcal{F}_{\texttt{AGG}}$-maximal value for γ_i in $\mathbf{db}$. By Lemma 7.3.2, we have that $\mathcal{F}_{\texttt{AGG}}(\{\{v_1, \dots, v_k\}\})$ is the $\mathcal{F}_{\texttt{AGG}}$-maximal value for θ in $\mathbf{db}$. By program rule (7.5), the following statements are equivalent:

(a) for every $i \in \{1, \dots, k\}$, $\mathbf{db} \models KeyE\_\ell(\gamma_i(\vec{u}_{\ell-1}), \gamma_i(\vec{x}_\ell), c_i)$ if and only if c_i is the $\mathcal{F}_{\texttt{AGG}}$-maximal value for γ_i in $\mathbf{db}$; and

(b) $\mathbf{db} \models E\_(\ell-1)(\theta(\vec{u}_{\ell-1}), c)$ if and only if c is the $\mathcal{F}_{\mathtt{AGG}}$-maximal value for θ in $\mathbf{db}$.

Therefore, to show the desired result ((b)), it suffices to show ((a)).

To show ((a)), let $i \in \{1, \dots, k\}$. Let $\theta_1^+, \theta_2^+, \dots, \theta_p^+$ enumerate all ℓ-embeddings of q in $\mathbf{db}$ that extend γ_i. For every $j \in \{1, \dots, p\}$, let v_j^+ be the $\mathcal{F}_{\mathtt{AGG}}$-maximal MCS of θ_j^+. By the induction hypothesis (7.6) for ℓ, for each $j \in \{1, 2, \dots, p\}$,

$$\mathbf{db} \models E\_\ell(\theta_j^+(\vec{u}_\ell), c) \text{ if and only if } c = v_j^+. \tag{7.7}$$

Since $\mathcal{K}(q) \models \mathsf{Key}(F_\ell) \rightarrow \mathsf{vars}(F_\ell)$, it is clear that $\mathcal{F}_{\mathtt{MAX}}(\{\{v_1^+, v_2^+, \dots, v_p^+\}\})$ is the $\mathcal{F}_{\mathtt{AGG}}$-maximal value for γ_i in $\mathbf{db}$. From program rule (7.4), it follows that ((b)) holds true. This concludes the proof of Lemma 7.3.1. □

7.3.2 Proof of Theorem 7.0.2

We can now provide the proof of Theorem 7.0.2.

Proof of Theorem 7.0.2. Assume that q is κ-acyclic. By Lemma 7.2.2, it is possible to construct, in polynomial time in the size of q, a kernel q' of q. Let $g'()$ be a numerical query in $\mathsf{AGGR[sjfBCQ]}$ with a head $\mathtt{AGG}(r)$ and such that $\mathsf{BCQ}(g'()) = q'$. We consider two cases:

Case that $\mathcal{F}_{\mathtt{AGG}}(\emptyset)$ is defined. By Lemma 7.1.1, $\mathsf{LUB\text{-}CQA}(g()) \leq_{\mathsf{nr\text{-}datalog}} \mathsf{LUB\text{-}CQA}(g'())$. By Definition 7.2.1, q' is irreducible. Since q is κ-acyclic, the attack graph of q' is acyclic. By Lemma 7.3.1, $\mathsf{LUB\text{-}CQA}(g'())$ is expressible in $\mathsf{AGGR[nr\text{-}datalog]}$. Consequently, $\mathsf{LUB\text{-}CQA}(g())$ can be solved in $\mathsf{AGGR[nr\text{-}datalog]}$ by first reducing it to $\mathsf{LUB\text{-}CQA}(g'())$ using the $\mathsf{nr\text{-}datalog}$ reduction of Lemma 7.1.1, and then solving $\mathsf{LUB\text{-}CQA}(g'())$.

Case that $\mathcal{F}_{\mathtt{AGG}}(\emptyset)$ is undefined. By Proposition 7.2.4, the attack graph of q is acyclic. By Theorem 2.0.1, one can test in $\mathsf{AGGR[FOL]}$ whether the database instance input to $\mathsf{LUB\text{-}CQA}(g())$ is a "no"-instance of the problem $\mathsf{CERTAINTY}(q)$ and, if so, return $\perp$ as the answer to $\mathsf{LUB\text{-}CQA}(g())$. The remaining case is when the input is a "yes"-instance of $\mathsf{CERTAINTY}(q)$. The proofs of Lemmas 7.1.1 and 7.3.1 rely on Eq. (5.2) from Theorem 5.0.4, which applies if $\mathcal{F}_{\mathtt{AGG}}(\emptyset)$ is defined or if the input database instance is a "yes"-instance of $\mathsf{CERTAINTY}(q)$. The requirement that $\mathcal{F}_{\mathtt{AGG}}(\emptyset)$ is defined is built into the statements of these lemmas. However, they remain valid even when $\mathcal{F}_{\mathtt{AGG}}(\emptyset)$ is undefined, as long as the

R

$\underline{x}$	y
a_1	b_1
a_1	b_2
a_2	b_2

S

$\underline{y}$	$\underline{z}$	r
b_1	c_1	1
b_1	c_1	4
b_1	c_2	3
b_2	c_3	5
b_2	c_3	6

E_2

x	y	z	r	$r_2 := r$
a_1	b_1	c_1	1	1
a_1	b_1	c_1	4	4
a_1	b_1	c_2	3	3
a_1	b_2	c_3	5	5
a_1	b_2	c_3	6	6
a_2	b_2	c_3	6	5
a_2	b_2	c_3	6	6

KeyE_2

x	y	z	$m_2 :=$ MAX(r_2)
a_1	b_1	c_1	4
a_1	b_1	c_2	3
a_1	b_2	c_3	6
a_2	b_2	c_3	6

E_1

x	y	$r_1 :=$ SUM(m_2)
a_1	b_1	$4 + 3$
a_1	b_2	6
a_2	b_2	6

KeyE_1

x	$m_1 :=$ MAX(r_1)
a_1	7
a_2	6

E_0

SUM(m_1)
$7 + 6$

Figure 7.2: Computation of LUB-CQA($g_0()$) for $g_0() := \texttt{SUM}(r) \leftarrow R(\underline{x}, y), S(\underline{y, z}, r)$.

problem $\mathsf{LUB\text{-}CQA}(g())$ is restricted to "yes"-instances of $\mathsf{CERTAINTY}(q)$. Therefore, "yes"-instances of $\mathsf{CERTAINTY}(q)$ can be treated as in the first case—i.e., the case where $\mathcal{F}_{\mathtt{AGG}}(\emptyset)$ is defined. $\square$

7.3.3 Elaborated Example and Optimization

Let $g() := \mathtt{AGG}(r) \leftarrow R_1(\underline{x}, y), R_2(\underline{y, z}, r)$, where $\mathcal{F}_{\mathtt{AGG}}$ is monotone and associative, and r is a variable. The following program computes $\mathsf{LUB\text{-}CQA}(g())$ as the unique value b for which $\mathit{E\_0}(b)$ holds ($b = 13$ in Fig. 7.2).

$$\begin{aligned}
\mathit{E\_2}(x, y, z, r, \mathtt{AGG}(r)) &\leftarrow R_1(\underline{x}, y), R_2(\underline{y, z}, r)\\
\mathit{KeyE\_2}(x, y, z, \mathtt{MAX}(r_2)) &\leftarrow \mathit{E\_2}(x, y, z, r, r_2)\\
\mathit{E\_1}(x, y, \mathtt{AGG}(m_2)) &\leftarrow \mathit{KeyE\_2}(x, y, z, m_2)\\
\mathit{KeyE\_1}(x, \mathtt{MAX}(r_1)) &\leftarrow \mathit{E\_1}(x, y, r_1)\\
\mathit{E\_0}(\mathtt{AGG}(m_1)) &\leftarrow \mathit{KeyE\_1}(x, m_1)
\end{aligned}$$

Fig. 7.2 presents a complete run of this program for the case where $\mathtt{AGG} = \mathtt{SUM}$. We briefly note that there are ways to optimize such programs, without delving into technical details. In particular, the optimization involves avoiding the full join expressed in rule (7.3) by distributing it across multiple rules, as illustrated below.

$$\begin{aligned}
\mathit{E\_2}'(y, z, r, \mathtt{AGG}(r)) &\leftarrow R_2(\underline{y, z}, r)\\
\mathit{KeyE\_2}'(y, z, \mathtt{MAX}(r_2)) &\leftarrow R_2(\underline{y, z}, r), \mathit{E\_2}'(y, z, r, r_2)\\
\mathit{E\_1}'(y, \mathtt{AGG}(m_2)) &\leftarrow \mathit{KeyE\_2}'(y, z, m_2)\\
\mathit{KeyE\_1}(x, \mathtt{MAX}(r_1)) &\leftarrow R_1(\underline{x}, y), \mathit{E\_1}'(y, r_1)\\
\mathit{E\_0}(\mathtt{AGG}(m_1)) &\leftarrow \mathit{KeyE\_1}(x, m_1)
\end{aligned}$$

For the database instance in Fig. 7.2 and $\mathtt{AGG} = \mathtt{SUM}$, the latter program computes the same relations for *KeyE_1* and *E_0* as shown in Fig. 7.2. However, the intermediate relations—being projections of the corresponding relations in Fig. 7.2—are smaller in size:

$\mathit{E\_2}'$

y	z	r	r_2
b_1	c_1	1	1
b_1	c_1	4	4
b_1	c_2	3	3
b_2	c_3	5	5
b_2	c_3	6	6

$\mathit{KeyE\_2}'$

y	z	m_2
b_1	c_1	4
b_1	c_2	3
b_2	c_3	6

$\mathit{E\_1}'$

y	r_1
b_1	7
b_2	6

Roughly, the envisioned optimization consists of projecting away from E_ℓ and $KeyE\_\ell$ those variables on which the computed numerical value does not depend. In the preceding rewritings, the value m_2 in $KeyE\_2(x, y, z, m_2)$ does not depend on x, which is therefore projected away in $KeyE\_2'(y, z, m_2)$.

7.4. Inexpressibility Result (Left-to-Right Implication in Theorem 7.0.1)

The right-to-left implication in Theorem 7.0.1 follows from Theorem 7.0.2. We now show the left-to-right implication, whose contraposition reads as follows: for each numerical query $g()$ in AGGR[sjfBCQ] with a head SUM(r) and $q =$ BCQ($g()$), if q is not κ-acyclic, then GLB-CQA($g()$) is not expressible in AGGR[FOL]. Note that, by the nontriviality condition in Definition 3.4.1, we have that if r is a constant, then $r \neq 0$. The proof relies on a reduction from the 2DM problem (a.k.a. *Bipartite Perfect Matching*), which we recall next.

2-DIMENSIONAL MATCHING (2DM)

INSTANCE: A set $M \subseteq A \times B$, where A and B are disjoint sets having the same number n of elements.

QUESTION: Does M contain a matching, that is, a subset $M' \subseteq M$ such that $|M'| = n$ and no two elements of M' agree in any coordinate?

Lemma 7.4.1. *Let $g()$ be a numerical query in* AGGR[sjfBCQ] *with a head* SUM(r) *and $q =$* BCQ($g()$). *If the attack graph of q has a cycle, then* 2DM $\leq_{\mathsf{FO}}$ LUB-CQA($g()$).

The proof of Lemma 7.4.1 is provided in Section D.4 in Appendix D. A key crux in the following proof of Theorem 7.0.1 is a deep result by Hella et al. Hella et al. (2001), which implies that every query in AGGR[FOL] is Hanf-local.

Proof of Theorem 7.0.1. The right-to-left implication of Theorem 7.0.1 follows from Theorem 7.0.2. We show the left-to-right implication by contraposition. Assume that q is not κ-acyclic. Let q' be a kernel of q, and let $g'()$ be the numerical query with the same head as $g()$ and BCQ($g'()$) $= q'$. Since q is not κ-acyclic, the attack graph of q' contains a cycle. By Lemma 7.4.1, 2DM $\leq_{\mathsf{FO}}$ LUB-CQA($g'()$). By Lemma 7.1.1, LUB-CQA($g'()$) $\leq_{\mathsf{FO}}$ LUB-CQA($g()$). Since first-order reductions compose, 2DM $\leq_{\mathsf{FO}}$ LUB-CQA($g()$). In (Libkin, 2004, Corollary 8.26 and Exercise 8.16), it is established that every query in a logic called $\mathcal{L}_{\mathrm{aggr}}$ is Hanf-local. It is easily verified that every query in $\mathsf{AGGR}[\mathsf{FOL}]$ is expressible in $\mathcal{L}_{\mathrm{aggr}}$, and therefore cannot express $\mathsf{2DM}$. It follows that $\mathsf{LUB\text{-}CQA}(g())$ cannot be expressed in $\mathsf{AGGR}[\mathsf{FOL}]$.

□

7.5. The κ-acyclicity of Cparsimony

Theorem 7.0.1 implies that every query in Cparsimony must be κ-acyclic. In this section we provide a proof of the following result that is based solely on syntax.

Proposition 7.5.1. *Every query in* Cparsimony *is κ-acylic.*

Proof of Proposition 7.5.1. Let q be a query in Cparsimony. Let q' be an arbitrary kernel of q. It suffices to show that the attack graph of q' is acyclic. Assume, for the sake of a contradiction, that the attack graph of q' contains a cycle. From (Koutris & Wijsen, 2017, Lemma 3.6), there are two distinct atoms $F, G \in q'$ such that $F \overset{q'}{\rightsquigarrow} G$ and $G \overset{q'}{\rightsquigarrow} F$. Let $\vec{x}$ be an id-set for q.

From condition (1) in Definition 4.1.3 and $\mathcal{K}(q) \equiv \mathcal{K}(q')$, it follows $\mathcal{K}(q') \models \vec{x} \rightarrow \mathsf{vars}(q)$. Consequently, $\mathcal{K}(q') \models \vec{x} \rightarrow \mathsf{Key}(F)$. If G occurs in a shortest sequential proof of $\mathcal{K}(q') \models \vec{x} \rightarrow \mathsf{Key}(F)$, then $\mathcal{K}(q' \setminus \{F\}) \models \vec{x} \rightarrow \mathsf{Key}(G)$. Otherwise $\mathcal{K}(q' \setminus \{G\}) \models \vec{x} \rightarrow \mathsf{Key}(F)$. Consequently, either $\mathcal{K}(q' \setminus \{G\}) \models \vec{x} \rightarrow \mathsf{Key}(F)$ or $\mathcal{K}(q' \setminus \{F\}) \models \vec{x} \rightarrow \mathsf{Key}(G)$ (or both). Assume $\mathcal{K}(q' \setminus \{G\}) \models \vec{x} \rightarrow \mathsf{Key}(F)$ (the case $\mathcal{K}(q' \setminus \{F\}) \models \vec{x} \rightarrow \mathsf{Key}(G)$ is symmetrical). By Lemma C.1.1, since $\mathcal{K}(q' \setminus \{G\}) \models \vec{x} \rightarrow \mathsf{Key}(F)$ and $G \overset{q'}{\rightsquigarrow} F$, it follows that $G \overset{q'}{\rightsquigarrow} v$ for some variable v in $\vec{x}$. Thus, there is a path $(w_1, \dots, w_n)$ in $\mathcal{G}aifman(q')$ such that $w_1 \in \mathsf{notKey}(G)$, $w_n = v$, and for every $i \in \{1, \dots, n\}$, $\mathcal{K}(q' \setminus \{G\}) \not\models \mathsf{Key}(G) \rightarrow w_i$. Since $\mathcal{G}aifman(q) = \mathcal{G}aifman(q')$, the path $(w_1, \dots, w_n)$ also exists in $\mathcal{G}aifman(q)$. We now show the following claims.

Claim 2. For every $i \in \{1, \dots, n\}$, $\mathcal{K}(q') \not\models \emptyset \rightarrow w_i$.

Proof. Assume, for the sake of contradiction, that $\mathcal{K}(q') \models \emptyset \rightarrow w_i$ for some $i \in \{1, \dots, n\}$. Since $G \overset{q'}{\rightsquigarrow} w_i$, it follows that $\mathcal{K}(q' \setminus \{G\}) \not\models \emptyset \rightarrow w_i$. This implies that $\mathcal{K}(q' \setminus \{G\}) \models \emptyset \rightarrow \mathsf{Key}(G)$. Since $F \overset{q'}{\rightsquigarrow} G$, F attacks some variable in $\mathsf{Key}(G)$. Thus, we have that $\mathcal{K}(q' \setminus \{G, F\}) \not\models \emptyset \rightarrow \mathsf{Key}(G)$. This implies that $\mathcal{K}(q' \setminus \{G, F\}) \models \emptyset \rightarrow \mathsf{Key}(F)$, hence $G \overset{q'}{\not\rightsquigarrow} F$, a contradiction. This concludes the proof of Claim 2. □

Since $\mathcal{K}(q) \equiv \mathcal{K}(q')$, it follows that for every $i \in \{1, \dots, n\}$, $\mathcal{K}(q) \not\models \emptyset \rightarrow w_i$, and therefore no variable in $(w_1, \dots, w_n)$ is frozen in q.

Claim 3. There is an atom H in q satisfying:

(a) $\mathcal{K}(q) \models \emptyset \rightarrow \mathsf{Key}(H) \setminus \mathsf{Key}(G)$; and

(b) there is a path $(z_1, \dots, z_m)$ in $\mathcal{G}aifman(q)$ such that $z_1 \in \mathsf{notKey}(H)$, $z_m = w_1$, and for every $i \in \{1, \dots, m\}$, $z_i \notin \mathsf{Key}(H) \cup \mathsf{frozen}(q)$.

Proof. Since $w_1 \in \mathsf{notKey}(G)$, it follows that $\mathcal{K}(q') \models \mathsf{Key}(G) \rightarrow w_1$. Since $\mathcal{K}(q') \equiv \mathcal{K}(q)$, it follows that $\mathcal{K}(q) \models \mathsf{Key}(G) \rightarrow w_1$. We can assume a shortest sequence

$$(H_1, \dots, H_\ell) \tag{7.8}$$

that is a sequential proof of $\mathcal{K}(q) \models \mathsf{Key}(G) \rightarrow w_1$. Note that, since $\mathcal{K}(q' \setminus \{G\}) \not\models \mathsf{Key}(G) \rightarrow w_1$, we have that $w_1 \notin \mathsf{Key}(G)$, and thus, $\ell \geq 1$. By definition of a sequential proof, it follows that $w_1 \in \mathsf{notKey}(H_\ell)$. We distinguish two possibilities.

Case that $\mathcal{K}(q) \models \emptyset \rightarrow \mathsf{Key}(H_\ell) \setminus \mathsf{Key}(G)$. Then Claim 3 is proved by choosing $H = H_\ell$, and using the empty path (w_1) in $\mathcal{G}aifman(q)$.

Case that there is $z_1 \in \mathsf{Key}(H_\ell) \setminus \mathsf{Key}(G)$ such that $\mathcal{K}(q) \not\models \emptyset \rightarrow z_1$. Since the sequence in (7.8) is a sequential proof, there is $k \in \{1, \dots, \ell - 1\}$ such that $z_1 \in \mathsf{notKey}(H_k)$. Let k be the smallest index such that $z_1 \in \mathsf{notKey}(H_k)$. This choice implies that for every $j \in \{1, \dots, k\}$, $z_1 \notin \mathsf{Key}(H_j)$. If $\mathcal{K}(q) \models \emptyset \rightarrow \mathsf{Key}(H_k) \setminus \mathsf{Key}(G)$, then Claim 3 is proved by choosing $H = H_k$ and using the path (z_1, w_1) in $\mathcal{G}aifman(q)$. Otherwise, there is a variable $z_2 \in \mathsf{Key}(H_k) \setminus \mathsf{Key}(G)$ such that $\mathcal{K}(q) \not\models \emptyset \rightarrow z_2$, and we can repeat the same reasoning as before, replacing z_1 with z_2, and H_ℓ with H_k. By repeating the same reasoning, we eventually reach an atom H in (7.8) such that $\mathcal{K}(q) \models \emptyset \rightarrow \mathsf{Key}(H) \setminus \mathsf{Key}(G)$.

The proof of Claim 3 is now concluded. □

Let H and $(z_1, \dots, z_m)$ be as in the statement of Claim 3. For every $i \in \{1, 2, \dots, n\}$, we have $w_i \notin \mathsf{Key}(H)$, or else, since $w_i \notin \mathsf{Key}(G)$, $\mathcal{K}(q) \models \emptyset \rightarrow w_i$, a contradiction. Since $z_m = w_1$, $\mathcal{G}aifman(q)$ has a path

$$(z_1, z_2, \dots, z_{m-1}, w_1, w_2, \dots, w_n)$$

where $w_n = v$, $z_1 \in \mathsf{notKey}(H)$, $v \in \mathsf{vars}(\vec{x})$, and no variable in the path belongs to $\mathsf{Key}(H) \cup \mathsf{frozen}(q)$. The existence of this path shows that $q \notin \mathsf{Cparsimony}$, a contradiction. □

$$\begin{aligned} \mathit{E\_2}(x, y, r, r_2) &\leftarrow R(\underline{x}, y), T(\underline{y}, r), r = r_2. \\ \mathit{KeyE\_1}(x, \mathtt{MAX}(r_2)) &\leftarrow \mathit{E\_2}(x, y, r, r_2). \\ \mathit{E\_0}(\mathtt{SUM}(m_1)) &\leftarrow \mathit{KeyE\_1}(x, m_1). \end{aligned}$$

Figure 7.3: Calculation of $\mathsf{LUB\text{-}CQA}(g_1())$ for $g_1() := \mathtt{SUM}(r) \leftarrow R(\underline{x}, y), T(\underline{y}, r)$. $\mathsf{LUB\text{-}CQA}(g_1())$ is the unique value b for which $\mathit{E\_0}(b)$ holds true.

Example 7.5.1. Consider the $\mathtt{SUM}$-query $g_1() := \mathtt{SUM}(r) \leftarrow R(\underline{x}, y), T(\underline{y}, r)$. It is easily verifiable that $\mathsf{BCQ}(g_1()) \in \mathsf{Cparsimony}$ and, by Proposition 7.5.1, $\mathsf{BCQ}(g_1())$ is κ-acyclic. The difference from the query $g_0()$, whose rewriting is shown in Fig. 7.2, is that the variable y shared by the two atoms constitutes the primary key of one of the atoms. The computation of $\mathsf{GLB\text{-}CQA}(g_1())$ is given in Fig. 7.3. Compared to Fig. 7.2, the IDB predicates *KeyE_2* and *E_1* can be skipped. $\lhd$

CHAPTER 8

Exploring Other Aggregation Queries

In this chapter, we explore the expressibility of the problems $\mathsf{GLB\text{-}CQA}(g(\vec{x}))$ and $\mathsf{LUB\text{-}CQA}(g(\vec{x}))$ in $\mathsf{AGGR[FOL]}$ for numerical terms $g(\vec{x})$ that were not covered in previous chapters. We introduce in Section 8.1 a manifestation of non-monotonicity, called *descending chains*, which is used to identify cases where $\mathsf{GLB\text{-}CQA}(g())$ is not expressible in $\mathsf{AGGR[FOL]}$ under the above assumptions. Descending chains are also used in Section 8.2 in the study of $\mathsf{LUB\text{-}CQA}(g())$ and $\mathsf{GLB\text{-}CQA}(g())$ via the use of dual aggregate operators, and in Section 8.3 to demonstrate that unconstrained numeric columns (i.e., columns not constrained to $\mathbb{Q}_{\geq 0}$) suffice for moving from expressibility in $\mathsf{AGGR[FOL]}$ to non-expressibility. We conclude with some positive results: In Section 8.4, we show that when $g()$ is a $\texttt{MIN}$-query, it is decidable whether or not $\mathsf{GLB\text{-}CQA}(g())$ and $\mathsf{LUB\text{-}CQA}(g())$ are expressible in $\mathsf{AGGR[FOL]}$ and, in Section 8.5, we explain how the results provided in previous chapters can be extended to numerical terms with free variables.

8.1. Aggregate Operators with Descending Chains

We first show that the inverse of Theorem 3.5.6 does not hold for every numerical query: Lemmas 8.1.1 and 8.1.2 introduce numerical queries with acyclic attack graphs that however do not allow glb rewriting in $\mathsf{AGGR[FOL]}$. Their proofs rely on the existence of a (possibly bounded) descending chain for an aggregate operator $\mathcal{F}_{\texttt{AGG}}$, which implies that $\mathcal{F}_{\texttt{AGG}}$ lacks monotonicity. Concrete examples of such aggregate operators are $\mathcal{F}_{\texttt{AVG}}$ and $\mathcal{F}_{\texttt{PRODUCT}}$, as expressed by Corollary 8.1.3.1 (assuming that the numeric domain is $\mathbb{Q}_{\geq 0}$).

Definition 8.1.1 (Descending chain)**.** For $i \in \mathbb{N}$ and $t \in \mathbb{Q}$, we write $i\#t$ as a shorthand for $\overbrace{t, t, \ldots, t}^{i \text{ times}}$, i.e., i occurrences of t. We say that an aggregate operator $\mathcal{F}_{\mathtt{AGG}}$ has a *descending chain* if there exist $s, t \in \mathbb{Q}_{\geq 0}$ such that for every $i \in \mathbb{N}$, $\mathcal{F}_{\mathtt{AGG}}(\{\{s, i\#t\}\}) > \mathcal{F}_{\mathtt{AGG}}(\{\{s, (i+1)\#t\}\})$. Note that s and t need not be distinct. Such a descending chain is said to be *bounded* if for every $i \in \mathbb{N}$, there exists $m_i \in \mathbb{Q}_{\geq 0}$ such that for all $j \in \mathbb{N}_{>0}$, for all $k', k \in \mathbb{N}$ such that $k' \leq k \leq i$, we have $\mathcal{F}_{\mathtt{AGG}}(\{\{s, k'\#t\}\}) < \mathcal{F}_{\mathtt{AGG}}(\{\{j\#m_i, s, k\#t\}\})$. Informally, this expression means that $\mathcal{F}_{\mathtt{AGG}}$ will strictly increase if at least one copy of m_i is added, regardless of any addition of copies of t. Notice that m_i depends on i. See the proof of Lemma 8.1.3 for examples. □

Lemma 8.1.1. *Let $\mathcal{F}_{\mathtt{AGG}}$ be an aggregate operator with a descending chain (which may not be bounded). Then,* $\mathsf{GLB\text{-}CQA}(g())$ *is* NL*-hard for* $g() := \mathtt{AGG}(r) \leftarrow R(\underline{x}, y, r), S_1(\underline{y}, x), S_2(\underline{y}, x)$*. Consequently,* $\mathsf{GLB\text{-}CQA}(g())$ *is not expressible in* $\mathsf{AGGR[FOL]}$*.*

Proof. First-order reduction from 2DM (2-DIMENSIONAL MATCHING) which is known to be NL-hard Chandra et al. (1984).

2-DIMENSIONAL MATCHING (2DM)

INSTANCE: A set $M \subseteq A \times B$, where A and B are disjoint sets having the same number n of elements.

QUESTION: Does M contain a matching, that is, a subset $M' \subseteq M$ such that $|M'| = n$ and no two elements of M' agree in any coordinate?

Since $\mathcal{F}_{\mathtt{AGG}}$ has a descending chain, we can assume $s, t \in \mathbb{Q}_{\geq 0}$ such that $\mathcal{F}_{\mathtt{AGG}}(\{\{s\}\}) > \mathcal{F}_{\mathtt{AGG}}(\{\{s, t\}\}) > \mathcal{F}_{\mathtt{AGG}}(\{\{s, t, t\}\}) > \mathcal{F}_{\mathtt{AGG}}(\{\{s, t, t, t\}\}) > \cdots$. Given an instance M of 2DM, construct a database instance $\mathbf{db}_M$ as follows:

- for every $(a, b) \in M$, add $R(\underline{a}, b, t)$, $S_1(\underline{b}, a)$, and $S_2(\underline{b}, a)$; and
- add $R(\underline{\bot_A}, \bot_B, s)$, $S_1(\underline{\bot_B}, \bot_A)$, and $S_2(\underline{\bot_B}, \bot_A)$, where $\bot_A$ and $\bot_B$ are fresh constants.

If M has a matching, then $\mathbf{db}_M$ has a repair on which $g()$ returns $\ell := \mathcal{F}_{\mathtt{AGG}}(\{\{s, n\#t\}\})$. If M has no matching, then every repair of $\mathbf{db}_M$ has less than n embeddings of $g()$'s body, and hence on any repair, $g()$ returns the value $\mathcal{F}_{\mathtt{AGG}}(\{\{s, m\#t\}\})$ with $m < n$, hence $\mathcal{F}_{\mathtt{AGG}}(\{\{s, m\#t\}\}) > \ell$. Consequently, $\mathsf{GLB\text{-}CQA}(q)$ returns ℓ if and only if M has a matching. □

Lemma 8.1.2. *Let $\mathcal{F}_{\mathtt{AGG}}$ be an aggregate operator with a bounded descending chain. Then,* $\mathsf{GLB\text{-}CQA}(g())$ *is* NP*-hard for* $g() := \mathtt{AGG}(r) \leftarrow S_1(\underline{x}, c_1)$, $S_2(\underline{y}, c_2)$, $T(\underline{x, y}, r)$, *where c_1 and c_2 are (not necessarily distinct) constants. Consequently,* $\mathsf{GLB\text{-}CQA}(g())$ *is not expressible in* $\mathsf{AGGR[FOL]}$.

Proof of Lemma 8.1.2. Proof adapted from a similar proof in (Dixit & Kolaitis, 2022). First order-reduction from SIMPLE MAX CUT, which is known to be NP-hard (Garey et al., 1976).

SIMPLE MAX CUT

INSTANCE: Graph $G = (V, E)$, positive integer K.

QUESTION: Is there a partition of V into disjoint sets V_1 and V_2 such that the number of edges from E that have one endpoint in V_1 and one endpoint in V_2 is at least K?

Since $\mathcal{F}_{\mathtt{AGG}}$ has a bounded descending chain, we can assume $s, t \in \mathbb{Q}_{\geq 0}$ such that $\mathcal{F}_{\mathtt{AGG}}(\{\{s\}\}) > \mathcal{F}_{\mathtt{AGG}}(\{\{s, t\}\}) > \mathcal{F}_{\mathtt{AGG}}(\{\{s, t, t\}\}) > \mathcal{F}_{\mathtt{AGG}}(\{\{s, t, t, t\}\}) > \cdots$. Let $e = 2{*}|E|$. Moreover, there is $m_e \in \mathbb{Q}_{\geq 0}$ such that for all $j \in \mathbb{N}_{>0}$, for all $k', k \in \mathbb{N}$ such that $k' \leq k \leq e$, we have $\mathcal{F}_{\mathtt{AGG}}(\{\{s, k'\#t\}\}) < \mathcal{F}_{\mathtt{AGG}}(\{\{j\#m_e, s, k\#t\}\})$.

Given an instance $G = (V, E)$ of SIMPLE MAX CUT, construct a database $\mathbf{db}_G$ as follows. We can assume $E \neq \emptyset$, and that the graph is simple. Let d be a constant such that $c_1 \neq d \neq c_2$.

- for every $v \in V$, $\mathbf{db}_G$ contains $S_1(\underline{v}, c_1)$ and $S_1(\underline{v}, d)$;
- for every $v \in V$, $\mathbf{db}_G$ contains $S_2(\underline{v}, c_2)$ and $S_2(\underline{v}, d)$;
- for every edge $\{u, v\}$ in E, $\mathbf{db}_G$ contains both $T(\underline{u, v}, t)$ and $T(\underline{v, u}, t)$;
- for every $v \in V$, $\mathbf{db}_G$ contains $T(\underline{v, v}, m_e)$;
- $\mathbf{db}_G$ contains $S_1(\underline{\bot}, c_1)$, $S_2(\underline{\bot}, c_2)$, $T(\underline{\bot, \bot}, s)$, where $\bot$ is a fresh constant. It follows that every repair of $\mathbf{db}_G$ satisfies

$$\exists x \exists y \exists r \left(S_1(\underline{x}, c_1) \land S_2(\underline{y}, c_2) \land T(\underline{x, y}, r) \right).$$

Note that the T-relation of $\mathbf{db}_G$ is consistent, and hence belongs to every repair. We show that $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}_G} \leq \mathcal{F}_{\mathtt{AGG}}(\{\{s, K\#t\}\})$ if and only if G is a "yes"-instance of SIMPLE MAX CUT.

$\boxed{\Longleftarrow}$ Assume that G is a "yes"-instance of SIMPLE MAX CUT, as witnessed by a partition of V into V_1 an V_2. Construct a repair $\mathbf{r}$ as follows:

- for every $v \in V_1$, $\mathbf{r}$ contains $S_1(\underline{v}, c_1)$ and $R_2(\underline{v}, d)$;
- for every $v \in V_2$, $\mathbf{r}$ contains $R_2(\underline{v}, c_2)$ and $S_1(\underline{v}, d)$.

It is easily verified that $[\![g()]\!]^{\mathbf{r}} \leq \mathcal{F}_{\texttt{AGG}}(\{\{s, K\#t\}\})$.

$\boxed{\Longrightarrow}$ Assume that $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}_G} \leq \mathcal{F}_{\texttt{AGG}}(\{\{s, K\#t\}\})$. We can assume a repair $\mathbf{r}$ such that $[\![g()]\!]^{\mathbf{r}} = [\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}_G}$. Construct V_1 and V_2 as follows. Whenever $\mathbf{r}$ contains $S_1(\underline{v}, c_1)$, then $v \in V_1$. Whenever $\mathbf{r}$ contains $S_2(\underline{v}, c_2)$, then $v \in V_2$. Whenever $\mathbf{r}$ contains both $S_1(\underline{v}, d)$ and $S_2(\underline{v}, d)$, then $v \in V_1$ (the choice is arbitrary). Let $j := |V_1 \cap V_2|$. Let k be the number of valuations θ over $\{x, y\}$ with $\theta(x) \neq \theta(y)$ such that $(\mathbf{s}, \theta) \models S_1(\underline{x}, c_1) \wedge S_2(\underline{y}, c_2) \wedge T(\underline{x, y}, r)$. Then, $[\![g()]\!]^{\mathbf{r}} = \mathcal{F}_{\texttt{AGG}}(\{\{j\#m_e, s, k\#t\}\})$. Clearly, $k \leq e$. We show $j = 0$. Assume for the sake of a contradiction $j \geq 1$. Let $\mathbf{r}'$ be the repair obtained from $\mathbf{r}$ by replacing $S_2(\underline{v}, c_2)$ with $S_2(\underline{v}, d)$ for every $v \in V_1 \cap V_2$. Then, $[\![g()]\!]^{\mathbf{r}'} = \mathcal{F}_{\texttt{AGG}}(\{\{s, k'\#t\}\})$ for some k' with $k' \leq k$. Then, $\mathcal{F}_{\texttt{AGG}}(\{\{s, k'\#t\}\}) < \mathcal{F}_{\texttt{AGG}}(\{\{j\#m_e, s, k\#t\}\})$, contradicting that $g()$ reaches a minimum in $\mathbf{r}$. We conclude by contradiction that $j = 0$, hence $[\![g()]\!]^{\mathbf{r}} = \mathcal{F}_{\texttt{AGG}}(\{\{s, k\#t\}\})$. Since $\mathcal{F}_{\texttt{AGG}}(\{\{s, k\#t\}\}) \leq \mathcal{F}_{\texttt{AGG}}(\{\{s, K\#t\}\})$, we have $k \geq K$. Consequently, the number of edges from E that have one endpoint in V_1 and one end-point in V_2 is at least K. $\square$

Lemma 8.1.3. *$\mathcal{F}_{\texttt{AVG}}$ and $\mathcal{F}_{\texttt{PRODUCT}}$ have bounded descending chains.*

Proof. For $\mathcal{F}_{\texttt{AVG}}$, we have $\mathcal{F}_{\texttt{AVG}}(\{\{1\}\}) > \mathcal{F}_{\texttt{AVG}}(\{\{1, 0\}\}) > \mathcal{F}_{\texttt{AVG}}(\{\{1, 0, 0\}\} > \mathcal{F}_{\texttt{AVG}}(\{\{1, 0, 0, 0\}\} > \cdots$. Take $s = 1$ and $t = 0$. For every $i \in \mathbb{N}$, let $m_i = i + 2$. Whenever $j > 0$ and $0 \leq k' \leq k \leq i$, we have $\frac{1}{k'+1} = \mathcal{F}_{\texttt{AVG}}(\{\{1, k'\#0\}\} < \mathcal{F}_{\texttt{AVG}}(\{\{j\#(i+2), 1, k\#0\}\} = \frac{j*(i+2)+1}{j+k+1}$, as desired.

For $\mathcal{F}_{\texttt{PRODUCT}}$, we have

$$\mathcal{F}_{\texttt{PRODUCT}}(\{\{\frac{1}{2}\}\}) > \mathcal{F}_{\texttt{PRODUCT}}(\{\{\frac{1}{2}, \frac{1}{2}\}\}) > \mathcal{F}_{\texttt{PRODUCT}}(\{\{\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\}\}) > \cdots.$$

Take $s = t = \frac{1}{2}$. Let $i \in \mathbb{N}$. For $j > 0$ and $0 \leq k' \leq k \leq i$, we obtain $\mathcal{F}_{\texttt{PRODUCT}}(\{\{\frac{1}{2}, k'\#\frac{1}{2}\}\} < \mathcal{F}_{\texttt{PRODUCT}}(\{\{j\#m_i, \frac{1}{2}, k\#\frac{1}{2}\}\}$ by choosing $m_i = 2^{i+1}$. $\square$

Corollary 8.1.3.1. $\mathsf{GLB\text{-}CQA}(g())$ *is not expressible in* $\mathsf{AGGR[FOL]}$ *with* $g()$ *as in Lemma 8.1.1 or Lemma 8.1.2 and* $\texttt{AGG} \in \{\texttt{AVG}, \texttt{PRODUCT}\}$.

8.2. Dual Aggregate Operators

Definition 8.2.1 (Dual aggregate operator)**.** The *dual* of positive aggregate operator $\mathcal{F}_{\texttt{AGG}}$, denoted $\mathcal{F}_{\texttt{AGG}}^{\mathsf{dual}}$, is the mapping such that for every finite multiset X of non-negative rational numbers,

- if $\mathcal{F}_{\texttt{AGG}}(X)$ is undefined, then $\mathcal{F}^{\mathsf{dual}}_{\texttt{AGG}}(X)$ is undefined; and
- if $\mathcal{F}_{\texttt{AGG}}(X)$ is defined, then $\mathcal{F}^{\mathsf{dual}}_{\texttt{AGG}}(X) = -1 * \mathcal{F}_{\texttt{AGG}}(X)$.

The dual of a positive aggregate operator is also called a *dual aggregate operator.* □

Note that a dual aggregate operator is not a positive aggregate operator itself, as it can return negative rational numbers. Despite this, Definition 8.1.1 of (bounded) descending chain applies also to dual aggregate operators. Additionally, it can be verified that Lemmas 8.1.1 and 8.1.2 remain valid for dual aggregate operators, since their proofs do not rely on the signs of the aggregated values.

As formalized by Proposition 8.2.1, the function problem $\mathsf{LUB\text{-}CQA}(g())$ for $g() := \texttt{AGG}(r) \leftarrow q(\vec{u})$ is the same up to a sign as $\mathsf{GLB\text{-}CQA}(h())$ where $h()$ has the same body as $g()$, but uses the dual of $\texttt{AGG}$ in its head:

$$h() := \texttt{AGG}^{\mathsf{dual}}(r) \leftarrow q(\vec{u}),$$

where the aggregate symbol $\texttt{AGG}^{\mathsf{dual}}$ is interpreted by $\mathcal{F}^{\mathsf{dual}}_{\texttt{AGG}}$, i.e., the dual of $\mathcal{F}_{\texttt{AGG}}$. The semantics of $h()$ on a database instance $\mathbf{db}$ is naturally defined: if $g()$ returns a rational number r, then $h()$ returns $-1 * r$. The problems $\mathsf{GLB\text{-}CQA}(h())$ and $\mathsf{LUB\text{-}CQA}(h())$ are defined as before. The proof of the following propostion is straightforward.

Proposition 8.2.1. *Let* $g() := \texttt{AGG}(r) \leftarrow q(\vec{u})$ *be a numerical query in the class* $\mathsf{AGGR[sjfBCQ]}$. *Let* $h() := \texttt{AGG}^{\mathsf{dual}}(r) \leftarrow q(\vec{u})$. *Then, for every database instance* $\mathbf{db}$*, we have*

- $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}} = [\![\mathsf{GLB\text{-}CQA}(h())]\!]^{\mathbf{db}} = \perp$ *if* $\mathbf{db} \not\models_{\mathsf{cqa}} \exists\vec{u}(q(\vec{u}))$ *and* $\mathcal{F}_{\texttt{AGG}}(\emptyset)$ *is undefined;*
- $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}} = -1 * [\![\mathsf{GLB\text{-}CQA}(h())]\!]^{\mathbf{db}}$ *otherwise.*

If we let $\texttt{AGG} = \texttt{SUM}$ in the numerical term $g()$ of Lemma 8.1.1, then $\mathsf{GLB\text{-}CQA}(g())$ is in $\mathsf{AGGR[FOL]}$ (by Theorem 6.0.2), but $\mathsf{LUB\text{-}CQA}(g())$ is not, as a consequence of the following lemma.

Theorem 8.2.2. $\mathsf{LUB\text{-}CQA}(g())$ *is not expressible in* $\mathsf{AGGR[FOL]}$ *with* $g()$ *as in Lemma 8.1.1 and* $\texttt{AGG} \in \{\texttt{SUM}, \texttt{AVG}, \texttt{PRODUCT}\}$.

Proof. From $\mathcal{F}^{\mathsf{dual}}_{\texttt{AVG}}(\{\{0\}\}) > \mathcal{F}^{\mathsf{dual}}_{\texttt{AVG}}(\{\{0,1\}\}) > \mathcal{F}^{\mathsf{dual}}_{\texttt{AVG}}(\{\{0,1,1\}\} > \cdots$, it follows that $\mathcal{F}^{\mathsf{dual}}_{\texttt{AVG}}$ has a descending chain. From $\mathcal{F}^{\mathsf{dual}}_{\texttt{SUM}}(\{\{1\}\}) > \mathcal{F}^{\mathsf{dual}}_{\texttt{SUM}}(\{\{1,1\}\}) >$

$\mathcal{F}^{\mathsf{dual}}_{\mathtt{SUM}}(\{\{1,1,1\}\} > \cdots$, it follows that $\mathcal{F}^{\mathsf{dual}}_{\mathtt{SUM}}$ has a descending chain. From $\mathcal{F}^{\mathsf{dual}}_{\mathtt{PRODUCT}}(\{\{2\}\}) > \mathcal{F}^{\mathsf{dual}}_{\mathtt{PRODUCT}}(\{\{2,2\}\}) > \mathcal{F}^{\mathsf{dual}}_{\mathtt{PRODUCT}}(\{\{2,2,2\}\} > \cdots$, it follows that $\mathcal{F}^{\mathsf{dual}}_{\mathtt{PRODUCT}}$ has a descending chain. The desired result then follows by Proposition 8.2.1 and Lemma 8.1.1. □

It can be easily verified that the descending chain for PRODUCT in the proof of Theorem 8.2.2 is bounded by choosing $m_i = \frac{1}{2^{i+1}}$, which implies that LUB-CQA($g()$) is not expressible in AGGR[FOL] with $g()$ as in Lemma 8.1.2 and AGG = PRODUCT.

8.3. Unconstrained Numerical Columns

So far, we have restricted our attention to database instances in which all numbers occurring in numeric columns are non-negative. For the following theorem, it is relevant to note that while $\mathcal{F}_{\mathtt{SUM}}$ is monotone over both $\mathbb{N}$ and $\mathbb{Q}_{\geq 0}$ (and covered by previous chapters), it becomes non-monotone if these domains are extended by even a single negative number (in our treatment, the integer -1). Since for the numerical query $g()$ of Theorem 8.3.1 we have that BCQ($g()$) is in Cforest, it disproves a claim in (Fuxman, 2007) stating that GLB-CQA($g()$) is expressible in AGGR[FOL] for all SUM-queries $h()$ such that BCQ($h()$) is in Cforest.

Theorem 8.3.1. *Assume that the third attribute of T is a numeric column that can contain numbers in $\mathbb{N} \cup \{-1\}$. Then,* GLB-CQA($g()$) *is* NP*-hard for $g() :=$ $\mathtt{SUM}(r) \leftarrow S_1(\underline{x}, c_1), S_2(\underline{y}, c_2), T(\underline{x, y}, r)$, where c_1 and c_2 are (not necessarily distinct) constants. Consequently,* GLB-CQA($g()$) *is not in* AGGR[FOL].

Proof sketch. It can be easily verified that $\mathcal{F}_{\mathtt{SUM}}$ has a bounded descending chain if -1 can be used. The desired result then follows from Lemma 8.1.2. □

8.4. MIN and MAX

$\mathcal{F}_{\mathtt{MIN}}$ is not monotone, since its value can decrease when extending a multiset, for example, $\mathcal{F}_{\mathtt{MIN}}(\{\{3\}\}) > \mathcal{F}_{\mathtt{MIN}}(\{\{2,3\}\})$. Despite this, rewritability in AGGR[FOL] is decidable for MIN-queries, for both glb and lub.

Theorem 8.4.1. *Let $g() := \mathtt{MIN}(r) \leftarrow q(\vec{y})$ be a query in* AGGR[sjfBCQ]. *Then,*

- GLB-CQA($g()$) *is expressible in* AGGR[FOL] *if and only if the attack graph of $\exists\vec{y}(q(\vec{y}))$ is acyclic.*

- $\mathsf{LUB\text{-}CQA}(g())$ *is expressible in* $\mathsf{AGGR[FOL]}$ *if and only if the attack graph of* $\exists\vec{y}(q(\vec{y}))$ *is acyclic.*

Proof. If the attack graph of $\exists\vec{y}(q(\vec{y}))$ is cyclic, then by Theorems 3.5.2 and 3.5.3, both $\mathsf{GLB\text{-}CQA}(g())$ and $\mathsf{LUB\text{-}CQA}(g())$ are not expressible in $\mathsf{AGGR[FOL]}$.

Assume from here on that the attack graph of $\exists\vec{y}(q(\vec{y}))$ is acyclic. Let $\mathbf{db}$ be a database instance. From Theorem 2.0.1, there is a first-order formula ψ such that $\mathbf{db} \models \psi$ if and only if $\mathbf{db} \models_{\mathsf{cqa}} \exists\vec{y}(q(\vec{y}))$. If $\mathbf{db} \not\models \psi$, then $\mathsf{GLB\text{-}CQA}(g()) = \mathsf{LUB\text{-}CQA}(g()) = \bot$. Assume $\mathbf{db} \models \psi$ from here on. It can be easily seen that $[\![\mathsf{GLB\text{-}CQA}(g())]\!]^{\mathbf{db}} = [\![g()]\!]^{\mathbf{db}}$, hence $\mathsf{GLB\text{-}CQA}(g())$ is expressible in $\mathsf{AGGR[FOL]}$. It remains to show that $\mathsf{LUB\text{-}CQA}(g())$ is also expressible in $\mathsf{AGGR[FOL]}$. To this end, define $(\mathbb{Q}, \leq')$ such that $r \leq' s$ if and only if $s \leq r$, that is, $\leq'$ reverses the natural order on the rational numbers. Then, $\mathsf{LUB\text{-}CQA}(g())$ relative to $\leq$ coincides with $\mathsf{GLB\text{-}CQA}(\mathtt{MAX}(r) \leftarrow q(\vec{y}))$ relative to $\leq'$. From Theorem 6.0.2, there is a formula $\varphi()$ in $\mathsf{AGGR[FOL]}$ that expresses $\mathsf{GLB\text{-}CQA}(\mathtt{MAX}(r) \leftarrow q(\vec{y}))$ relative to $\leq'$. $\square$

On the other hand, for `MAX`-queries $g()$, $\mathsf{LUB\text{-}CQA}(g())$ is expressible in $\mathsf{AGGR[FOL]}$ even when the attack graph of the body of $g()$ is cyclic. This different behavior between `MAX`-queries and `MIN`-queries arises from the fact that, with $\mathbb{Q}_{\geq 0}$ as numerical domain, $\mathcal{F}_{\mathtt{MAX}}(\emptyset)$ is defined, whereas $\mathcal{F}_{\mathtt{MIN}}(\emptyset)$ is not. Thus, Theorem 3.5.3 is not applicable to `MAX`-queries.

Theorem 8.4.2. *Let* $g() := \mathtt{MAX}(r) \leftarrow q(\vec{y})$ *be a query in* $\mathsf{AGGR[sjfBCQ]}$. *Then,*

- $\mathsf{GLB\text{-}CQA}(g())$ *is expressible in* $\mathsf{AGGR[FOL]}$ *if and only if the attack graph of* $\exists\vec{y}(q(\vec{y}))$ *is acyclic.*
- $\mathsf{LUB\text{-}CQA}(g())$ *is expressible in* $\mathsf{AGGR[FOL]}$.

Proof. The first item follows from Theorems 3.5.6 and 6.0.2. It remains to show that $\mathsf{LUB\text{-}CQA}(g())$ is also expressible in $\mathsf{AGGR[FOL]}$. Let $\mathbf{db}$ be a database instance. It can be easily seen that $[\![\mathsf{LUB\text{-}CQA}(g())]\!]^{\mathbf{db}} = [\![g()]\!]^{\mathbf{db}}$, hence $\mathsf{LUB\text{-}CQA}(g())$ is expressible in $\mathsf{AGGR[FOL]}$. $\square$

We point out that Theorem 8.4.2 does not contradict (Amezian El Khalfioui & Wijsen, 2024a, Theorem 7.11), although it may seem so at first sight. In the latter paper, for every `AGG`-query $g()$, $\mathsf{LUB\text{-}CQA}(g())$ is defined to return $\bot$ whenever some repair falsifies $\mathsf{BCQ}(g())$, thereby avoiding the need to handle $\mathcal{F}_{\mathtt{AGG}}(\emptyset)$, the result of applying the aggregate operator to the empty multiset. In this thesis, we adopt a different, arguably more practical, approach: if $\mathcal{F}_{\mathtt{AGG}}(\emptyset)$ is defined, then that value can be a valid output of $\mathsf{LUB\text{-}CQA}(g())$. In particular, we define $\mathcal{F}_{\texttt{MAX}}(\emptyset) = 0$, and thus a $\texttt{MAX}$-query $g()$ returns 0 on the empty repair—which differs from $\mathsf{LUB\text{-}CQA}(g())$ unless $g()$ returns 0 on every repair.

8.5. Aggregation Queries with Free Variables

So far, we have focused on numerical terms $g()$ without free variables. We now explain how our results extend to numerical terms $g(\vec{x}) = \mathsf{Aggr}_{\mathcal{F}}\vec{y}\,[r, q(\vec{x},\vec{y})]$ with free variables $\vec{x} := (x_1, \dots, x_k)$, where $q(\vec{x},\vec{y})$ is self-join-free and $\mathcal{F}_{\texttt{AGG}}$ is both monotone and associative. Let $\vec{c} = (c_1, \dots, c_k)$ be a sequence of distinct constants. Let $q_{\vec{c}}(\vec{y})$ be the conjunction obtained from $q(\vec{x},\vec{y})$ by replacing, for $i \in \{1, \dots, k\}$, each occurrence of each x_i by c_i. Then,

- $\mathsf{GLB\text{-}CQA}(g(\vec{c}))$ is expressible in $\mathsf{AGGR[FOL]}$ if and only if $\exists\vec{y}(q_{\vec{c}}(\vec{y}))$ is acyclic; and
- If $\mathcal{F}_{\texttt{AGG}} = \mathcal{F}_{\texttt{SUM}}$, then $\mathsf{LUB\text{-}CQA}(g(\vec{c}))$ is expressible in $\mathsf{AGGR[FOL]}$ if and only if $\exists\vec{y}(q_{\vec{c}}(\vec{y}))$ is κ-acyclic. This expressibility (right-to-left implicaiton) holds not only for $\mathcal{F}_{\texttt{SUM}}$, but for every aggregate operator that is monotone and associative.

It is now not hard to show that since $q(\vec{x},\vec{y})$ is self-join-free, different sequences of constants involve the same calculations up to a renaming of constants. This implies that we can perform the calculation once by treating the free variables in $\vec{x}$ as distinct constants. This treatment of free variables is often used in consistent query answering (and in logic in general (Libkin, 2004, Lemma 2.3)); however, it fails for CQA in the presence of self-joins. This is one of the reasons why assuming self-join-freeness serves as a simplifying assumption in much work on CQA.

CHAPTER 9

Conclusions and Open Problems

In this thesis, our goal was to study the expressibility of the function problems $\mathsf{GLB\text{-}CQA}(g())$ and $\mathsf{LUB\text{-}CQA}(g())$ in $\mathsf{AGGR[FOL]}$ for numerical queries $g()$ in the class $\mathsf{AGGR[sjfBCQ]}$. We defined the problems $\mathsf{GLB\text{-}CQA}(g())$ and $\mathsf{LUB\text{-}CQA}(g())$ following the range semantics introduced by Arenas et al. (2001). These function problems take a database instance $\mathbf{db}$ as input, and return, respectively, the greatest lower bound (glb) and the least upper bound (lub) of query answers across all repairs of $\mathbf{db}$. We also studied how the problem $\mathsf{GLB\text{-}CQA}(g())$ is related to the problem $\mathsf{CERTAINTY}(\mathsf{BCQ}(g()))$, obtaining some inexpressibility results for $\mathsf{GLB\text{-}CQA}(g())$ in $\mathsf{AGGR[FOL]}$.

Taking the PhD thesis of Fuxman (2007) as a starting point, we first explored these problems under the hypothesis that $\mathbf{db} \models_{\mathsf{cqa}} \mathsf{BCQ}(g())$, which implies that every aggregation is over a non-empty multiset. We introduced the semantic property of *admitting parsimonious counting*: if a counting query $g() := \mathtt{SUM}(1) \leftarrow q(\vec{u})$ has this property, then on database instances that are “yes”-instances of $\mathsf{CERTAINTY}(\exists\vec{u}(q(\vec{u})))$, the values $\mathsf{GLB\text{-}CQA}(g())$ and $\mathsf{LUB\text{-}CQA}(g())$ can be computed by executing first-order queries followed by simple counting steps. In Theorem 4.0.2, we provided a syntactic characterization of the set of all counting queries in $\mathsf{AGGR[sjfBCQ]}$ that admit parsimonious counting. We then introduced the more general semantic concept of *admitting parsimonious aggregation*, and studied syntactic classes of $\mathtt{AGG}$-queries with this property.

Then, we dropped the hypothesis that $\mathbf{db} \models_{\mathsf{cqa}} \mathsf{BCQ}(g())$ and focused on each of the problems $\mathsf{GLB\text{-}CQA}(g())$ and $\mathsf{LUB\text{-}CQA}(g())$ individually. We proved that, for $\mathtt{AGG}$-queries $g()$ in $\mathsf{AGGR[sjfBCQ]}$ such that $\mathcal{F}_{\mathtt{AGG}}$ is monotone and associative, $\mathsf{GLB\text{-}CQA}(g())$ can be expressed in $\mathsf{AGGR[FOL]}$ if and only if the attack graph of $\mathsf{BCQ}(g())$ is acyclic. This property is decidable in quadratic time in the size of the query. Moreover, an $\mathsf{AGGR[FOL]}$-expression for $\mathsf{GLB\text{-}CQA}(g())$ can be effectively constructed, if it exists, in quadratic time.

Regarding $\mathsf{LUB\text{-}CQA}(g())$, we proved that, for $\mathtt{SUM}$-queries $g()$ in the class $\mathsf{AGGR[sjfBCQ]}$, $\mathsf{LUB\text{-}CQA}(g())$ can be expressed in $\mathsf{AGGR[FOL]}$ if and only if $\mathsf{BCQ}(g())$ is κ-acyclic. This property is decidable in polynomial time in the size of the query. Moreover, an $\mathsf{AGGR[FOL]}$-expression for $\mathsf{LUB\text{-}CQA}(g())$ can be effectively constructed, if it exists, in polynomial time. We also showed that the if-direction holds not only for $\mathtt{SUM}$-queries, but for all $\mathtt{AGG}$-queries such that $\mathcal{F}_{\mathtt{AGG}}$ is monotone and associative.

This thesis mainly focused on monotone and associative aggregate operators, with $\mathtt{SUM}$ as a prototypical case. The complexity of $\mathsf{GLB\text{-}CQA}(g())$ and $\mathsf{LUB\text{-}CQA}(g())$ for non-monotone or non-associative operators remains largely open. Chapter 8 contains some results in this direction.

Another open question concerns shifting our focus from expressibility in $\mathsf{AGGR[FOL]}$ to computability in P. By changing the focus of Theorem 6.0.1, we can formulate the following conjecture: Given a numerical query $g()$ in $\mathsf{AGGR[sjfBCQ]}$ whose aggregate operator is both monotone and associative, $\mathsf{GLB\text{-}CQA}(g())$ is either in P or coNP-hard, and it can be decided which of the two cases holds. We implicitly assume here that the underlying aggregate operator is computable in polynomial time; that is, no exponential behavior is incurred by the aggregation itself. We now outline a possible route to proving this conjecture. In (Koutris & Wijsen, 2017), each cycle in the attack graph of a query $q \in \mathsf{sjfBCQ}$ is classified as either *weak* or *strong*, which is a decidable property. It is then shown that the decision problem $\mathsf{CERTAINTY}(q)$ is in P if q's attack graph contains no strong cycles, and $\mathsf{CERTAINTY}(q)$ is coNP-complete otherwise. Let $g() := \mathtt{AGG}(r) \leftarrow q(\vec{u})$ be a numerical query in $\mathsf{AGGR[sjfBCQ]}$. Since, under some constraints (Theorem 3.5.1), a solution to $\mathsf{GLB\text{-}CQA}(g())$ also solves $\mathsf{CERTAINTY}(\exists\vec{u}(q(\vec{u})))$, it follows that $\mathsf{GLB\text{-}CQA}(g())$ is coNP-hard if the attack graph of $\exists\vec{u}(q(\vec{u}))$ contains a strong cycle. It suffices therefore to establish that if all cycles in the attack graph of $\exists\vec{u}(q(\vec{u}))$ are weak, and $\mathcal{F}_{\mathtt{AGG}}$ is both monotone and associative, then $\mathsf{GLB\text{-}CQA}(g())$ is in P. For attack graphs without cycles, the latter follows from Theorem 6.0.1 under the assumption that $\mathsf{AGGR[FOL]}$ is in P (which requires that aggregate operators are computable in polynomial time). So the remaining problem concerns handling weak cycles in attack graphs, which we anticipate might be solvable by employing the constructs developed in (Koutris & Wijsen, 2021, Section 8) or (Figueira et al., 2023, Section 4). Specificallly, it should be investigated whether ∀embeddings can be generalized to address weak cycles, enabling GLB-CQA($g()$) to be solved along the lines presented in Chapter 6 of this thesis—i.e., by computing the smallest aggregated value over all $\subseteq$-maximal consistent sets of ∀embeddings.

Finally, a more ambitious open problem is to syntactically characterize the class of all (i.e., not necessarily in AGGR[sjfBCQ]) numerical queries for which GLB-CQA($g()$) and LUB-CQA($g()$) can be expressed in AGGR[FOL]. This problem is largely open, because it is already a notorious open problem to syntactically characterize the class of conjunctive queries that have a consistent first-order rewriting.

Appendices

APPENDIX **A**

Helping Constructs and Lemmas

We recall some definitions from graph theory. In a directed acyclic graph (DAG), a vertex with zero indegree is called a *source*. If there is a directed path from u to v in a DAG, with $u \neq v$, then v is called a *descendant* of u, and u an *ancestor* of v.

The following lemma states that whenever R and S are distinct sources that are weakly connected in a DAG, then there is a sequence of sources that starts with R and ends with S, such that every two adjacent vertices in the sequence have a descendant in common.

Lemma A.0.1 (Sources of a graph are linked by sources)**.** *Let $G = (V, E)$ be a DAG. Let $R, S \in V$ be distinct sources that are weakly connected. There is a sequence $(H_1, \ldots, H_n)$ of sources such that $H_1 = R$, $H_n = S$, and every two adjacent sources in the sequence have a common descendant.*

Proof. Since R and S are weakly connected, there is a sequence $(F_1, \ldots, F_n)$ of atoms in V such that $F_1 = R$, $F_n = S$ and for every $i \in \{1, \ldots, n-1\}$, either $(F_i, F_{i+1}) \in E$ or $(F_{i+1}, F_i) \in E$. We construct a new sequence by repeating the following step as long as possible: replace some contiguous subsequence $(I, \ldots, J)$ with (I, J) if I is a descendant or an ancestor of J. Let the new sequence be $(G_1, \ldots, G_m)$. By construction, for all $i \in \{1, \ldots, m-2\}$,

- if G_i is a descendant of G_{i+1}, then G_{i+1} is an ancestor of G_{i+2}; and
- if G_i is an ancestor of G_{i+1}, then G_{i+1} is a descendant of G_{i+2}.

It is easily verified that $G_1 = R$, $G_m = S$, and that m is an odd number. By construction, for every $i \in \{1, \ldots, m\}$ such that i is even, we have that

G_i is a common descendant of G_{i-1} and G_{i+1}. We now change the sequence $(G_1, \ldots, G_m)$ as follows: for every $j \in \{1, \ldots, m\}$ such that j is odd, if G_j is not a source, replace G_j by a source that is an ancestor of G_j (such a source obviously exists in a DAG). It is easily verified that if we now omit the vertices that are not sources (i.e., $G_2, G_4, \ldots, G_{m-1}$), we obtain a sequence of sources in which every two adjacent sources have a descendant in common. $\square$

The following lemma states that sequential proofs provide a sound and complete characterization of logical implication.

Lemma A.0.2. *Let q be a self-join free conjunctive query. Let $Z \subseteq \mathsf{vars}(q)$ and $w \in \mathsf{vars}(q)$. Then the following are equivalent:*

1. *$\mathcal{K}(q) \models Z \rightarrow w$; and*
2. *there is a sequence $(F_1, \ldots, F_n)$ of atoms that is a sequential proof of $\mathcal{K}(q) \models Z \rightarrow w$.*

Proof. Sequential proofs mimic a standard algorithm for logical implication of functional dependencies; see for example (Abiteboul et al., 1995, Algorithm 8.2.7). $\square$

Lemma A.0.3 (Attacks with same endpoint)**.** *Let q be a self-join-free conjunctive query. Let F and G be two distinct atoms in q that both attack a same atom. If $F \overset{q}{\not\rightsquigarrow} G$, then either $G \overset{q}{\rightsquigarrow} F$ or $\mathsf{Key}(G) \subseteq F^{+,q}$.*

Proof. Assume $F \overset{q}{\not\rightsquigarrow} G$. There is H such that $F \overset{q}{\rightsquigarrow} H$ and $G \overset{q}{\rightsquigarrow} H$. There is a sequence $(x_0, x_1, \ldots, x_n)$ $(n \geqslant 0)$ of bound variables not in $F^{+,q}$ such that $x_0 \in \mathsf{notKey}(R)$, $x_n \in \mathsf{vars}(H)$, and every two adjacent variables occur together in some atom of q. Likewise, there is a sequence $(y_0, y_1, \ldots, y_m)$ $(m \geqslant 0)$ of bound variables not in $G^{+,q}$ such that $y_0 \in \mathsf{notKey}(G)$, $y_m \in \mathsf{vars}(H)$, and every two adjacent variables occur together in some atom of q. Clearly, for every $i \in \{0, 1, \ldots, m\}$, $S \overset{q}{\rightsquigarrow} y_i$. In the sequence $(x_0, x_1, \ldots, x_n, y_m, y_{m-1}, \ldots, y_0)$, it holds that $x_0 \in \mathsf{notKey}(F)$, $y_0 \in \mathsf{notKey}(G)$, and every two adjacent variables occur together in some atom of q. By our hypothesis that $F \overset{q}{\not\rightsquigarrow} G$, there is $i \in \{0, \ldots, m\}$ such that $y_i \in F^{+,q}$. By Lemma A.0.2, there exists a shortest sequence $(H_1, H_2, \ldots, H_\ell)$ that is a sequential proof of $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F) \rightarrow y_i$. Two cases are possible.

Case that $\ell = 0$. Then $y_i \in \mathsf{Key}(F)$. From $G \overset{q}{\rightsquigarrow} y_i$, it follows $G \overset{q}{\rightsquigarrow} F$.

Case that $\ell > 0$. If $S \in \{H_1, \dots, H_\ell\}$, then by Lemma A.0.2 it is correct to conclude $\mathsf{Key}(G) \subseteq F^{+,q}$, and the desired result obtains. Assume $G \notin \{H_1, \dots, H_\ell\}$ from here on. For technical reasons, define $H_0 := F$. We show that for every $k \in \{1, \dots, \ell\}$, the following holds true:

> *Back Property:* if $G \overset{q}{\rightsquigarrow} H_k$, then there is $g < k$ such that $G \overset{q}{\rightsquigarrow} H_g$.

To this end, assume $G \overset{q}{\rightsquigarrow} H_k$ with $k \geqslant 1$. By definition of attacks, there is a bound variable $u \in \mathsf{Key}(H_k)$ such that $G \overset{q}{\rightsquigarrow} u$. If $u \in \mathsf{vars}(F)$, then the desired result obtains because we let $H_0 = F$. Assume $u \notin \mathsf{vars}(F)$ from here on. Then there exists $g \in \{1, 2, \dots, k-1\}$ such that $u \in \mathsf{notKey}(H_g)$. Informally, H_g is the atom that introduces u in $F^{+,q}$. Since $G \overset{q}{\rightsquigarrow} u$ and $H_g \neq S$, it follows $G \overset{q}{\rightsquigarrow} H_g$. This concludes the proof of the *Back Property.*

Since our sequential proof is as short as possible, $y_i \in \mathsf{notKey}(H_\ell)$. From $G \overset{q}{\rightsquigarrow} y_i$ and $H_\ell \neq G$, it follows $G \overset{q}{\rightsquigarrow} H_\ell$. By repeated application of the *Back Property*, we obtain $G \overset{q}{\rightsquigarrow} H_0$ with $H_0 = F$, as desired. This concludes the proof. □

APPENDIX B

Proofs of Chapter 4

B.1. Proof of Proposition 4.1.2

We present a number of helping lemmas.

Lemma B.1.1. *Let $q \in \mathsf{Cparsimony}$. Let $\vec{x}$ be a minimal id-set for q. Then, for every variable v in $\vec{x}$, we have $\mathcal{K}(q) \not\models \emptyset \rightarrow v$. Consequently, a minimal id-set contains no frozen variables.*

Proof. Assume for the sake of contradiction that there is a variable v in $\vec{x}$ such that $\mathcal{K}(q) \models \emptyset \rightarrow v$. It can be easily verified that $\vec{x} \setminus \{v\}$ still satisfies conditions (1) and (2) in Definition 4.1.3, contradicting that $\vec{x}$ is minimal. ☐

Lemma B.1.2. *Let q be a self-join-free conjunctive query. If a variable v of q is attacked in q, then $v \notin \mathsf{frozen}(q)$.*

Proof. Assume $F \overset{q}{\rightsquigarrow} v$ for some atom F. Assume for the sake of contradiction that $v \in \mathsf{frozen}(q)$. Then there is a sequential proof of $\mathcal{K}(q) \models \emptyset \rightarrow v$ such that no atom in the sequential proof attacks v. Since $F \overset{q}{\rightsquigarrow} v$, we have that F is not used in the sequential proof. Consequently, $\mathcal{K}(q \setminus \{F\}) \models \emptyset \rightarrow v$, and hence $v \in F^{+,q}$. Consequently, $F \overset{q}{\not\rightsquigarrow} v$, a contradiction. ☐

Lemma B.1.3. *Let $q \in \mathsf{Cparsimony}$. Let $\vec{x}$ be a minimal id-set for q. For every variable x of $\vec{x}$, the following hold:*

(a) if x occurs in an atom F of q, then x occurs at a primary-key position of R;

(b) x is unattacked; and

(c) x occurs at a primary-key position in an unattacked atom of q.

Proof. $\boxed{\text{Proof of (a)}}$ Assume for the sake of contradiction that for some atom F in q, $\vec{x}$ contains some variable x of $\mathsf{notKey}(F)$. Since x is a bound variable, the Gaifman graph of q contains an empty path from x to x, which is a path from a variable in $\mathsf{notKey}(F)$ to a variable of $\vec{x}$ that uses no variable of $\mathsf{Key}(F)$. From Lemma B.1.1, it follows that x is not frozen. It follows that condition (2) in Definition 4.1.3 is violated, a contradiction.

$\boxed{\text{Proof of (b)}}$ Assume for the sake of contradiction that some variable x of $\vec{x}$ is attacked. There is an atom F in q that attacks x, with a path $(x_1, \dots, x_n)$ in the Gaifman graph of q between some variable in $\mathsf{notKey}(F)$ and x such that for every $i \in \{1, \dots, n\}$, $x_i \notin F^{+,q}$. Since $\mathsf{Key}(F) \subseteq F^{+,q}$, no x_i in this sequence occurs in $\mathsf{Key}(F)$. By Lemma B.1.2, no x_i is frozen. It follows that condition (2) in Definition 4.1.3 is violated, a contradiction.

$\boxed{\text{Proof of (c)}}$ By item (a), we can assume an atom F such that $x \in \mathsf{Key}(R)$. If F is unattacked, then the desired result obtains. Assume from here on that F is attacked by an atom G in q. By definition of an attack graph, there is a a path $(v_1, \dots, v_n)$ in the Gaifman graph of q between some variable in $\mathsf{notKey}(G)$ and some variable in $\mathsf{Key}(F)$ such that for every $i \in \{1, \dots, n\}$, $v_i \notin G^{+,q}$, and therefore $v_i \notin \mathsf{Key}(G)$. By Lemma B.1.2, no v_i is frozen. From Lemma B.1.1, it follows that x is not frozen. If $x \notin \mathsf{Key}(G)$, then $(v_1, \dots, v_n, x)$ is a (not necessarily simple) path in the Gaifman graph from a variable in $\mathsf{notKey}(G)$ to a variable of $\vec{x}$ that uses no variable of $\mathsf{Key}(G)$, contradicting condition (2) in Definition 4.1.3. We conclude by contradiction that $x \in \mathsf{Key}(G)$. If G is unattacked, then the desired result obtains. Otherwise there is an atom H that attacks G, and we can repeat the same reasoning as before. The same reasoning can however not be applied forever, since the attack graph of q is finite and contains no cycles. Therefore, at some point we will find an unattacked atom whose primary key contains x. □

We can now give the proof of Proposition 4.1.2.

Proof of Proposition 4.1.2. $\boxed{V \subseteq \mathsf{vars}(\vec{x}).}$ Let $v \in V$. By the construction of V, we can assume an unattacked atom F such that v is a bound variable in $\mathsf{Key}(F) \setminus N$. By condition (1) in Definition 4.1.3, we have $\mathcal{K}(q) \models \vec{x} \rightarrow v$. Since $v \notin N$, $\mathcal{K}(q) \models \vec{x} \rightarrow v$ implies that v belongs to $\vec{x}$.

$\boxed{\mathsf{vars}(\vec{x}) \subseteq V.}$ Let x be an arbitrary variable in the sequence $\vec{x}$. By item (c) in Lemma B.1.3, we can assume an unattacked atom F such that $x \in \mathsf{Key}(F)$.

By item (a) in Lemma B.1.3, $x \notin N$. Then, by construction of V, we have $x \in V$.

$\boxed{\text{Proof of (B).}}$ Assume F, G are unattacked atoms that are weakly connected in q's attack graph. We show $\mathsf{Key}(F) \cap \vec{x} \subseteq \mathsf{Key}(G)$. The desired result then follows by symmetry. By Lemma A.0.1, there is a sequence of weakly connected, unattacked atoms $(H_1, H_2, \ldots, H_n)$ such that $H_1 = F$, $H_n = G$, and every two adjacent atoms have a descendant in common. We will show that $\mathsf{Key}(H_1) \cap \vec{x} \subseteq \mathsf{Key}(H_2)$. We can assume that H_{1-2} is a descendant shared by H_1 and H_2. Since the attack graph of q is acyclic, it is transitive by (Koutris & Wijsen, 2017, Lemma 3.5). Assume $x \in \mathsf{Key}(H_1) \cap \vec{x}$. Since $H_1 \overset{q}{\rightsquigarrow} H_{1-2}$, there is a path $(x_1, \ldots, x_n)$ in the Gaifman graph of q between some variable in $\mathsf{notKey}(H_1)$ and some variable in $\mathsf{Key}(H_{1-2})$. It follows that H_1 attacks x_i for every $i \in \{1, \ldots, n\}$. Since $H_2 \overset{q}{\rightsquigarrow} H_{1-2}$, there is a path $(y_1, \ldots, y_m)$ in the Gaifman graph of q between some variable in $\mathsf{notKey}(H_2)$ and some variable in $\mathsf{Key}(H_{1-2})$ such that, for every $i \in \{1, \ldots, m\}$, $y_i \notin {H_2}^{+,q}$. It follows that $y_1, \ldots, y_m \notin \mathsf{Key}(H_2)$. By Lemma B.1.2, no variable in $\{x_1, \ldots, x_n, y_1, \ldots, y_m\}$ is frozen. Then $(x, x_1, \ldots, x_n, y_m, \ldots, y_1)$ is a path in the Gaifman graph of q between x and y_1. For every $i \in \{1, \ldots, n\}$, $x_i \notin \mathsf{Key}(H_2)$, or else $H_1 \overset{q}{\rightsquigarrow} H_2$, a contradiction. Assume towards a contradiction that $x \notin \mathsf{Key}(H_2)$. The reverse path is a path from the variable $y_1 \in \mathsf{notKey}(H_2)$ to a variable of $\vec{x}$ that uses no variable of $\mathsf{Key}(H_2) \cup \mathsf{frozen}(q)$. By condition (2) in Definition 4.1.3, $\vec{x}$ is not an id-set, a contradiction. We conclude by contradiction that $x \in \mathsf{Key}(H_2)$.

By repeating the same reasoning, we obtain $\mathsf{Key}(H_2) \cap \vec{x} \subseteq \mathsf{Key}(H_3)$, $\mathsf{Key}(H_3) \cap \vec{x} \subseteq \mathsf{Key}(H_4)$,... It follows $\mathsf{Key}(H_1) \cap \vec{x} \subseteq \mathsf{Key}(H_n)$. □

B.2. Proof of Lemma 4.1.6

Proof of Lemma 4.1.6. Since $\mathbf{db} \models q'(\vec{d_i})$ for every $i \in \{1, \ldots, n\}$, there is a sequence $(\theta_1, \ldots, \theta_n)$ of embeddings of q in $\mathbf{db}$ such that for every $i \in \{1, \ldots, n\}$, condition (a) and condition (c) are satisfied. We can assume, without loss of generality, that for every $i, j \in \{1, \ldots, n\}$, if $i \leq j$ then $\theta_i(r) \geq \theta_j(r)$. In other words, the sequence $(\theta_1, \ldots, \theta_n)$ is sorted in descending order with respect to the r-value of each θ_i. Assume that condition (b) is violated. Then there are $i, j \in \{1, \ldots, n\}$ such that for some atom F in q, $\theta_i(F)$ and $\theta_j(F)$ are key-equal but distinct. Let M be the smallest subset of atoms in q such that:

- for every atom F in q, if $\theta_i(F)$ and $\theta_j(F)$ are key-equal but distinct, then $F \in M$ (hence $M \neq \emptyset$); and

- *Closure Property:* if M contains G and $\theta_i(v) \neq \theta_j(v)$ for some $v \in \mathsf{vars}(G)$, then M contains every atom of q in which v occurs.

We show that θ_i and θ_j agree on every variable of $\vec{x}$ that occurs in M. Assume for the sake of contradiction that some variable x of $\vec{x}$ satisfies $\theta_i(x) \neq \theta_j(x)$ and occurs in some atom M. By construction of M, there is an atom F_0 in q, and a sequence of variables $(v_1, \ldots, v_n)$, with $n \geq 1$, such that:

- $v_n = x$;
- for every $k \in \{1, \ldots, n\}$, $\theta_i(v_k) \neq \theta_j(v_k)$;
- $\theta_i(F_0)$ and $\theta_j(F_0)$ are key-equal but distinct, and $v_1 \in \mathsf{notKey}(F_0)$;
- every two adjacent variables in the sequence occur together in some atom of M.

Note that the sequence may reduce to (v_1) with $v_1 = x$. Then $(v_1, \ldots, v_n)$ is a path in the Gaifman graph of q that uses no variable of $\mathsf{Key}(F_0)$.

We show that no variable among $v_1, \ldots, v_n$ is frozen. Assume for the sake of contradiction that some v_k is frozen ($1 \leq k \leq n$). Let $q^*(v_k) := \nexists v_k\,[q]$. Since $\mathbf{db} \models_{\mathsf{cqa}} q$, it follows from (the proof of) (Koutris & Wijsen, 2021, Lemma 11) that for all constants f_1, f_2, if $\mathbf{db} \models q^*(f_1)$ and $\mathbf{db} \models q^*(f_2)$, then $f_1 = f_2$. Since $\mathbf{db} \models q^*(\theta_i(v_k))$ and $\mathbf{db} \models q^*(\theta_j(v_k))$, it follows $\theta_i(v_k) = \theta_j(v_k)$, a contradiction. We conclude by contradiction that no variable among $v_1, \ldots, v_n$ is frozen.

Then $(v_1, \ldots, v_n)$ is a path in the Gaifman graph of q between a variable of $\mathsf{notKey}(F_0)$ and a variable of $\vec{x}$ that uses no variable of $\mathsf{Key}(F_0) \cup \mathsf{frozen}(q)$. Consequently, $\vec{x}$ violates condition (2) in Definition 4.1.3. Then $\vec{x}$ is not an id-set, a contradiction. We conclude by contradiction that θ_i and θ_j agree on every variable of $\vec{x}$ that occurs in M.

Assume $i < j$. Let θ_j^* be a valuation over $\mathsf{vars}(q)$ such that for every $v \in \mathsf{vars}(q)$,

$$\theta_j^*(v) = \begin{cases} \theta_i(v) & \text{if } x \text{ occurs in some atom of } M; \\ \theta_j(v) & \text{otherwise.} \end{cases}$$

Since θ_i and θ_j agree on every variable of $\vec{x}$ that occurs in M, it follows $\theta_j^*(\vec{x}) = \theta_j(\vec{x})$, and thus $\theta_j^*(\vec{x}) = \vec{d_j}$. We now argue that $\theta_j^*(F) \in \mathbf{db}$ for every atom F in q. We consider two cases:

Case $F \in M$. Then $\theta_j^*(F) = \theta_i(F)$, and we have that $\theta_i(F) \in \mathbf{db}$;

Case $F \notin M$. Then $\theta_j^*(F) = \theta_j(F)$, and we have that $\theta_j(F) \in \mathbf{db}$. Note here that if some variable u of $\mathsf{vars}(F)$ occurs in M, then $F \notin M$ implies $\theta_i(u) = \theta_j(u)$.

Finally, we argue that $\theta_j^*(r) = \theta_j(r)$. Assume for the sake of contradiction that $\theta_j^*(r) \neq \theta_j(r)$. By construction of θ_j^*, it follows that $\theta_j^*(r) = \theta_i(r)$ and $\theta_i(r) \neq \theta_j(r)$. Since $i < j$, it follows that $\theta_i(r) > \theta_j(r)$ and, thus, $\theta_j^*(r) > \theta_j(r)$. We have that θ_j^* is an embedding of q in $\mathbf{db}$ such that $\theta_j^*(\vec{x}) = \vec{d_j}$ and $\theta_j^*(r) > \theta_j(r)$; which contradicts our hypothesis.

The sequence $\Big(\theta_1, \dots, \theta_i, \dots, \theta_{j-1}, \theta_j^*, \theta_{j+1}, \dots, \theta_n\Big)$ therefore is a sequence of embeddings of q in $\mathbf{db}$ satisfying condition (a) and condition (c). By our construction, for every atom F in q, if the facts $\theta_j^*(F)$ and $\theta_i(F)$ are key-equal, then they are equal. By repeatedly removing primary-key violations in this way, we eventually obtain a sequence that also satisfies (b). Note that our procedure will not be trapped in an infinite loop, because whenever two distinct key-equal facts have to be made equal, say $\theta_i(F)$ and $\theta_j(F)$, then only the valuation with the larger index will be modified. This means that θ_1 will never be modified; θ_2 will only be modified with respect to θ_1; and so on. $\square$

B.3. Proof of Lemma 4.1.14

Proof of Lemma 4.1.14. We can assume an atom F in q such that the Gaifman graph of q contains a path $\pi = (v_1, v_2, \dots, v_n)$ from some variable v_1 in $\mathsf{notKey}(F)$ to a variable v_n in $\vec{x}$ such that $v_1, \dots, v_n \notin \mathsf{Key}(F) \cup \mathsf{frozen}(q)$. Let $U = \{u \in \mathsf{vars}(q) \mid \mathcal{K}(q) \models \emptyset \rightarrow u\}$. We distinguish two cases.

Case that $v_1, \dots, v_n \notin U$. Let θ, μ, γ be three valuations over $\mathsf{vars}(q)$ such that for every variable $u \in \mathsf{vars}(q)$,

$$\theta(u) \neq \mu(u) \quad \text{if and only if} \quad u \in \{v_1, \dots, v_n\}; \tag{B.1}$$

$$\gamma(u) = \theta(u) \quad \text{if and only if} \quad u \in U; \tag{B.2}$$

$$\gamma(u) = \mu(u) \quad \text{if and only if} \quad u \in U. \tag{B.3}$$

The valuations are well-defined because $v_1, \dots, v_n \notin U$. We can choose θ, μ, γ as follows:

			variables of U			other variables			
v_1	$\cdots$	v_n	v_{n+1}	$\cdots$	v_m	v_{m+1}	$\cdots$	v_ℓ	
0	$\cdots$	0	0	$\cdots$	0	0	$\cdots$	0	(θ)
1	$\cdots$	1	0	$\cdots$	0	0	$\cdots$	0	(μ)
2	$\cdots$	2	0	$\cdots$	0	2	$\cdots$	2	(γ)

Let $\mathbf{db} = \theta(q) \cup \mu(q) \cup \gamma(q)$. We show that every repair of $\mathbf{db}$ includes $\gamma(q)$, and hence $\mathbf{db} \models_{\mathsf{cqa}} q$. To this end, let G be an atom of q. If $\mathsf{Key}(G) \nsubseteq U$, then by Eq. (B.2) and Eq. (B.3), $\gamma(G)$ is key-equal to neither $\theta(G)$ nor $\mu(G)$, and, consequently, every repair of $\mathbf{db}$ will contain $\gamma(G)$. Assume next that $\mathsf{Key}(G) \subseteq U$. Since $\mathcal{K}(q)$ contains $\mathsf{Key}(G) \rightarrow \mathsf{vars}(G)$, it follows that $\mathsf{vars}(G) \subseteq U$. Then, $\gamma(G) = \theta(G) = \mu(G)$ by Eq. (B.2) and Eq. (B.3).

Note that $\theta(F)$ and $\mu(F)$ are key-equal, because the path π contains no variable of $\mathsf{Key}(F)$. Also $\theta(F) \neq \mu(F)$, because $v_1 \in \mathsf{notKey}(F)$ and $\theta(v_1) \neq \mu(v_1)$. It follows that every repair must contain either $\theta(F)$ or $\mu(F)$, but not both.

Since $\theta(v_n) \neq \mu(v_n)$, we have $\theta(\vec{x}) \neq \mu(\vec{x})$. Let $\vec{b_\theta} = \theta(\vec{x})$ and $\vec{b_\mu} = \mu(\vec{x})$. Clearly, $\mathbf{db} \models q'(\vec{b_\theta})$ and $\mathbf{db} \models q'(\vec{b_\mu})$.

Let $\mathbf{r}$ be a repair that contains $\mu(F)$, and hence $\theta(F) \notin \mathbf{r}$. We show $\mathbf{r} \not\models q'(\vec{b_\theta})$. Assume for the sake of contradiction that there is a valuation ν over $\mathsf{vars}(q)$ such that $\nu(q) \subseteq \mathbf{r}$ and $\nu(\vec{x}) = \vec{b_\theta}$. We have that ν maps the sequence π to $(\nu(v_1), \dots, \nu(v_n))$. We have $\nu(v_n) = \theta(v_n)$. Since every two adjacent variables occur together in some atom, it follows from Eq. (B.1) that $\nu(v_1) = \theta(v_1)$, hence $\nu(F) = \theta(F)$, a contradiction. We conclude by contradiction that $\mathbf{r} \not\models q'(\vec{b_\theta})$. By similar reasoning, for a repair $\mathbf{r}$ that contains $\theta(F)$, we have $\mathbf{r} \not\models q'(\vec{b_\mu})$. It follows that for every repair $\mathbf{r}$ of $\mathbf{db}$, either $\mathbf{r} \not\models q'(\vec{b_\theta})$ or $\mathbf{r} \not\models q'(\vec{b_\mu})$.

Case that $\{v_1, \dots, v_n\} \cap U \neq \emptyset$. We can assume a greatest integer $1 \leq \ell \leq n$ such that $v_\ell \in U$. That is, $v_{\ell+1}, v_{\ell+2}, \dots, v_n \notin U$. We can assume a sequence σ of atoms that is a sequential proof of $\mathcal{K}(q) \models \emptyset \rightarrow v_\ell$. Since v_ℓ is not frozen, there is some atom G in the sequential proof σ such that $G \overset{q}{\rightsquigarrow} v_\ell$, and hence $v_\ell \notin G^{+,q}$.

We show $v_{\ell+1}, \dots, v_n \notin G^{+,q}$. Assume for the sake of contradiction $j \in \{\ell+1, \dots, n\}$ such that $\mathcal{K}(q \setminus \{G\}) \models \mathsf{Key}(G) \rightarrow v_j$. Since $\mathcal{K}(q \setminus \{G\}) \models \emptyset \rightarrow \mathsf{Key}(G)$ (because G occurs in the sequential proof σ), we obtain $\mathcal{K}(q \setminus \{G\}) \models \emptyset \rightarrow v_j$, hence $v_j \in U$, a contradiction. Consequently, $\ell \leq n$ and $v_\ell, v_{\ell+1}, \dots, v_n \notin G^{+,q}$.

Let θ, μ be two valuations over $\mathsf{vars}(q)$ such that for every $u \in \mathsf{vars}(q)$, $\theta(u) = \mu(u)$ if and only if $u \in G^{+,q}$.

Let $\mathbf{db} = \theta(q) \cup \mu(q)$. The only facts in $\mathbf{db}$ that are key-equal but distinct are $\theta(G)$ and $\mu(G)$. Note here that since G attacks some variable, there is a variable of $\mathsf{notKey}(G)$ that is not in $G^{+,q}$. Consequently, $\mathbf{db}$ has exactly two repairs, denoted $\mathbf{r}_1 := \mathbf{db} \setminus \{\mu(G)\}$ and $\mathbf{r}_2 := \mathbf{db} \setminus \{\theta(G)\}$.

Let $\vec{d_1} = \theta(\vec{x})$ and $\vec{d_2} = \mu(\vec{x})$. Since $v_n \notin G^{+,q}$, we have $\theta(v_n) \neq \mu(v_n)$. Since $v_n \in \mathsf{vars}(\vec{x})$, $\vec{d_1} \neq \vec{d_2}$. For the query $q'(\vec{x}) := \nexists \vec{x}\,[q]$, we obviously have $\mathbf{db} \models q'(\vec{d_1})$ and $\mathbf{db} \models q'(\vec{d_2})$. Using the same proof as (Wijsen, 2012, Proposition 6.4), we have $\mathbf{r}_1 \models q'(\vec{d_1})$ and $\mathbf{r}_1 \not\models q'(\vec{d_2})$. Symmetrically, $\mathbf{r}_2 \not\models q'(\vec{d_1})$ and $\mathbf{r}_2 \models q'(\vec{d_2})$. Clearly, $\mathbf{db} \models_{\mathsf{cqa}} q$.

In both cases, there is a database $\mathbf{db}$ such that $\mathbf{db} \models_{\mathsf{cqa}} q$, and $\mathbf{db}$ has no optimistic repair with respect to $q'(\vec{x})$. □

B.4. Proof of Theorem 4.1.15

Lemma B.4.1. *Let q be a* sjfBCQ *in* $\mathsf{Cforest}$*, and let $(F_1, \dots, F_n)$ be a path in the Fuxman graph of q. Then, $\mathcal{K}(q) \models \mathsf{Key}(F_1) \rightarrow \mathsf{Key}(F_n)$.*

Proof. By definition of a Fuxman graph, we have that for every $i \in \{1, \dots, n-1\}$, $\mathsf{Key}(F_{i+1}) \subseteq \mathsf{notKey}(F_i)$. This means that, for every $i \in \{1, \dots, n-1\}$, $\mathcal{K}(q) \models \mathsf{Key}(F_i) \rightarrow \mathsf{vars}(F_{i+1})$. By Armstrong's transitivity axiom, $\mathcal{K}(q) \models \mathsf{Key}(F_1) \rightarrow \mathsf{Key}(F_n)$. □

Lemma B.4.2. *Let q be a* sjfBCQ *in* $\mathsf{Cforest}$. *Let $x, y \in \mathsf{vars}(q)$ be distinct variables that are connected in the Gaifman graph of q. Let F, G be two distinct atoms such that $x \in \mathsf{notKey}(F)$ and $y \in \mathsf{vars}(G)$. Then, the Fuxman graph of q has a directed path from F to G.*

Proof. We can assume a shortest path $(v_1, \dots, v_n)$ in the Gaifman graph of q, where $v_1 = x$ and $v_n = y$. We can assume a sequence $(F_1, \dots, F_{n+1})$ of atoms in q such that:

- $F_1 = F$;
- $F_{n+1} = G$; and
- for every $i \in \{1, \dots, n\}$, $v_i \in \mathsf{vars}(F_i) \cap \mathsf{vars}(F_{i+1})$.

We will show by induction on increasing $i \in \{1, \dots, n\}$, that $(F_1, \dots, F_{i+1})$ is a path in the Fuxman graph of q.

$\boxed{\textit{Basis } i = 1.}$ Since $v_1 \notin \mathsf{Key}(F_1)$ and $v_1 \in \mathsf{vars}(F_1) \cap \mathsf{vars}(F_2)$, there is a directed edge from F_1 to F_2 in the Fuxman graph of q. $\boxed{\textit{Induction step } i-1 \rightarrow i.}$ The induction hypothesis is that $(F_1, \dots, F_i)$ is a path in the Fuxman graph of

q. We have that $v_i \in \mathsf{vars}(F_i) \cap \mathsf{vars}(F_{i+1})$. Assume for the sake of contradiction that $v_i \in \mathsf{Key}(F_i)$. Then, by definition of $\mathsf{Cforest}$, $v_i \in \mathsf{vars}(F_{i-1})$. Since we also have $v_{i-2} \in \mathsf{vars}(F_{i-1})$, the Gaifman graph of q contains an edge between v_{i-2} and v_i. Then, $(v_1, \ldots, v_{i-2}, v_i, \ldots, v_n)$ is a shorter path in the Gaifman graph of q between v_1 and v_n, a contradiction. We conclude by contradiction that $v_i \notin \mathsf{Key}(F_i)$. It follows that the Fuxman graph of q has a directed edge from F_i to F_{i+1} and, consequently, $(F_1, \ldots, F_{i+1})$ is a path in the Fuxman graph of q. By induction, we obtain that $(F_1, \ldots, F_{n+1})$ is a path in the Fuxman graph of q from F to G. □

Lemma B.4.3. *Let q be a* sjfBCQ *in* $\mathsf{Cforest}$. *Let F, G be two distinct atoms such that $F \overset{q}{\rightsquigarrow} G$. Then, the Fuxman graph of q has a directed path from F to G.*

Proof. Since $F \overset{q}{\rightsquigarrow} G$, there is a path $(v_1, \ldots, v_n)$ in the Gaifman graph of q such that $v_1 \in \mathsf{notKey}(F)$, $v_n \in \mathsf{Key}(G)$ and, for every $i \in \{1, \ldots, n\}$, $v_i \notin F^{+,q}$. The desired result then follows by Lemma B.4.2. □

Lemma B.4.4. *Every query in* $\mathsf{Cforest}$ *has an acyclic attack graph.*

Proof. Fuxman (2007) has shown that every query in $\mathsf{Cforest}$ has a consistent first-order rewriting. From Theorem 2.0.1, it follows that every query in $\mathsf{Cforest}$ has an acyclic attack graph. □

Lemma B.4.5. *Let q be a* $\mathsf{Cforest}$. *Let F be an atom in q and let G be the root atom in the tree of the Fuxman graph of q where F appears. Then, $\mathcal{K}(q) \models \mathsf{Key}(G) \rightarrow \mathsf{vars}(F)$.*

Proof. If $F = G$, then clearly $\mathcal{K}(q) \models \mathsf{Key}(G) \rightarrow \mathsf{vars}(F)$. Assume that $F \neq G$. It follows that there is a path between G and F in the Fuxman graph of q. From Lemma B.4.1, we have $\mathcal{K}(q) \models \mathsf{Key}(G) \rightarrow \mathsf{Key}(F)$. □

Lemma B.4.6. *Let q be a* sjfBCQ *in* $\mathsf{Cforest}$. *Let $\vec{x}$ be a $\subseteq$-minimal tuple of bound variables such that for every root R in the Fuxman graph of q, $\vec{x}$ contains every bound variable of* $\mathsf{Key}(R)$. *Then, $\vec{x}$ verifies conditions (1) and (2) in Definition 4.1.3.*

Proof. From Lemma B.4.5, it follows that $\vec{x}$ verifies condition (1) in Definition 4.1.3. We show that $\vec{x}$ verifies condition (2) in Definition 4.1.3. Assume towards a contradiction that there is an atom F in q, and a path $(v_1, \ldots, v_n)$ in the Gaifman graph of q such that:

- $v_n \in \vec{x}$;

- $v_1 \in \mathsf{notKey}(F)$; and
- for every $i \in \{1, \ldots, n\}$, $v_i \notin \mathsf{Key}(F)$.

Since $\vec{x}$ is $\subseteq$-minimal, we can assume an atom G that is a root in q's Fuxman graph such that $v_n \in \mathsf{Key}(G)$. Since $v_n \notin \mathsf{Key}(G)$, it follows $F \neq G$. By Lemma B.4.2, there is a directed path from F to G in the Fuxman graph of q, contradicting that G is a root. $\square$

Proof of Theorem 4.1.15. From Lemma B.4.4 and Lemma B.4.6, it follows that $\mathsf{Cforest} \subseteq \mathsf{Cparsimony}$. The inclusion is strict, because $\exists x \exists y (R(\underline{x}, y) \wedge S(\underline{x}, y))$ belongs to $\mathsf{Cparsimony} \setminus \mathsf{Cforest}$. $\square$

APPENDIX C

Proofs of Chapter 6

C.1. Helping Lemmas

We introduce two helping lemmas that will be used later on.

Lemma C.1.1. *Let q be query in* sjfBCQ. *For some $w \in \mathsf{vars}(q)$, $Z \subseteq \mathsf{vars}(q)$, $G \in q$, let $(F_1, F_2, \dots, F_n)$ be a sequential proof of $\mathcal{K}(q \setminus \{G\}) \models Z \rightarrow w$. If $G \overset{q}{\rightsquigarrow} x$ for some variable x in $Z \cup (\bigcup_{i=1}^{n} \mathsf{vars}(F_i))$, then there is $z \in Z$ such that $G \overset{q}{\rightsquigarrow} z$.*

Proof. Assume that $G \overset{q}{\rightsquigarrow} x$ for some variable x in $Z \cup (\bigcup_{i=1}^{n} \mathsf{vars}(F_i))$. The proof runs by induction on increasing n. For the basis of the induction, $n = 0$, we obtain $x \in Z$ with $G \overset{q}{\rightsquigarrow} x$, as desired.

For the induction step, $n - 1 \rightarrow n$, assume that the lemma holds for sequential proofs of length $< n$. Notice that for every $m \in \{0, 1, \dots, n-1\}$, there is some w' such that $(F_1, F_2, \dots, F_m)$ is a sequential proof of $\mathcal{K}(q \setminus \{G\}) \models Z \rightarrow w'$. Therefore, if x occurs in $Z \cup \left(\bigcup_{i=1}^{n-1} \mathsf{vars}(F_i)\right)$, then the desired result holds by the induction hypothesis. Assume from here on that $x \notin Z \cup \left(\bigcup_{i=1}^{n-1} \mathsf{vars}(F_i)\right)$. Then, $x \in \mathsf{notKey}(F_n)$. From $G \overset{q}{\rightsquigarrow} x$, it follows that there is $x' \in \mathsf{Key}(F_n)$ such that $G \overset{q}{\rightsquigarrow} x'$. By the definition of sequential proof, it must be the case that $x' \in Z \cup \left(\bigcup_{i=1}^{n-1} \mathsf{vars}(F_i)\right)$. Again, the desired result holds by the induction hypothesis. □

Lemma C.1.2. *Let q be a query in* sjfBCQ. *Let θ be a valuation over a subset* *of* $\mathsf{vars}(q)$. *Let* $F \in q$ *such that for every variable* $v \in \mathsf{dom}(\theta)$, $F \overset{q}{\not\rightsquigarrow} v$. *Then, for every variable* $v \in \mathsf{vars}(q) \setminus \mathsf{dom}(\theta)$, $F \overset{q}{\rightsquigarrow} v$ *if and only if* $\theta(F) \overset{\theta(q)}{\rightsquigarrow} v$.

Proof. $\boxed{\Longrightarrow}$ Let $v \in \mathsf{vars}(q) \setminus \mathsf{dom}(\theta)$ such that $F \overset{q}{\rightsquigarrow} v$. Since $F \overset{q}{\rightsquigarrow} v$, there is a sequence of variables $(w_1, \dots, w_n)$ such that $w_1 \in \mathsf{notKey}(F)$, $w_n = v$, every two adjacent variables appear together in some atom of q, and for each $j \in \{1, \dots, n\}$, we have $F \overset{q}{\rightsquigarrow} w_j$, which implies $w_j \notin F^{+,q}$ and $w_j \notin \mathsf{dom}(\theta)$. If for every $j \in \{1, \dots, n\}$, $w_j \notin \theta(F)^{+,\theta(q)}$, then it follows that $\theta(F) \overset{\theta(q)}{\rightsquigarrow} v$. Assume, for the sake of contradiction, that there is a $j \in \{1, \dots, n\}$ such that $w_j \in \theta(F)^{+,\theta(q)}$. It follows that, $\mathcal{K}(\theta(q) \setminus \{\theta(F)\}) \models \mathsf{Key}(\theta(F)) \rightarrow w_j$. Consequently, $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F) \cup \mathsf{dom}(\theta) \rightarrow w_j$. Since $F \overset{q}{\rightsquigarrow} w_j$, it follows by Lemma C.1.1 that $F \overset{q}{\rightsquigarrow} w$ for some variable $w \in \mathsf{Key}(F) \cup \mathsf{dom}(\theta)$, a contradiction. $\boxed{\Longleftarrow}$ Easy. □

C.2. Proof of Lemma 6.1.1

Proof of Lemma 6.1.1. Assume that θ is an n-∀embedding of q in **db** relative to $(q, \prec_1)$. We first show that the desired result holds for a topological sort $(q, \prec_2)$ obtained by swapping two adjacent atoms, say F_k and F_{k+1}. To this end, let

$$\begin{aligned}(q, \prec_2) &= (F_1, \dots, F_{k-1}, F_k, F_{k+1}, F_{k+2}, \dots, F_n),\\ &\qquad\qquad\qquad\quad \nwarrow\!\!\!\!\nearrow \\ (q, \prec_1) &= (F_1, \dots, F_{k-1}, F_{k+1}, F_k, F_{k+2}, \dots, F_n).\end{aligned}$$

We have $F_k \overset{q}{\not\rightsquigarrow} F_{k+1}$ and $F_{k+1} \overset{q}{\not\rightsquigarrow} F_k$.

Let θ' be the restriction of θ to $\bigcup_{i=1}^{k-1} \mathsf{vars}(F_i)$. Let $p = \{\theta'(F_k), \theta'(F_{k+1}), \dots, \theta'(F_n)\}$, and for $i \in \{1, 2, \dots, n-k+1\}$, let $G_i = \theta'(F_{i+k-1})$. Let

$$\begin{aligned}(p, \prec'_2) &= (G_1, G_2, G_3, \dots, G_{n-k+1}),\\ (p, \prec'_1) &= (G_2, G_1, G_3, \dots, G_{n-k+1}),\end{aligned}$$

where the topological sorts $\prec'_2$ and $\prec'_1$ are inherited from $\prec_2$ and $\prec_1$. It is known that these are indeed topological sorts of p's attack graph. Informally, p is obtained from q by first applying the partial valuation θ' on q and then omitting the facts $\theta'(F_1)$, $\theta'(F_2)$, …, $\theta'(F_{k-1})$. We have $G_1 \overset{p}{\not\rightsquigarrow} G_2$ and $G_2 \overset{p}{\not\rightsquigarrow} G_1$. Let μ be the restriction of θ to $\mathsf{vars}(p)$. It can be verified from our definitions that

$$\mu \text{ is a } (n-k+1)\text{-}\forall\text{embedding of } p \text{ in } \mathbf{db} \text{ relative to } (p, \prec'_1). \tag{C.1}$$

It suffices now to show that μ is a $(n-k+1)$-$\forall$embedding of p in $\mathbf{db}$ relative to $(p, \prec_2')$.

For $i \in \{1,2\}$, let $G_i = R_i(\underline{\vec{x}_i}, \vec{y}_i)$, in which $\vec{x}_i\vec{y}_i$ needs not be constant-free, and let $\mu(G_i) = R_i(\underline{\vec{a}_i}, \vec{b}_i)$. By (C.1),

$$\mathbf{db} \models_{\mathsf{cqa}} p_{[\vec{x}_2 \mapsto \vec{a}_2]} \tag{C.2}$$

$$\mathbf{db} \models_{\mathsf{cqa}} (p \setminus \{G_2\})_{[\vec{x}_2\vec{y}_2\vec{x}_1 \mapsto \vec{a}_2\vec{b}_2\vec{a}_1]} \tag{C.3}$$

To show the desired result for one swap, it suffices to show $\mathbf{db} \models_{\mathsf{cqa}} p_{[\vec{x}_1 \mapsto \vec{a}_1]}$ and $\mathbf{db} \models_{\mathsf{cqa}} (p \setminus \{G_1\})_{[\vec{x}_1\vec{y}_1\vec{x}_2 \mapsto \vec{a}_1\vec{b}_1\vec{a}_2]}$. In what follows, a fact is said to be *relevant* for a conjunctive query in a database instance if there is an embedding that maps a query atom to the fact.

Proof that $\mathbf{db} \models_{\mathsf{cqa}} p_{[\vec{x}_1 \mapsto \vec{a}_1]}$. Let $\mathbf{r}$ be an arbitrary repair of $\mathbf{db}$. Let $\vec{b}_2'$ be the (unique) sequence of constants, of length $|\vec{y}_2|$, such that $R_2(\underline{\vec{a}_2}, \vec{b}_2') \in \mathbf{r}$. From (C.2), it follows $\mathbf{r} \models p_{[\vec{x}_2 \mapsto \vec{a}_2]}$, hence $R_2(\underline{\vec{a}_2}, \vec{b}_2')$ is relevant for p in $\mathbf{r}$. From (C.3), it follows

$$\mathbf{r} \models (p \setminus \{G_2\})_{[\vec{x}_2\vec{y}_2\vec{x}_1 \mapsto \vec{a}_2\vec{b}_2\vec{a}_1]}.$$

Consequently,

$$\left(\mathbf{r} \setminus \{R_2(\underline{\vec{a}_2}, \vec{b}_2')\}\right) \cup \{R_2(\underline{\vec{a}_2}, \vec{b}_2)\} \models p_{[\vec{x}_2\vec{y}_2\vec{x}_1 \mapsto \vec{a}_2\vec{b}_2\vec{a}_1]}.$$

It follows

$$\left(\mathbf{r} \setminus \{R_2(\underline{\vec{a}_2}, \vec{b}_2')\}\right) \cup \{R_2(\underline{\vec{a}_2}, \vec{b}_2)\} \models p_{[\vec{x}_1 \mapsto \vec{a}_1]}.$$

Since $G_2 \overset{p}{\not\rightsquigarrow} G_1$, it follows by (Koutris & Wijsen, 2017, Lemma B.1) that

$$\mathbf{r} \models p_{[\vec{x}_1 \mapsto \vec{a}_1]}. \tag{C.4}$$

Proof that $\mathbf{db} \models_{\mathsf{cqa}} (p \setminus \{G_1\})_{[\vec{x}_1\vec{y}_1\vec{x}_2 \mapsto \vec{a}_1\vec{b}_1\vec{a}_2]}$. Let $\mathbf{r}$ be an arbitrary repair of $\mathbf{db}$. Let $\vec{b}_1'$ be the (unique) sequence of constants, of length $|\vec{y}_1|$, such that $R_1(\underline{\vec{a}_1}, \vec{b}_1') \in \mathbf{r}$. From (C.4), it follows that $R_1(\underline{\vec{a}_1}, \vec{b}_1')$ is relevant for p in $\mathbf{r}$. By (C.2),

$$\mathbf{r} \models p_{[\vec{x}_2 \mapsto \vec{a}_2]}.$$

Since $G_1 \overset{p}{\not\rightsquigarrow} G_2$, it follows by (Koutris & Wijsen, 2017, Lemma B.1) that

$$\left(\mathbf{r} \setminus \{R_1(\underline{\vec{a}_1}, \vec{b}_1')\}\right) \cup \{R_1(\underline{\vec{a}_1}, \vec{b}_1)\} \models p_{[\vec{x}_2 \mapsto \vec{a}_2]}.$$

Consequently,

$$\mathbf{r} \models (p \setminus \{G_1\})_{[\vec{x}_1\vec{y}_1\vec{x}_2 \mapsto \vec{a}_1\vec{b}_1\vec{a}_2]}.$$

This concludes the proof for one swap.

To conclude the proof of Lemma 6.1.1, it suffices to observe that every topological sort can be obtained from $(q, \prec_1)$ by zero, one, or more swaps, and that any swap results in an n-∀embedding. □

C.3. Proof of Lemma 6.1.2

Proof of Lemma 6.1.2. We define a formula $\varphi_q(\vec{u})$ such that for every database instance **db**, $\mathbf{db} \models \varphi_q(\vec{c})$ if and only if the valuation θ over $\mathsf{vars}(\vec{u})$ such that $\theta(\vec{u}) = \vec{c}$ is an n-∀embedding in **db**.

Let $p(\vec{x})$ be a conjunctive query with free variables $\vec{x}$. A *consistent first-order rewriting of* $p(\vec{x})$ is a first-order formula $\omega(\vec{x})$ such that for every database instance **db**, for every sequence $\vec{c}$ of constants, of length $|\vec{x}|$, we have $\mathbf{db} \models_{\mathsf{cqa}} p(\vec{c})$ if and only if $\mathbf{db} \models \omega(\vec{c})$. The following problem has been solved in Koutris & Wijsen (2017): given a self-join-free conjunctive query $p(\vec{x})$, decide whether $p(\vec{x})$ has a first-order rewriting, and if affirmative, construct such a a first-order rewriting.

Let q be a query in sjfBCQ with an acyclic attack graph. Assume that q's body is $F_1 \wedge F_2 \wedge \cdots \wedge F_n$, where the atoms are listed in a topological sort of the attack graph. For $j \in \{0, 1, 2, \ldots, n\}$, we inductively define a formula $\psi_j(\vec{u}_j)$ expressing that $\vec{u}_j$ is a j-∀embedding. The basis of the induction is $\psi_0() = \mathsf{true}$. For the induction step, $j \to j+1$, the formula $\psi_{j+1}(\vec{u}_{j+1})$ reads as follows:

$$\psi_{j+1}(\overbrace{\vec{u}_j, \vec{x}_{j+1}, \vec{y}_{j+1}}^{\vec{u}_{j+1}}) := \psi_j(\vec{u}_j) \wedge \omega_{j+1}(\vec{u}_j, \vec{x}_{j+1}) \wedge F_{j+1},$$

where $\omega_{j+1}(\vec{u}_j, \vec{x}_{j+1})$ is a consistent first-order rewriting of

$$\exists \vec{y}_{j+1} \exists \vec{x}_{j+2} \exists \vec{y}_{j+2} \cdots \exists \vec{x}_n \exists \vec{y}_n \left(F_{j+1} \wedge F_{j+2} \wedge \cdots \wedge F_n\right).$$

It follows from (Koutris & Wijsen, 2017) that $\omega_{j+1}(\vec{u}_j, \vec{x}_{j+1})$ exists, and can be constructed in linear time in the length $|q|$ of q. Then, our desired formula $\varphi_q(\vec{u}_n)$ is equal to $\psi_n(\vec{u}_n)$:

$$\varphi_q(\vec{u}_n) := \psi_n(\vec{u}_n).$$

It remains to show the quadratic upper bound on the construction of $\psi_n(\vec{u}_n)$. If T_{j+1} denotes the time for constructing ψ_{j+1}, then $T_0 = 1$ and for some constant c, we have $T_{j+1} \leq T_j + c \cdot |q|$. It follows that $T_n \leq n \cdot c \cdot |q|$. Since $n \leq |q|$, T_n is quadratic in the length of q. $\square$

We illustrated the construction with an example.

Example C.3.1. Let $q(x, y, z) = \exists x \exists y \exists z \left(R(\underline{x}, y) \wedge S(\underline{y, z}, c)\right)$.

$$\begin{aligned} \psi_1(x, y) &= \mathsf{true} \wedge \omega_1(x) \wedge R(\underline{x}, y), \\ \psi_2(x, y, z) &= \psi_1(x, y) \wedge \omega_2(x, z) \wedge S(\underline{y, z}, c), \end{aligned}$$

where

$$\begin{aligned} \omega_1(x) = {} & \exists y R(\underline{x}, y) \wedge \\ & \forall y \left(R(\underline{x}, y) \rightarrow \exists z \left(S(\underline{y, z}, c) \wedge \forall u \left(S(\underline{y, z}, u) \rightarrow u = c\right)\right)\right), \\ \omega_2(x, z) = {} & S(\underline{y, z}, c) \wedge \forall u \left(S(\underline{y, z}, u) \rightarrow u = c\right). \end{aligned}$$

Putting all together, with some simplifications:

$$\begin{aligned} \psi_2(x, y, z) = {} & R(\underline{x}, y) \wedge S(\underline{y, z}, c) \\ & \wedge \forall y \left(R(\underline{x}, y) \rightarrow \exists z \left(S(\underline{y, z}, c) \wedge \forall u \left(S(\underline{y, z}, u) \rightarrow u = c\right)\right)\right) \\ & \wedge \forall u \left(S(\underline{y, z}, u) \rightarrow u = c\right). \end{aligned}$$

$\triangleleft$

C.4. Proof of Lemma 6.1.3

We introduce some helping constructs and lemmas.

Definition C.4.1. Let q be a query in sjfBCQ. Let $\mathbf{db}$ be a consistent database instance. Let V be a non-empty subset of $\mathsf{vars}(q)$. We define $\mathsf{Reify}(q, \mathbf{db}, V)$ as the $\subseteq$-minimal set of valuations over V that contains θ if θ can be extended to a valuation μ over $\mathsf{vars}(q)$ such that $(\mathbf{db}, \mu) \models q$. $\square$

The following helping lemmas extend (Koutris & Wijsen, 2017, Lemma 4.4).

Lemma C.4.1. *Let q be a query in sjfBCQ. Let $\mathbf{db}$ be a database instance. Let V be a non-empty subset of $\mathsf{vars}(q)$ such that no variable of V is attacked in q. Then, there is a repair $\mathbf{r}$ of $\mathbf{db}$ such that*

$$\mathsf{Reify}(q, \mathbf{r}, V) = \bigcap_{\mathbf{s} \in \mathsf{rset}(\mathbf{db})} \mathsf{Reify}(q, \mathbf{s}, V).$$

The following lemma extends Lemma C.4.1 by allowing V to contain variables v that are attacked in q, provided that v is only attacked by atoms whose corresponding relation in $\mathbf{db}$ is consistent.

Lemma C.4.2. *Let q be a query in* sjfBCQ. *Let $\mathbf{db}$ be a database instance. Let V be a non-empty subset of* $\mathsf{vars}(q)$ *such that for every $v \in V$, for every atom $F = R(\underline{\vec{x}}, \vec{y})$ in q, if $F \overset{q}{\rightsquigarrow} v$, then the R-relation of $\mathbf{db}$ is consistent. Then, there is a repair $\mathbf{r}$ of $\mathbf{db}$ such that*

$$\mathsf{Reify}(q, \mathbf{r}, V) = \bigcap_{\mathbf{s} \in \mathsf{rset}(\mathbf{db})} \mathsf{Reify}(q, \mathbf{s}, V).$$

Proof. Let M be a set containing a fresh atom $R'(\underline{\vec{x}}, \vec{y})$ for every atom $R(\underline{\vec{x}}, \vec{y})$ in q such that the R-relation of $\mathbf{db}$ is consistent. Let $q' = q \cup M$. Let $\mathbf{db}'$ be the smallest database instance that includes $\mathbf{db}$ and includes $\{R'(\underline{\vec{a}}, \vec{c}) \mid R(\underline{\vec{a}}, \vec{c}) \in \mathbf{db}\}$ for every atom $R'(\underline{\vec{x}}, \vec{y})$ in M. It follows from the hypotheses of the lemma that no variable of V is attacked in q'. For every repair $\mathbf{r}$ of $\mathbf{db}$, let $f(\mathbf{r})$ the smallest database instance that includes $\mathbf{r}$ and includes $\{R'(\underline{\vec{a}}, \vec{b}) \mid R(\underline{\vec{a}}, \vec{b}) \in \mathbf{db}\}$ for every atom $R'(\underline{\vec{x}}, \vec{y})$ in M. One can easily verify that every n-embedding of q in $\mathbf{r}$ is also an n-embedding of q' in $f(\mathbf{r})$, and vice versa. Thus, for every repair $\mathbf{r}$ of $\mathbf{db}$, $\mathsf{Reify}(q, \mathbf{r}, V) = \mathsf{Reify}(q', f(\mathbf{r}), V)$. Since $\mathbf{db}' \setminus \mathbf{db}$ is consistent by construction, we have that $f : \mathsf{rset}(\mathbf{db}) \to \mathsf{rset}(\mathbf{db}')$ is a bijective mapping. Since no variable of V is attacked in q', it follows by Lemma C.4.1 that there is a repair $\mathbf{r}'$ of $\mathbf{db}'$ such that

$$\mathsf{Reify}(q', \mathbf{r}', V) = \bigcap_{\mathbf{s}' \in \mathsf{rset}(\mathbf{db}')} \mathsf{Reify}(q', \mathbf{s}', V).$$

From what precedes, it follows

$$\begin{aligned}\mathsf{Reify}(q, f^{-1}(\mathbf{r}'), V) &= \bigcap_{\mathbf{s}' \in \mathsf{rset}(\mathbf{db}')} \mathsf{Reify}(q, f^{-1}(\mathbf{s}'), V) \\ &= \bigcap_{\mathbf{s} \in \mathsf{rset}(\mathbf{db})} \mathsf{Reify}(q, \mathbf{s}, V)\end{aligned}$$

Then, $f^{-1}(\mathbf{r}')$ is a repair of $\mathbf{db}$ that proves the lemma. $\square$

The proof of Lemma 6.1.3 follows.

Proof of Lemma 6.1.3. The desired result is obvious if there is a repair $\mathbf{r}^*$ such that $\mathbf{r}^* \not\models \exists \vec{u}\,(q(\vec{u}))$. Assume from here on that for every repair $\mathbf{s}$, $\mathbf{s} \models \exists \vec{u}\,(q(\vec{u}))$.

For every $i \in \{1, \ldots, n\}$, let R_i be the relation name of F_i. Let $\mathbf{r}$ be a repair of $\mathbf{db}$. We show that for every $i \in \{0, 1, 2, \ldots, n\}$, there is a repair $\mathbf{s}_i$ of $\mathbf{db}$ such that:

(a) every i-embedding of q in $\mathbf{s}_i$ is an i-embedding of q in $\mathbf{r}$; and

(b) every i-embedding of q in $\mathbf{s}_i$ is an i-$\forall$embedding of q in $\mathbf{db}$.

The construction runs by induction on increasing i. For the induction basis, $i = 0$, let $\mathbf{s}_0$ be an arbitrary repair of $\mathbf{db}$. From our assumption that every repair of $\mathbf{db}$ satisfies $\exists\vec{u}\,(q(\vec{u}))$, it follows that the empty set is both a 0-embedding in $\mathbf{r}$ and a 0-$\forall$embedding in $\mathbf{db}$, as desired.

For the induction step, $i \to i+1$, the induction hypothesis is that there is a repair $\mathbf{s}_i$ of $\mathbf{db}$ that satisfies conditions (a) and (b). Let $\mathbf{db}^{(i)}$ be the smallest database instance that includes $\mathbf{s}_i$ and includes, for every $j \in \{i+1, i+2, \ldots, n\}$, the R_j-relation of $\mathbf{db}$. For every $m \in \{1, 2, \ldots, n\}$, if $F_m \overset{q}{\rightsquigarrow} v$ with $v \in \bigcup_{j=1}^{i} \mathsf{vars}(F_j)$, then $m \leq i$, and hence, by construction, the R_m-relation of $\mathbf{db}^{(i)}$ is consistent. Then, by Lemma C.4.2, there is a repair $\mathbf{s}^*$ of $\mathbf{db}^{(i)}$ such that

$$\mathsf{Reify}(q, \mathbf{s}^*, \vec{u}_i\vec{x}_{i+1}) = \bigcap_{\mathbf{t} \in \mathsf{rset}(\mathbf{db}^{(i)})} \mathsf{Reify}(q, \mathbf{t}, \vec{u}_i\vec{x}_{i+1}). \tag{C.5}$$

Claim 4. For every $\nu \in \mathsf{Reify}(q, \mathbf{s}^*, \vec{u}_i\vec{x}_{i+1})$, we have

$$(\mathbf{db}, \nu) \models_{\mathsf{cqa}} \{F_{i+1}, F_{i+2}, \ldots, F_n\}.$$

Proof. Assume for the sake of a contradiction that there is a repair $\mathbf{t}$ of $\mathbf{db}$ such that

$$(\mathbf{t}, \nu) \not\models \{F_{i+1}, F_{i+2}, \ldots, F_n\}. \tag{C.6}$$

Let $\mathbf{t}'$ be the smallest database instance such that

- for every $j \in \{1, 2, \ldots, i\}$, $\mathbf{t}'$ contains all R_j-facts of $\mathbf{s}_i$; and
- for every $j \in \{i+1, i+2, \ldots, n\}$, $\mathbf{t}'$ contains all R_j-facts of $\mathbf{t}$.

Clearly, $\mathbf{t}'$ is a repair of $\mathbf{db}^{(i)}$. Thus, by (C.5), $\nu \in \mathsf{Reify}(q, \mathbf{t}', \vec{u}_i\vec{x}_{i+1})$. Hence, $(\mathbf{t}', \nu) \models \{F_1, F_2, \ldots, F_n\}$, and thus $(\mathbf{t}', \nu) \models \{F_{i+1}, F_{i+2}, \ldots, F_n\}$. Since $\mathbf{t}$ and $\mathbf{t}'$ contain the same R_j-facts for every $j \in \{i+1, i+2, \ldots, n\}$, it follows $(\mathbf{t}, \nu) \models \{F_{i+1}, F_{i+2}, \ldots, F_n\}$, which contradicts (C.6). This concludes the proof of Claim 4. $\square$

Let $\theta_1, \theta_2, \ldots, \theta_g$ enumerate all n-embeddings of q in $\mathbf{s}^*$. For every $k \in$

$\{1, 2, \ldots, g\}$, let A_k be the (unique) R_{i+1}-fact of $\mathbf{r}$ that is key-equal to $\theta_k(F_{i+1})$. Let

$$\mathbf{s}_{i+1} := \left(\mathbf{s}^* \setminus \{\theta_k(F_{i+1})\}_{k=1}^{g}\right) \cup \{A_1, A_2, \ldots, A_g\}.$$

Clearly, $\mathbf{s}_{i+1}$ is a repair of $\mathbf{db}^{(i)}$.

Claim 5. $\mathsf{Reify}(q, \mathbf{s}^*, \vec{u}_i\vec{x}_{i+1}) = \mathsf{Reify}(q, \mathbf{s}_{i+1}, \vec{u}_i\vec{x}_{i+1})$.

Proof. The $\subseteq$-inclusion is straightforward. The proof of the $\supseteq$-inclusion is analogous to (Koutris & Wijsen, 2017, Lemma B.1.), by remarking that each fact in $\{\theta_k(F_{i+1})\}_{k=1}^{g}$ is relevant for q in $\mathbf{s}^*$. $\square$

We are now ready to show that conditions (a) and (b) hold true for $i+1$. To this end, let η be an arbitrary n-embedding of q in $\mathbf{s}_{i+1}$. Consequently, $\eta\restriction_{\vec{u}_i\vec{x}_{i+1}} \in \mathsf{Reify}(q, \mathbf{s}_{i+1}, \vec{u}_i\vec{x}_{i+1})$. By Claims 4 and 5, it follows

$$(\mathbf{db}, \eta\restriction_{\vec{u}_i\vec{x}_{i+1}}) \models_{\mathsf{cqa}} \{F_{i+1}, F_{i+2}, \ldots, F_n\}. \tag{C.7}$$

Since for every $j \in \{1, 2, \ldots, i\}$, the set of R_i-facts of $\mathbf{s}_{i+1}$ is identical to that of $\mathbf{s}_i$ (and identical to that of $\mathbf{db}^{(i)}$), it follows that $\eta\restriction_{\vec{u}_i}$ is an i-embedding of q in $\mathbf{s}_i$. By the induction hypothesis,

(A) $\eta\restriction_{\vec{u}_i}$ an i-embedding of q in $\mathbf{r}$; and

(B) $\eta\restriction_{\vec{u}_i}$ is an i-$\forall$embedding of q in $\mathbf{db}$.

From (B) and (C.7), it follows that $\eta\restriction_{\vec{u}_{i+1}}$ is an $(i+1)$-$\forall$embedding of q in $\mathbf{db}$.

Claim 6. There is $k \in \{1, 2, \ldots, g\}$ such that $\eta(F_{i+1}) = A_k$.

Proof. Assume for the sake of a contradiction that $\eta(F_{i+1}) \notin \{A_1, A_2, \ldots, A_g\}$. Then, by the construction of $\mathbf{s}_{i+1}$ from $\mathbf{s}^*$, it follows that η is an n-embedding of q in s^*. Hence, we can assume $k \in \{1, 2, \ldots, g\}$ such that $\eta = \theta_k$, and thus $\eta(F_{i+1}) = \theta_k(F_{i+1})$. From $\eta(q) \subseteq \mathbf{s}_{i+1}$, it follows $\eta(F_{i+1}) \in \mathbf{s}_{i+1}$. Hence, $\theta_k(F_{i+1}) \in \mathbf{s}_{i+1}$, which can happen only if $\theta_k(F_{i+1}) = A_k$, contradicting $\eta(F_{i+1}) \notin \{A_1, A_2, \ldots, A_g\}$. $\square$

From (A) and Claim 6, it follows that $\eta\restriction_{\vec{u}_{i+1}}$ is an $(i+1)$-embedding of q in $\mathbf{r}$. This concludes the induction step. The proof of Lemma 6.1.3 is concluded by letting $\mathbf{r}^* = \mathbf{s}_n$. $\square$

C.5. Equivalence of Superfrugal and n-Minimal Repairs

Let $\exists\vec{u}\,(q(\vec{u}))$ be a Boolean conjunctive query where $q(\vec{u})$ is quantifier-free. The notion of i-minimality is defined relative to a duplicate-free sequence $(F_1, F_2, \dots, F_n)$ containing exactly the atoms of $q(\vec{u})$. Let $i \in \{1, 2, \dots, n\}$, and $X := \bigcup_{j=1}^{i} \mathsf{vars}(F_i)$. Let $\mathbf{db}$ be a database instance. A repair $\mathbf{r}$ of $\mathbf{db}$ is *i-minimal* (Figueira et al., 2023) if there is no repair $\mathbf{s}$ of $\mathbf{db}$ such that

1. for every valuation θ over X, if $(\mathbf{s}, \theta) \models q(\vec{u})$, then $(\mathbf{r}, \theta) \models q(\vec{u})$; and
2. for some valuation μ over X, $(\mathbf{r}, \mu) \models q(\vec{u})$ and $(\mathbf{s}, \mu) \not\models q(\vec{u})$.

An i-minimal repair is called $\preceq_q^X$-frugal in (Koutris & Wijsen, 2017).

Lemma C.5.1. *Let $q := \exists\vec{u}\,(q(\vec{u}))$ be a query in* sjfBCQ *with an acyclic attack graph, where $q(\vec{u})$ is quantifier-free. Let $(F_1, F_2, \dots, F_n)$ be a topological sort of q's attack graph. Let* $\mathbf{db}$ *be a database instance. Then,*

(i) every superfrugal repair of $\mathbf{db}$ *is n-minimal; and*

(ii) every n-minimal repair of $\mathbf{db}$ *is superfrugal.*

Proof. $\boxed{\text{Proof of (i)}}$ Let $\mathbf{r}$ be a superfrugal repair of $\mathbf{db}$. Assume for the sake of a contradiction that $\mathbf{r}$ is not n-minimal. Then, there is a repair $\mathbf{s}$ of $\mathbf{db}$ such that

(a) for every valuation θ over $\vec{u}$, if $(\mathbf{s}, \theta) \models q(\vec{u})$, then $(\mathbf{r}, \theta) \models q(\vec{u})$; and

(b) for some valuation μ over $\vec{u}$, $(\mathbf{r}, \mu) \models q(\vec{u})$ and $(\mathbf{s}, \mu) \not\models q(\vec{u})$.

Since $(\mathbf{r}, \mu) \models q(\vec{u})$ and $\mathbf{r}$ is superfrugal, it follows that μ is an n-$\forall$embedding of q in $\mathbf{db}$. Let $\ell \in \{1, 2, \dots, n\}$ be the largest index such that for $X := \bigcup_{j=1}^{\ell-1} \mathsf{vars}(F_j)$, we have $(\mathbf{s}, \mu\restriction_X) \models q(\vec{u})$. Let $Y := X \cup \mathsf{Key}(F_\ell)$. By the definition of $\forall$embedding, $(\mathbf{s}, \mu\restriction_Y) \models q(\vec{u})$. Therefore, $\mu\restriction_Y$ can be extended to a valuation ν over $\vec{u}$ such that $(\mathbf{s}, \nu) \models q(\vec{u})$. Since $\mu(F_\ell)$ and $\nu(F_\ell)$ are distinct and key-equal, it follows $(\mathbf{r}, \nu) \not\models q(\vec{u})$, contradicting (a).

$\boxed{\text{Proof of (ii)}}$ Let $\mathbf{r}$ be an n-minimal repair of $\mathbf{db}$. By Lemma 6.1.3, there exists a superfrugal repair $\mathbf{r}^*$ of $\mathbf{db}$ such that for every valuation θ over $\vec{u}$, if $(\mathbf{r}^*, \theta) \models q(\vec{u})$, then $(\mathbf{r}, \theta) \models q(\vec{u})$. Since $\mathbf{r}$ is n-minimal, there is no valuation μ over $\vec{u}$ such that $(\mathbf{r}, \mu) \models q(\vec{u})$ and $(\mathbf{r}^*, \mu) \not\models q(\vec{u})$. It follows that every embedding of $q(\vec{u})$ in $\mathbf{r}$ is also an embedding of $q(\vec{u})$ in the superfrugal repair $\mathbf{r}^*$, and hence is a $\forall$embedding of q in $\mathbf{db}$. This concludes the proof. $\square$

C.6. Proof of Lemma 6.2.1

Proof of Lemma 6.2.1. $\boxed{\text{Proof of (1).}}$ Let $\mathbf{r}$ be a superfrugal repair of $\mathbf{db}$. Assume for the sake of a contradiction that $\mathbf{Emb}_q(\mathbf{r})$ is not an MCS of $\forall\mathbf{Emb}_q(\mathbf{db})$. By definition of a superfrugal repair, every element in $\mathbf{Emb}_q(\mathbf{r})$ is a $\forall$embedding of q in $\mathbf{db}$, and therefore $\mathbf{Emb}_q(\mathbf{r}) \subseteq \forall\mathbf{Emb}_q(\mathbf{db})$. Since $\mathbf{Emb}_q(\mathbf{r}) \models \mathcal{K}(q)$ but $\mathbf{Emb}_q(\mathbf{r})$ is not an MCS of $\forall\mathbf{Emb}_q(\mathbf{db})$, there is an MCS N^* of $\forall\mathbf{Emb}_q(\mathbf{db})$ such that $\mathbf{Emb}_q(\mathbf{r}) \subsetneq N^* \subseteq M$. Then, we can assume a valuation θ in $N^* \setminus \mathbf{Emb}_q(\mathbf{r})$. Let $(F_1, \dots, F_n)$ be a topological sort of q's attack graph. Let ℓ be the greatest integer in $\{0, \dots, n\}$ such that $\theta(\{F_1, \dots, F_\ell\}) \subseteq \mathbf{r}$. If $\ell = n$, then $\theta \in \mathbf{Emb}_q(\mathbf{r})$, a contradiction. Assume next that $\ell < n$. Let γ be the restriction of θ to $\mathsf{vars}(\{F_1, \dots, F_\ell\}) \cup \mathsf{Key}(F_{\ell+1})$. By the definition of $\forall$embedding, we have that $(\mathbf{r}, \gamma) \models \{F_{\ell+1}, \dots, F_n\}$. Thus, there is an extension θ' of γ such that $\theta' \in \mathbf{Emb}_q(\mathbf{r})$ and $\{\theta, \theta'\} \not\models \mathsf{Key}(F_{\ell+1}) \rightarrow \mathsf{vars}(F_{\ell+1})$. Since $\mathbf{Emb}_q(\mathbf{r}) \subsetneq N^*$, we also have that $\theta' \in N^*$. From $\theta, \theta' \in N^*$ and $\{\theta, \theta'\} \not\models \mathsf{Key}(F_{\ell+1}) \rightarrow \mathsf{vars}(F_{\ell+1})$, it follows $N^* \not\models \mathcal{K}(q)$, contradicting that N^* is an MCS. We conclude by contradiction that (1) holds true.

$\boxed{\text{Proof of (2).}}$ Let N be an MCS of $\forall\mathbf{Emb}_q(\mathbf{db})$. Since $N \models \mathcal{K}(q)$, there is a repair $\mathbf{r}_0$ of $\mathbf{db}$ such that $N \subseteq \mathbf{Emb}_q(\mathbf{r}_0)$. Conversely, since N is an MCS, it contains every $\forall$embedding θ of q in $\mathbf{db}$ such that $\theta(q) \subseteq \mathbf{r}_0$. By Lemma 6.1.3, there is a superfrugal repair $\mathbf{r}$ of $\mathbf{db}$ such that every embedding of q in $\mathbf{r}$ is also an embedding of q in $\mathbf{r}_0$. By the definition of superfrugal repair, every element in $\mathbf{Emb}_q(\mathbf{r})$ is a $\forall$embedding of q in $\mathbf{db}$. Consequently, $\mathbf{Emb}_q(\mathbf{r}) \subseteq N$. By (1), $\mathbf{Emb}_q(\mathbf{r})$ is an MCS of $\forall\mathbf{Emb}_q(\mathbf{db})$, and therefore $\mathbf{Emb}_q(\mathbf{r})$ cannot be a strict subset of N. Consequently, $\mathbf{Emb}_q(\mathbf{r}) = N$. So $\mathbf{r}$ is a repair of $\mathbf{db}$ such that the set of embeddings of q in $\mathbf{r}$ is exactly N, which concludes the proof. Note that the repair that proves (2) is superfrugal. $\square$

C.7. Proof of *Decomposition Lemma* (Lemma 6.3.1)

Using the notation $\vec{u}_\ell$ defined in Section 6.1, the following lemma states that for every $(\ell+1)$-$\forall$key-embedding γ, if an MCS of $\mathsf{Ext}(\gamma \restriction_{\vec{u}_\ell})$ is restricted to those valuations that include γ, the result is an MCS of $\mathsf{Ext}(\gamma)$.

Lemma C.7.1. *Let q, $(F_1, \dots, F_n)$, ℓ, and* $\mathbf{db}$ *be as in Definition C.8.1. Let θ be an ℓ-$\forall$embedding of q in* $\mathbf{db}$*, and let N be an MCS of* $\mathsf{Ext}(\theta)$*. Let γ be an $(\ell{+}1)$-$\forall$key-embedding of q in* $\mathbf{db}$ *that extends θ. Let N_γ be the subset of N containing all (and only) $\forall$embeddings that extend γ. Then, N_γ is an MCS of* $\mathsf{Ext}(\gamma)$*.*

Proof. Assume, for the sake of a contradiction, that N_γ is not an MCS of $\mathsf{Ext}(\gamma)$. Since $N_\gamma \models \mathcal{K}(q)$, there is an MCS N^* of $\mathsf{Ext}(\gamma)$ such that $N_\gamma \subsetneq N^*$. We can assume $\mu \in N^* \setminus N_\gamma$. From $\mu \notin N_\gamma$, it follows $\mu \notin N$. Let k be the greatest integer in $\{\ell, \dots, n\}$ such that for some $\mu' \in N$, we have $\mu(\{F_1, \dots, F_k\}) = \mu'(\{F_1, \dots, F_k\})$. It is easily verified that such k exists. If $k = n$, then $\mu \in N$, a contradiction. Assume from here on that $k < n$. Let μ_k be the restriction of μ to $\mathsf{vars}(\{F_1, \dots, F_k\}) \cup \mathsf{Key}(F_{k+1})$. Using the same reasoning as in the proof of Lemma 6.2.1, there is a repair $\mathbf{r}$ of $\mathbf{db}$ such that N contains all and only embeddings of q in $\mathbf{r}$ that extend θ. Thus, we have that $\mu_k(\{F_1, \dots, F_k\}) \subseteq \mathbf{r}$, and by the definition of $\forall$embedding, $(\mathbf{r}, \mu_k) \models \{F_{k+1}, \dots, F_n\}$. Thus, N contains some extension μ' of μ_k such that $\{\mu, \mu'\} \not\models \mathsf{Key}(F_{k+1}) \rightarrow \mathsf{vars}(F_{k+1})$. Since $\mu' \in N$, it follows that $\mu' \in N_\gamma$, and thus $\mu' \in N^*$. From $\mu, \mu' \in N^*$ and $\{\mu, \mu'\} \not\models \mathsf{Key}(F_{k+1}) \rightarrow \mathsf{vars}(F_{k+1})$, it follows $N^* \not\models \mathcal{K}(q)$, contradicting that N^* is an MCS of $\mathsf{Ext}(\gamma_i)$. We conclude by contradiction that N_γ is an MCS of $\mathsf{Ext}(\gamma)$. □

We can now proceed with the proof of Lemma 6.3.1.

Proof of Lemma 6.3.1. Assume that we have $\bigcup_{i=1}^{k} N_i \models \mathcal{K}(q)$, which implies that $\bigcup_{i=1}^{k} N_i$ is an MCS of $\mathsf{Ext}(\theta)$. Let N be an $\mathcal{F}_{\mathsf{AGG}}$-minimal MCS of $\mathsf{Ext}(\theta)$, i.e,

$$m = \mathcal{F}_{\mathsf{AGG}}\left(\{\{\mu(r) \mid \mu \in N\}\}\right).$$

For ease of notation, we define

$$\widehat{m} := \mathcal{F}_{\mathsf{AGG}}\left(\{\{\mu(r) \mid \mu \in \bigcup_{i=1}^{k} N_i\}\}\right). \tag{C.8}$$

For each $i \in \{1, \dots, k\}$, let N_i^- be the subset of N containing all (and only) $\forall$embeddings that extend γ_i, and define $v_i^- := \mathcal{F}_{\mathsf{AGG}}\left(\{\{\mu(r) \mid \mu \in N_i^-\}\}\right)$. Note that $\{N_1^-, N_2^-, \dots, N_k^-\}$ is a partition of N. Since $\mathcal{F}_{\mathsf{AGG}}$ is associative, it follows that

$$m = \mathcal{F}_{\mathsf{AGG}}\left(\{\{v_1^-, v_2^-, \dots, v_k^-\}\}\right), \tag{C.9}$$

and

$$\widehat{m} = \mathcal{F}_{\mathsf{AGG}}\left(\{\{v_1, v_2, \dots, v_k\}\}\right). \tag{C.10}$$

By Lemma C.7.1, for every $i \in \{1, \dots, k\}$, N_i^- is an MCS of $\mathsf{Ext}(\gamma_i)$. By the definition of $\mathcal{F}_{\mathsf{AGG}}$-minimal MCS, we have that for every $i \in \{1, \dots, k\}$, $v_i \leq v_i^-$. Thus, since $\mathcal{F}_{\mathsf{AGG}}$ is monotone, we have that

$$\mathcal{F}_{\mathsf{AGG}}\left(\{\{v_1, v_2, \dots, v_k\}\}\right) \leq \mathcal{F}_{\mathsf{AGG}}\left(\{\{v_1^-, v_2^-, \dots, v_k^-\}\}\right). \tag{C.11}$$

By (C.9), (C.10) and (C.11), it follows that

$$\widehat{m} \leq m. \tag{C.12}$$

Since m is the $\mathcal{F}_{\mathsf{AGG}}$-minimal value for θ, it follows that

$$\widehat{m} = m. \tag{C.13}$$

By (C.10) and (C.13), we can conclude that

$$m = \mathcal{F}_{\mathsf{AGG}}\left(\{\{v_1, v_2, \ldots, v_k\}\}\right).$$

This concludes the proof. □

C.8. Proof of *Consistent Extension Lemma* (Lemma 6.3.2)

We first introduce some helping definitions and lemmas.

Definition C.8.1 (Branch)**.** Let q be a query in sjfBCQ with an acyclic attack graph. Let $(F_1, \ldots, F_n)$ be a topological sort of q's attack graph. Let $\ell \in \{0, 1, \ldots, n-1\}$. Let $\mathbf{db}$ be a database instance. Let θ be an ℓ-$\forall$embedding. A valuation γ that extends θ is called a *branch (of θ)* if both the following hold:

(A) each variable in $\mathsf{dom}(\gamma) \setminus \mathsf{dom}(\theta)$ is unattacked in $\theta(\{F_{\ell+1}, \ldots, F_n\})$; and

(B) γ is included in some n-$\forall$embedding of q in $\mathbf{db}$.

Such a branch is called an $(\ell+1)$-*$\forall$key-embedding* if $\mathsf{dom}(\gamma) = \mathsf{dom}(\theta) \cup \mathsf{Key}(F_{\ell+1})$. □

Lemma C.8.1. *Let q be a query in sjfBCQ such that the attack graph of q is acyclic. Let $(F_1, \ldots, F_n)$ be a topological sort of q's attack graph. Let $\mathbf{db}$ be a database instance. Let θ_1 and θ_2 be two ℓ-$\forall$embeddings of q in $\mathbf{db}$ such that $\{\theta_1, \theta_2\} \models \mathcal{K}(\{F_1, \ldots, F_\ell\})$. For $i \in \{1, 2\}$, let γ_i be a branch of θ_i such that $\mathsf{dom}(\gamma_1) = \mathsf{dom}(\gamma_2)$. Then, for every $X, Y \subseteq \mathsf{dom}(\gamma_1)$, if $\mathcal{K}(q) \models X \rightarrow Y$, then $\{\gamma_1, \gamma_2\} \models X \rightarrow Y$.*

Proof. From (Koutris & Wijsen, 2017, Lemma 4.4), it follows that for $i \in \{1, 2\}$, $(\mathbf{db}, \gamma_i) \models_{\mathsf{cqa}} \{F_{\ell+1}, \ldots, F_n\}$. Since $\{\theta_1, \theta_2\} \models \mathcal{K}(\{F_1, \ldots, F_\ell\})$, we can assume a repair $\mathbf{r}$ of $\mathbf{db}$ such that for $i \in \{1, 2\}$,

$$\theta_i(\{F_1, \ldots, F_\ell\}) \subseteq \mathbf{r}. \tag{C.14}$$

Let $i \in \{1, 2\}$. Since $(\mathbf{db}, \gamma_i) \models_{\mathsf{cqa}} \{F_{\ell+1}, \dots, F_n\}$, we have that $(\mathbf{r}, \gamma_i) \models \{F_{\ell+1}, \dots, F_n\}$. Hence, there exists a valuation γ_i^+ over $\mathsf{vars}(q)$ that extends γ_i such that $\gamma_i^+(\{F_{\ell+1}, \dots, F_n\}) \subseteq \mathbf{r}$. From (C.14), it follows $\gamma_i^+(\{F_1, \dots, F_n\}) \subseteq \mathbf{r}$. Consequently, $\{\gamma_1^+, \gamma_2^+\} \models \mathcal{K}(q)$. Let $X, Y \subseteq \mathsf{dom}(\gamma_1)$ such that $\mathcal{K}(q) \models X \to Y$. Since $\{\gamma_1^+, \gamma_2^+\} \models \mathcal{K}(q)$, it follows $\{\gamma_1^+, \gamma_2^+\} \models X \to Y$, hence $\{\gamma_1, \gamma_2\} \models X \to Y$. □

Lemma C.8.2. *Let q, $(F_1, \dots, F_n)$, ℓ, and* $\mathbf{db}$ *be as in Definition C.8.1. Let γ be a branch of some ℓ-∀embedding of q in* $\mathbf{db}$ *such that $\mathsf{Key}(F_{\ell+1}) \subseteq \mathsf{dom}(\gamma)$, and define $q_\gamma := \gamma(\{F_\ell, \dots, F_n\})$. Let $q_1 \subseteq \{F_\ell, \dots, F_n\}$ such that:*

(a) $F_\ell \in q_1$;

(b) the atoms of $\gamma(q_1)$ form a maximal weakly connected component of the attack graph of q_γ; and

(c) for every $v \in \mathsf{vars}(q_\gamma)$, if $\mathcal{K}(q) \models \mathsf{Key}(F_\ell) \to v$, then v is attacked in q_γ.

Let $q_2 := \{F_\ell, \dots, F_n\} \setminus q_1$. Then, $\mathsf{vars}(\gamma(q_1)) \cap \mathsf{vars}(\gamma(q_2)) = \emptyset$.

Proof. For every $k \in \{\ell + 1, \dots, n\}$, we have $F_k \overset{q}{\not\rightsquigarrow} F_\ell$. Consequently, by Lemma C.1.2, $\gamma(F_\ell)$ is unattacked in q_γ. Assume $v \in \mathsf{vars}(\gamma(q_1))$. We show $v \notin \mathsf{vars}(\gamma(q_2))$. Since $v \in \mathsf{vars}(\gamma(q_1))$ and acyclic attack graphs are known to be transitive, the variable v must occur in an atom of $\gamma(q_1)$ that is either unattacked in the attack graph of $\gamma(q_1)$ or attacked by another atom that itself is unattacked. Hence, there is an atom $F \in q_1$ such that $\gamma(F)$ is unattacked in q_γ, and one of the following holds:

(a) $v \in \mathsf{vars}(\gamma(F))$, hence $\mathcal{K}(q) \models \mathsf{Key}(F) \to v$; or

(b) $v \in \mathsf{vars}(\gamma(G))$ for some atom $G \in q_1$ such that $\gamma(F) \overset{q_\gamma}{\rightsquigarrow} \gamma(G)$.

Claim 7. $\mathcal{K}(q) \models \mathsf{Key}(F_\ell) \to \mathsf{Key}(F)$.

Proof. The desired result is obvious if $F = F_\ell$. Assume $F \neq F_\ell$ from here on.

Since the atoms $\gamma(F)$ and $\gamma(F_\ell)$ are unattacked in q_γ and belong to the same weakly connected component of the attack graph, and since acyclic attack graphs are known to be transitive, there is a sequence of atoms $(G_1, \dots, G_k)$ ($k \geq 2$) in q_1 such that $G_1 = F_\ell$, $G_k = F$, and such that for every $i \in \{1, \dots, k - 1\}$,

- $\gamma(G_i) \overset{q_\gamma}{\not\rightsquigarrow} \gamma(G_{i+1})$, $\gamma(G_{i+1}) \overset{q_\gamma}{\not\rightsquigarrow} \gamma(G_i)$; and

- there is an atom $H_i \in q_1$ such that $\gamma(G_i) \overset{q_\gamma}{\rightsquigarrow} \gamma(H_i)$ and $\gamma(G_{i+1}) \overset{q_\gamma}{\rightsquigarrow} \gamma(H_i)$.

By Lemma C.1.2, for every $i \in \{1, \ldots, k-1\}$, $G_i \overset{q}{\not\rightsquigarrow} G_{i+1}$, $G_{i+1} \overset{q}{\not\rightsquigarrow} G_i$, $G_i \overset{q}{\rightsquigarrow} H_i$, and $G_{i+1} \overset{q}{\rightsquigarrow} H_i$. By repeated application of Lemma A.0.3 and logical implication of functional dependencies, we obtain $\mathcal{K}(q) \models \mathsf{Key}(F_\ell) \rightarrow \mathsf{Key}(F)$. This concludes the proof of Claim 7. $\square$

Assume for the sake of a contradiction that $v \in \mathsf{vars}(\gamma(q_2))$. We show

$$\mathcal{K}(q) \models \mathsf{Key}(F_\ell) \rightarrow v, \tag{C.15}$$

which immediately follows from Claim 7 if (a) holds true. Assume next that (b) holds true. Since the atoms of $\gamma(q_1)$ form a maximal weakly connected component of the attack graph of q_γ, it follows that v is not attacked in q_γ. By Lemma C.1.2, it follows $F \overset{q}{\not\rightsquigarrow} v$, which in turn implies $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F) \rightarrow v$. By Claim 7, we obtain (C.15). Then, by the hypothesis of the lemma, v is attacked in q_γ, a contradiction. This concludes the proof. $\square$

Let $q = q_1 \uplus q_2$ be as in Lemma C.8.2. The following lemma implies that an $\mathcal{F}_{\mathtt{AGG}}$-minimal MCS relative to q can be obtained by taking the cross product of two MCS, N_1 and N_2, which are calculated relative to q_1 and q_2, respectively.

Lemma C.8.3. *Let $\exists\vec{u}(q(\vec{u}))$ be a query in* sjfBCQ *with an acyclic attack graph, topologically sorted $(F_1, \ldots, F_n)$. Let $g()$ be the* AGGR[FOL] *query defined as $g() := \mathtt{AGG}(r) \leftarrow q(\vec{u})$, where $\mathcal{F}_{\mathtt{AGG}}$ is monotone and associative. Under the same hypotheses as Lemma C.8.2, for every $i \in \{1, 2\}$, let M_i be the set of $\forall$embeddings of $\gamma(q_i)$ in* **db***, and let N_i be an MCS of M_i such that*

- *N_i is $\mathcal{F}_{\mathtt{AGG}}$-minimal if r is a variable in $\gamma(q_i)$; and*
- *N_i is $\mathcal{F}_{\mathtt{COUNT}}$-minimal otherwise (i.e., if r is a variable not in $\gamma(q_i)$ or a constant).*

Then, $\{\gamma \cdot \theta_1 \cdot \theta_2 \mid \theta_1 \in N_1 \text{ and } \theta_2 \in N_2\}$ is an $\mathcal{F}_{\mathtt{AGG}}$-minimal MCS of $\mathsf{Ext}(\gamma)$.

Proof. Since $\mathsf{vars}(\gamma(q_1)) \cap \mathsf{vars}(\gamma(q_2)) = \emptyset$ by Lemma C.8.2, it follows that for every $\theta_1 \in N_1$ and $\theta_2 \in N_2$, we have that $\gamma \cdot \theta_1 \cdot \theta_2$ is a valid embedding of q in **db**. Let $N = \{\gamma \cdot \theta_1 \cdot \theta_2 \mid \theta_1 \in N_1 \text{ and } \theta_2 \in N_2\}$. Clearly, N is a subset of $\mathsf{Ext}(\gamma)$. Moreover, since N_1 and N_2 are MCSs of M_1 and M_2 respectively, it follows that N is an MCS of $\mathsf{Ext}(\gamma)$.

Let N^* be an $\mathcal{F}_{\mathsf{AGG}}$-minimal MCS of $\mathsf{Ext}(\gamma)$, and for $i \in \{1, 2\}$, define $N_i^* := \{\theta \restriction_{\mathsf{vars}(\gamma(q_i))} \mid \theta \in N^*\}$. Then, $N^* = \{\gamma \cdot \theta_1 \cdot \theta_2 \mid \theta_1 \in N_1^* \text{ and } \theta_2 \in N_2^*\}$. Since $\mathsf{vars}(\gamma(q_1)) \cap \mathsf{vars}(\gamma(q_2)) = \emptyset$ by Lemma C.8.2, and since N^* is an MCS of of $\mathsf{Ext}(\gamma)$, it is easily seen that each N_i^* is an MCS of M_i ($i \in \{1, 2\}$). By the definition of $\mathcal{F}_{\mathsf{AGG}}$-minimal MCS, it follows that

$$\mathcal{F}_{\mathsf{AGG}}\left(\{\{\mu(r) \mid \mu \in N^*\}\}\right) \leq \mathcal{F}_{\mathsf{AGG}}\left(\{\{\mu(r) \mid \mu \in N\}\}\right). \tag{C.16}$$

To show that N is $\mathcal{F}_{\mathsf{AGG}}$-minimal, we distinguish two cases. To ease the notation, for $i \in \{1, 2\}$, we define $c_i := |N_i|$ and $c_i^* := |N_i^*|$.

Case that $r \in \mathsf{dom}(\gamma)$ or r is a constant. Then, $\gamma(r)$ is a constant. Indeed, since γ is the identity on constants, if r is a constant, then $r = \gamma(r)$. We have that for every $i \in \{1, 2\}$, N_i is an $\mathcal{F}_{\mathsf{COUNT}}$-minimal MCS of M_i. With the construct $i\#t$ as defined in Definition 8.1.1, it follows

$$\mathcal{F}_{\mathsf{AGG}}\left(\{\{\mu(r) \mid \mu \in N\}\}\right) = \mathcal{F}_{\mathsf{AGG}}\left(\{\{(c_1 * c_2)\#\gamma(r)\}\}\right), \tag{C.17}$$

and

$$\mathcal{F}_{\mathsf{AGG}}\left(\{\{\mu(r) \mid \mu \in N^*\}\}\right) = \mathcal{F}_{\mathsf{AGG}}\left(\{\{(c_1^* * c_2^*)\#\gamma(r)\}\}\right). \tag{C.18}$$

By the definition of $\mathcal{F}_{\mathsf{COUNT}}$-minimal MCS, for every $i \in \{1, 2\}$, $c_i \leq c_i^*$. Since $\mathcal{F}_{\mathsf{AGG}}$ is monotone,

$$\mathcal{F}_{\mathsf{AGG}}\left(\{\{(c_1 * c_2)\#\gamma(r)\}\}\right) \leq \mathcal{F}_{\mathsf{AGG}}\left(\{\{(c_1^* * c_2^*)\#\gamma(r)\}\}\right). \tag{C.19}$$

From (C.17), (C.18), and (C.19), it follows

$$\mathcal{F}_{\mathsf{AGG}}\left(\{\{\mu(r) \mid \mu \in N\}\}\right) \leq \mathcal{F}_{\mathsf{AGG}}\left(\{\{\mu(r) \mid \mu \in N^*\}\}\right). \tag{C.20}$$

From (C.16) and (C.20), it follows

$$\mathcal{F}_{\mathsf{AGG}}\left(\{\{\mu(r) \mid \mu \in N\}\}\right) = \mathcal{F}_{\mathsf{AGG}}\left(\{\{\mu(r) \mid \mu \in N^*\}\}\right). \tag{C.21}$$

It follows that N is an $\mathcal{F}_{\mathsf{AGG}}$-minimal MCS of $\mathsf{Ext}(\gamma)$.

Case that r is a variable not in $\mathsf{dom}(\gamma)$. Assume, without loss of generality, that $r \in \mathsf{vars}(\gamma(q_1))$. By Lemma C.8.2, we have that $r \notin \mathsf{vars}(\gamma(q_2))$. Thus, N_1 is an $\mathcal{F}_{\mathsf{AGG}}$-minimal MCS of M_1, and N_2 is an $\mathcal{F}_{\mathsf{COUNT}}$-minimal MCS of M_2. It follows that

$$\mathcal{F}_{\mathsf{AGG}}\left(\{\{\mu(r) \mid \mu \in N\}\}\right) = \mathcal{F}_{\mathsf{AGG}}\left(\{\{c_2\#\mu(r) \mid \mu \in N_1\}\}\right), \tag{C.22}$$

and

$$\mathcal{F}_{\texttt{AGG}}\left(\{\{\mu(r) \mid \mu \in N^*\}\}\right) = \mathcal{F}_{\texttt{AGG}}\left(\{\{c_2^* \# \mu(r) \mid \mu \in N_1^*\}\}\right). \tag{C.23}$$

Since $\mathcal{F}_{\texttt{AGG}}$ is associative,

$$\mathcal{F}_{\texttt{AGG}}\left(\{\{\mu(r) \mid \mu \in N\}\}\right) = \mathcal{F}_{\texttt{AGG}}\left(\{\{c_2 \# \mathcal{F}_{\texttt{AGG}}\left(\{\{\mu(r) \mid \mu \in N_1\}\}\right)\}\}\right), \tag{C.24}$$

and

$$\mathcal{F}_{\texttt{AGG}}\left(\{\{\mu(r) \mid \mu \in N^*\}\}\right) = \mathcal{F}_{\texttt{AGG}}\left(\{\{c_2^* \# \mathcal{F}_{\texttt{AGG}}\left(\{\{\mu(r) \mid \mu \in N_1^*\}\}\right)\}\}\right). \tag{C.25}$$

By the definition of $\mathcal{F}_{\texttt{COUNT}}$-minimal MCS, $c_2 \leq c_2^*$. By definition of an $\mathcal{F}_{\texttt{AGG}}$-minimal MCS, $\mathcal{F}_{\texttt{AGG}}\left(\{\{\mu(r) \mid \mu \in N_1\}\}\right) \leq \mathcal{F}_{\texttt{AGG}}\left(\{\{\mu(r) \mid \mu \in N_1^*\}\}\right)$. Since $\mathcal{F}_{\texttt{AGG}}$ is monotone, we obtain that

$$\begin{array}{c} \mathcal{F}_{\texttt{AGG}}\left(\{\{c_2 \# \mathcal{F}_{\texttt{AGG}}\left(\{\{\mu(r) \mid \mu \in N_1\}\}\right)\}\}\right) \\ \rotatebox{90}{$\geq$} \\ \mathcal{F}_{\texttt{AGG}}\left(\{\{c_2^* \# \mathcal{F}_{\texttt{AGG}}\left(\{\{\mu(r) \mid \mu \in N_1^*\}\}\right)\}\}\right). \end{array} \tag{C.26}$$

From (C.24), (C.25) and (C.26), it follows

$$\mathcal{F}_{\texttt{AGG}}\left(\{\{\mu(r) \mid \mu \in N\}\}\right) \leq \mathcal{F}_{\texttt{AGG}}\left(\{\{\mu(r) \mid \mu \in N^*\}\}\right). \tag{C.27}$$

From (C.16) and (C.27), it follows

$$\mathcal{F}_{\texttt{AGG}}\left(\{\{\mu(r) \mid \mu \in N\}\}\right) = \mathcal{F}_{\texttt{AGG}}\left(\{\{\mu(r) \mid \mu \in N^*\}\}\right). \tag{C.28}$$

It follows that N is an $\mathcal{F}_{\texttt{AGG}}$-minimal MCS of $\mathsf{Ext}(\gamma)$.

The proof is now concluded. □

We can now proceed with the proof of Lemma 6.3.2.

Proof of Lemma 6.3.2. For readability, we show the lemma for $k = 2$. The proof can easily be generalized for $k > 2$.

In the first part of the proof, we show that for $i \in \{1, 2\}$, there is an ℓ-$\forall$embedding θ_i extending γ_i such that:

- every $\mathcal{F}_{\texttt{AGG}}$-minimal MCS of $\mathsf{Ext}(\theta_i)$ is also an $\mathcal{F}_{\texttt{AGG}}$-minimal MCS of $\mathsf{Ext}(\gamma_i)$; and
- $\{\theta_1, \theta_2\} \models \mathcal{K}(\{F_1, \ldots, F_\ell\})$.

Note that for $i \in \{1,2\}$, $\mathsf{dom}(\gamma_i) = \left(\bigcup_{j=1}^{\ell-1} \mathsf{vars}(F_j)\right) \cup \mathsf{Key}(F_\ell)$, and $\mathsf{dom}(\theta_i) = \mathsf{dom}(\gamma_i) \cup \mathsf{notKey}(F_\ell)$. We distinguish two cases:

Case that γ_1 and γ_2 disagree on a variable of $\mathsf{Key}(F_\ell)$. Let $i \in \{1,2\}$. Let N_i^* be an $\mathcal{F}_{\mathtt{AGG}}$-minimal MCS of $\mathsf{Ext}(\gamma_i)$. Let θ_i be the (unique) ℓ-$\forall$embedding such that every embedding in N_i^* extends θ_i. Clearly, every $\mathcal{F}_{\mathtt{AGG}}$-minimal MCS of $\mathsf{Ext}(\theta_i)$ is also an MCS of $\mathsf{Ext}(\gamma_i)$. Since γ_1 and γ_2 disagree on some variable of $\mathsf{Key}(F_\ell)$, it follows that $\{\theta_1, \theta_2\} \models \mathcal{K}(\{F_1, \dots, F_\ell\})$.

Case that γ_1 and γ_2 agree on $\mathsf{Key}(F_\ell)$. Let $i \in \{1,2\}$. Let $\vec{w}$ be a shortest sequence containing each variable $v \in \mathsf{vars}(\gamma_i(\{F_\ell, \dots, F_n\}))$ such that v is not attacked in $\gamma_i(\{F_\ell, \dots, F_n\})$ and $\mathcal{K}(q) \models \mathsf{Key}(F_\ell) \rightarrow v$. Note that $\vec{w}$ is the same for $i = 1$ and $i = 2$. By Lemma C.8.1, there is a sequence of constants $\vec{a}$, of length $|\vec{w}|$, such that for every $j \in \{1,2\}$, for every $\mu \in \mathsf{Ext}(\gamma_j)$, $\mu(\vec{w}) = \vec{a}$. Let γ_i^+ be the extension of γ_i to $\mathsf{dom}(\gamma_i) \cup \mathsf{vars}(\vec{w})$ such that $\gamma_i^+(\vec{w}) = \vec{a}$. It is clear that

- $\gamma_1^+(\vec{w}) = \gamma_2^+(\vec{w})$; and
- since $\mathsf{Ext}(\gamma_i) = \mathsf{Ext}(\gamma_i^+)$, every $\mathcal{F}_{\mathtt{AGG}}$-minimal MCS of $\mathsf{Ext}(\gamma_i^+)$ is also an $\mathcal{F}_{\mathtt{AGG}}$-minimal MCS of $\mathsf{Ext}(\gamma_i)$.

Claim 8. For every variable $v \in \mathsf{vars}(\gamma_i^+(\{F_\ell, \dots, F_n\}))$, if $\mathcal{K}(q) \models \mathsf{Key}(F_\ell) \rightarrow v$, then v is attacked in $\gamma_i^+(\{F_\ell, \dots, F_n\})$.

Proof. Straightforward from Lemma C.1.2 and the construction of $\vec{w}$. □

Let $q_\alpha \subseteq \{F_\ell, \dots, F_n\}$ such that $F_\ell \in q_\alpha$, and the atoms of $\gamma_i^+(q_\alpha)$ form a maximal weakly connected component of the attack graph of $\gamma_i^+(\{F_\ell, \dots, F_n\})$. Let $q_\beta = \{F_\ell, \dots, F_n\} \setminus q_\alpha$.

Claim 9. For every $v \in \mathsf{vars}(q_\alpha) \cap \mathsf{dom}(\gamma_1^+)$, we have $\gamma_1^+(v) = \gamma_2^+(v)$.

Proof. Let $v \in \mathsf{vars}(q_\alpha) \cap \mathsf{dom}(\gamma_1^+)$. We need to show $\gamma_1^+(v) = \gamma_2^+(v)$. This is obvious if $v \in \mathsf{vars}(\vec{w})$. Assume for the sake of a contradiction that $v \notin \mathsf{vars}(\vec{w})$. Then, $v \in \mathsf{dom}(\gamma_1)$. It follows $v \in \mathsf{vars}(\{F_1, \dots, F_{\ell-1}\})$, and therefore

$$\text{for every } k \in \{\ell, \dots, n\}, F_k \overset{q}{\not\rightsquigarrow} v. \tag{C.29}$$

Define $q_{\gamma^+} := \gamma_1^+(\{F_\ell, \dots, F_n\})$. It is worth noting that the choice of using γ_1^+ instead of γ_2^+ in the definition of q_{γ^+} is unimportant, as the resulting attacks will be the same regardless of the choice. Since $v \in \mathsf{vars}(q_\alpha)$ and since acyclic attack graphs are transitive, there is an atom $F \in q_\alpha$ such that $\gamma_1^+(F)$ is unattacked in q_{γ^+}, and one of the following holds:

- $v \in \mathsf{vars}(\gamma_1^+(F))$, hence $v \in \mathsf{vars}(F)$; or
- there is $G \in q_\alpha$ such that $v \in \mathsf{vars}(\gamma_1^+(G))$ and $\gamma_1^+(F) \overset{q_{\gamma^+}}{\rightsquigarrow} \gamma_1^+(G)$, hence $v \in \mathsf{vars}(G)$ and, by Lemma C.1.2, $F \overset{q}{\rightsquigarrow} G$.

Since $F \overset{q}{\not\rightsquigarrow} v$ by (C.29), it is correct to conclude $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F) \rightarrow v$. By the same reasoning as in the proof of Claim 7, we obtain $\mathcal{K}(q\}) \models \mathsf{Key}(F_\ell) \rightarrow \mathsf{Key}(F)$. Consequently, $\mathcal{K}(q) \models \mathsf{Key}(F_\ell) \rightarrow v$. From (C.29) and Lemma C.1.2, it follows that v is not attacked in $\gamma_i(\{F_\ell, \dots, F_n\})$. Then, by our definition of $\vec{w}$, we have that $\vec{w}$ contains v, a contradiction. This concludes the proof of Claim 9. □

Claim 9 implies $\gamma_1^+(q_\alpha) = \gamma_2^+(q_\alpha)$. Let $N^{(\alpha)}$ be an MCS of $\mathsf{Ext}(\gamma_1^+ \mid \gamma_1^+(q_\alpha), \mathbf{db})$ such that $N^{(\alpha)}$ is $\mathcal{F}_{\texttt{AGG}}$-minimal if $r \in \mathsf{vars}(\gamma_1^+(q_\alpha))$, and $\mathcal{F}_{\texttt{COUNT}}$-minimal otherwise. For $j \in \{1, 2\}$, let $N_j^{(\beta)}$ be an MCS of $\mathsf{Ext}(\gamma_j^+ \mid \gamma_j^+(q_\beta), \mathbf{db})$ such that $N_j^{(\beta)}$ is $\mathcal{F}_{\texttt{AGG}}$-minimal if $r \in \mathsf{vars}(\gamma_j^+(q_\beta))$, and $\mathcal{F}_{\texttt{COUNT}}$-minimal otherwise. By Claim 8 and Lemma C.8.3, the set $N_i^* := \{\gamma_i^+ \cdot \delta \cdot \epsilon \mid \delta \in N^{(\alpha)} \text{ and } \epsilon \in N_i^{(\beta)}\}$ is an $\mathcal{F}_{\texttt{AGG}}$-minimal MCS of $\mathsf{Ext}(\gamma_i^+)$. Let θ_i be the (unique) ℓ-$\forall$embedding such that every valuation in N_i^* extends θ_i. Clearly, every $\mathcal{F}_{\texttt{AGG}}$-minimal MCS of $\mathsf{Ext}(\theta_i)$ is also an $\mathcal{F}_{\texttt{AGG}}$-minimal MCS of $\mathsf{Ext}(\gamma_i^+)$, and therefore also an MCS of $\mathsf{Ext}(\gamma_i)$. Since $N^{(\alpha)}$ is the same for $i = 1$ and $i = 2$, and since $F_\ell \in q_\alpha$, it follows that $\theta_1(F_\ell) = \theta_2(F_\ell)$. Consequently, $\{\theta_1, \theta_2\} \models \mathcal{K}(\{F_1, \dots, F_\ell\})$.

So it is correct to conclude that θ_1, θ_2 with the desired properties exist, which concludes the first part of the proof.

We are now ready to prove that the lemma holds for every choice of ℓ in the statement of the lemma. The proof is by induction on decreasing ℓ. It is straightforward to see that the lemma holds true when $\ell = n$. We next show that the lemma holds true when $\ell = g$, assuming that it holds true when $\ell = g + 1$. Let $i \in \{1, 2\}$. Let $\gamma_1^{+,(i)}, \dots, \gamma_{k_i}^{+,(i)}$ enumerate all extensions of θ_i that are $(g+1)$-$\forall$key-embeddings of q in $\mathbf{db}$, where θ_i is the g-$\forall$embedding whose existence was proved in the first part of the proof. By the induction hypothesis, for every $j \in \{1, \dots, k_i\}$, there is an $\mathcal{F}_{\texttt{AGG}}$-minimal MCS $N_j^{+,(i)}$ of

$\mathsf{Ext}(\gamma_j^{+,(i)})$ such that

$$\overbrace{\left(\bigcup_{j=1}^{k_1} N_j^{+,(1)}\right)}^{N_1} \cup \overbrace{\left(\bigcup_{j=1}^{k_2} N_j^{+,(2)}\right)}^{N_2} \models \mathcal{K}(q),$$

in which we define N_1 and N_2 as shown above. It remains to show that N_i is an $\mathcal{F}_{\texttt{AGG}}$-minimal MCS of $\mathsf{Ext}(\gamma_i)$. Clearly, it suffices to to show that N_i is an $\mathcal{F}_{\texttt{AGG}}$-minimal MCS of $\mathsf{Ext}(\theta_i)$. To this end, let m_i denote the $\mathcal{F}_{\texttt{AGG}}$-minimal value for θ_i in **db**, as defined in Definition 5.0.2. Since $N_i \models \mathcal{K}(q)$, it follows from Lemma 6.3.1 that

$$m_i = \mathcal{F}_{\texttt{AGG}}\left(\{\{v_1, v_2, \ldots, v_{k_i}\}\}\right),$$

where for every $j \in \{1, \ldots, k_i\}$, $v_j := \mathcal{F}_{\texttt{AGG}}\left(\{\{\mu(r) \mid \mu \in N_j^{+,(i)}\}\}\right)$. From this, it is correct to conclude that N_i is an $\mathcal{F}_{\texttt{AGG}}$-minimal MCS of $\mathsf{Ext}(\gamma_i)$, which concludes the proof of Lemma 6.3.2. $\square$

APPENDIX **D**

Proofs of Chapter 7

D.1. Proof of Lemma 7.2.3

We first need some helping lemmas on functional dependencies. In the following lemma, Σ needs not be irreducible. The following definition serves as a natural analogue to the notion of sequential proof introduced in Chapter 2.

Definition D.1.1. Let Σ be a set of functional dependencies. Let $X \to \{y\}$ be a functional dependency. A *sequential proof of* $\Sigma \models X \to \{y\}$ is a (possibly empty) sequence $(X_1 \to Y_1, X_2 \to Y_2, \dots, X_\ell \to Y_\ell)$ of FDs in Σ such that $y \in X \cup Y_\ell$ and for every $i \in \{1, 2, \dots, \ell\}$, $X_i \subseteq X \cup \left(\bigcup_{j=1}^{i-1} (X_{j-1} \cup Y_{j-1})\right)$. $\square$

Lemma D.1.1. *Let S be a set of attributes. Let Σ be a set of functional dependencies on S such that for every $X \to Y$ in Σ, we have $|Y| = 1$ and $Y \nsubseteq X$. Let $<$ be a linear order on S such that for every $X \to \{y\}$ in Σ, for every $v \in X$, we have $v < y$. For all $U \subseteq S$ and $w \in S$, if $\Sigma \models U \to \{w\}$, then $\Sigma \models \{v \in U \mid v < w\} \to y$.*

Proof. Assume $\Sigma \models U \to \{w\}$. There is a shortest sequence

$$(X_1 \to y_1, X_2 \to y_2, \dots, X_\ell \to y_\ell) \tag{D.1}$$

of FDs in Σ such that for $j \in \{1, 2, \dots, \ell\}$

- $X_j \subseteq U \cup \left(\bigcup_{i=1}^{j-1} X_i \cup \{y_i\}\right)$;
- $y_\ell = w$.

Let $U_{<w} = \{v \in U \mid v < w\}$. We show by induction on decreasing $i = \ell, \ell - 1, \dots, 2, 1$ that $X_i \subseteq U_{<w}$. For $i = \ell$, the desired result follows from $y_\ell = w$ and the hypothesis of the lemma. For the induction step, $i+1 \to i$, the induction hypothesis is that $X_{i+1} \cup X_{i+2} \cup \dots \cup X_\ell \subseteq U_{<w}$. Since $y_i \in X_{i+1} \cup X_{i+2} \cup \dots \cup X_\ell$ (because the sequence (D.1) is shortest possible), it follows $y_i < w$. By the hypothesis of the lemma, we conclude $X_i \subseteq U_{<w}$. This concludes the proof of Lemma D.1.1. $\square$

Definition D.1.2. Let Σ be a set of functional dependencies (FDs) that is irreducible (a.k.a. a minimal cover). The *carrier set of* Σ is the set of attributes that occur in the FDs of Σ. A linear sort $<$ of the carrier set of Σ is said to be *compliant with* Σ if for every $X \to y$ in Σ, for every $v \in X$, we have $v < y$. We write $\mathcal{H}(\Sigma)$ for the directed graph whose vertex-set is Σ; there is a directed edge from $X \to y$ to $U \to v$ if $y \in U$. $\square$

Lemma D.1.2 (Existence of compliant order)**.** *Let Σ be a set of functional dependencies that is irreducible. Let S be the carrier set of Σ. If $\mathcal{H}(\Sigma)$ is acyclic, then there exists a linear order of S that is compliant with Σ.*

Proof. Assume $\mathcal{H}(q)$ is acyclic. Let $(X_1 \to y_1, X_2 \to y_2, \dots, X_n \to y_n)$ be a topological sort of $\mathcal{H}(\Sigma)$. Define $f : S \to \{1, 2, \dots, n, n+1\}$ such that for every $x \in S$,

- if $x \notin \bigcup_{i=1}^{n} X_i$, then $f(x) = n + 1$;
- otherwise $f(x)$ is the smallest $i \in \{1, 2, \dots, n\}$ such that $x \in X_i$.

Let $\prec$ be a linear order of S such that for all $v, w \in S$ such that $v \neq w$, if $f(v) < f(w)$, then $v \prec w$. Here, $<$ denotes the natural order on $\mathbb{N}$. Clearly, such a linear order $\prec$ exists.

We show that $\prec$ is compliant with Σ. To this end, let $j \in \{1, 2, \dots, n\}$. Let $v \in X_j$. It suffices to show $v \prec y_j$. Since Σ is irreducible, $y_j \notin X_j$, hence $f(y_j) \neq j$. It is easily verified that $f(v) \leq j$ and $j < f(y_j)$. Specifically, if $f(y_j) < j$, then $\mathcal{H}(\Sigma)$ would have a directed edge from $X_j \to y_j$ to a functional dependency that precedes $X_j \to y_j$ in the topological sort, a contradiction. Consequently, $f(v) < f(y_j)$, hence $v \prec y_j$. $\square$

The following theorem has already been proven under the restriction that each left-hand side (LHS) is a singleton (Lechtenbörger, 2004).

Theorem D.1.3. *Let Σ be a set of functional dependencies that is irreducible. If $\mathcal{H}(\Sigma)$ is acyclic, then every irreducible set of functional dependencies that is equivalent to Σ, is equal to Σ.*

Proof. Let Σ_1 be an irreducible set of FDs such that $\mathcal{H}(\Sigma_1)$ is acyclic. Let S be the carrier set of Σ_1. Let Σ_2 be an irreducible set of FDs such that $\Sigma_2 \equiv \Sigma_1$. We need to show $\Sigma_1 = \Sigma_2$.

By Lemma D.1.2, we can assume a linear order $<_1$ of S that is compliant with Σ_1. The following claim states that compliance extends to Σ_2.

Claim 10. Let $X \to y$ be an FD in Σ_2. For every $v \in X$, we have $v <_1 y$.

Proof. Assume for the sake of a contradiction that there is $v \in X$ such that $y <_1 v$. We have $\Sigma_1 \models X \to y$. By Lemma D.1.1, $\Sigma_1 \models \{u \in X \mid u < y\} \to y$. Consequently, $\Sigma_1 \models X \setminus \{v\} \to y$, contradicting that Σ_2 is irreducible. This concludes the proof of Claim 10. ☐

We first show that $\mathcal{H}(\Sigma_2)$ is acyclic. Assume for the sake of a contradiction an elementary cycle $(X_0 \to y_0, X_1 \to y_1, \dots, X_{\ell-1} \to y_{\ell-1}, X_0 \to y_0)$ in $\mathcal{H}(\Sigma_2)$. Since the cycle is elementary, we can assume $y_i \neq y_j$ if $i \neq j$. By Claim 10, $y_0 <_1 y_1 <_1 \cdots <_1 y_{\ell-1} <_1 y_0$, a contradiction. We conclude by contradiction that $\mathcal{H}(\Sigma_2)$ is acyclic. By Lemma D.1.2, we can assume an order $<_2$ of S that is compliant with Σ_2.

It remains to show that $\Sigma_1 = \Sigma_2$. Since Σ_2 is irreducible, it suffices to show $\Sigma_1 \subseteq \Sigma_2$. Let $X \to y$ be an FD in Σ_1. We need to show that $X \to y$ is in Σ_2. Since $\Sigma_2 \models \Sigma_1$, there is a shortest sequence $(G_1, G_2, \dots, G_\ell)$ that is a sequential proof of $\Sigma_2 \models X \to y$. Let $G_i = X_i \to y_i$ for $i \in \{1, 2, \dots, \ell\}$, hence $y_\ell = y$. The following claim implies that the desired result obtains if X_ℓ and X are comparable by $\subseteq$.

Claim 11. If $X_\ell \subseteq X$ or $X_\ell \subseteq X$, then $X_\ell = X$, hence $G_\ell = X \to y$ belongs to Σ_2.

Proof. Assume $X_\ell \subseteq X$ or $X_\ell \subseteq X$. Since Σ_1 and Σ_2 are minimal covers containing, respectively, $X \to y$ and $X_\ell \to y$, it follows from $\Sigma_1 \equiv \Sigma_2$ that $X = X_\ell$. ☐

In the sequel, we show that X_ℓ and X are indeed comparable by $\subseteq$. To this end, assume for the sake of a contradiction that X_ℓ and X are not comparable by $\subseteq$. Then necessarily $\ell \geq 2$.

Claim 12. For every $i \in \{1, 2, \dots, \ell - 1\}$, $\Sigma_1 \setminus \{X \to y\} \models X_i \to y_i$.

Proof. Let $i \in \{1, 2, \dots, \ell - 1\}$. From the sequential proof $(G_1, G_2, \dots, G_\ell)$ of $\Sigma_2 \models X \to y$, and by using Claim 10, it is easily verified that $y_i <_1 y$. Since $\Sigma_1 \models \Sigma_2$ and $X_i \to y_i \in \Sigma_2$, it follows $\Sigma_1 \models X_i \to y_i$. Let $\pi = (Z_1 \to w_1, Z_2 \to w_2, \dots, Z_k \to w_k)$ be a shortest sequence that is a sequential proof of $\Sigma_1 \models X_i \to y_i$. Hence $w_k = y_i \neq y$. It suffices to show that for $j \in \{1, 2, \dots, k\}$, we have $Z_j \to w_j \neq X \to y$. Assume for the sake of a contradiction that there is $j \in \{1, 2, \dots, k\}$ such that $Z_j \to w_j = X \to y$. Since $y_i \neq y$, it follows that $X \to y$ is not the last atom in the sequential proof π, hence $j < k$. Since the sequential proof π is the shortest possible, it can be seen that there is $z \in Z_k$ such that $y \leq_1 z$. Since $w_k = y_i$, it follows $z <_1 y_i$. Consequently, $y <_1 y_i$, a contradiction. □

Claim 13. $\Sigma_1 \setminus \{X \to y\} \models X_\ell \to y$.

Proof. Assume for the sake of a contradiction that $\Sigma_1 \setminus \{X \to y\} \not\models X_\ell \to y$. This means that every sequential proof of $\Sigma_1 \models X_\ell \to y$ (and there is at least one such proof) must use $X \to y$. Consequently, since $\Sigma_1 \equiv \Sigma_2$,

$$\Sigma_2 \models X_\ell \to X \setminus X_\ell. \tag{D.2}$$

Let $x_{i_1} <_2 x_{i_2} <_2 \cdots <_2 x_{i_g}$ enumerate all variables in $X \setminus X_\ell$, ordered by $<_2$. Note that since $X \to y$ belongs to the irreducible set Σ_1, it follows $X \subseteq \bigcup_{i=1}^{\ell} X_i$. We know $g \geq 1$ because X and X_ℓ are not comparable by $\subseteq$. By inspecting the sequential proof $(G_1, G_2, \dots, G_\ell)$ of $\Sigma_2 \models X \to y$, it can be verified that there exists $i \in \{1, 2, \dots, \ell - 1\}$ such that $x_{i_g} <_2 y_i$ and $y_i \in X_\ell$. Informally, x_{i_g} is used in the sequential proof to derive some y_j which either occurs in X_ℓ, or is smaller (with respect to $<_2$) than some other y_i occurring in X_ℓ. From (D.2) and Lemma D.1.1, it follows that for every $j \in \{1, 2, \dots, g\}$, $\Sigma_2 \models X_\ell \setminus \{y_i\} \to x_{i_j}$. Informally, we can remove y_i from the LHS because y_i occurs after the RHS in $<_2$. Consequently, $\Sigma_2 \models X_\ell \setminus \{y_i\} \to X \setminus X_\ell$. Moreover, since $y_i \notin X$, we have $\Sigma_2 \models X_\ell \setminus \{y_i\} \to X \cap X_\ell$. Consequently, $\Sigma_2 \models X_\ell \setminus \{y_i\} \to X$.

Since $\Sigma_2 \models X \to y$, we obtain $\Sigma_2 \models X_\ell \setminus \{y_i\} \to y$. Since Σ_2 contains $X_\ell \to y$ with $y_i \in X_\ell$, and Σ_2 logically implies $X_\ell \setminus \{y_i\} \to y$, it follows that Σ_2 is not a minimal cover, a contradiction. This concludes the proof of Claim 13. □

By Claims 12 and 13, it follows $\Sigma_1 \setminus \{X \to y\} \models X \to y$. It follows that Σ_1 is not a minimal cover, which yields the desired contradiction. This concludes the proof of Theorem D.1.3. □

We now provide the proof of Lemma 7.2.3.

Proof of Lemma 7.2.3. Let q_1 and q_2 be kernels of q. Assume that the attack graph of q_1 is acyclic. For $i \in \{1, 2\}$, let Σ_i be the set that contains $\mathsf{Key}(F) \to \mathsf{notKey}(F)$ for every atom of q_i that is not full-key. Since q_1 and q_2 are kernels of q, it follows that Σ_1 and Σ_2 are equivalent irreducible sets of functional dependencies. By Lemma 7.2.1, for all atoms F, G of q that are not full-key, if $\mathcal{H}(\Sigma_1)$ contains a directed edge from $\mathsf{Key}(F) \rightarrow \mathsf{notKey}(F)$ to $\mathsf{Key}(G) \rightarrow \mathsf{notKey}(G)$, then $F \overset{q_1}{\rightsquigarrow} G$. Since the attack graph of q_1 is acyclic, it follows that $\mathcal{H}(\Sigma_1)$ is acyclic. By Theorem D.1.3, $\Sigma_1 = \Sigma_2$.

Assume that F is an atom in q_1 such that $F \overset{q_1}{\rightsquigarrow} x$. Then, F is not full-key and $\mathcal{G}aifman(q_1)$ has a (possibly empty) path from the variable in $\mathsf{notKey}(F)$ to x such that no variable on the path uses a variable in F^{+,q_1}. Since $\Sigma_1 = \Sigma_2$, there is an atom F' in q_2 such that $\mathsf{Key}(F) = \mathsf{Key}(F')$, $\mathsf{notKey}(F) = \mathsf{notKey}(F')$, and $F^{+,q_1} = F'^{+,q_2}$. Since $\mathcal{G}aifman(q_1) = \mathcal{G}aifman(q_2)$, it follows $F' \overset{q_2}{\rightsquigarrow} x$.

Assume for the sake of a contradiction that the attack graph of q_2 contains a cycle. By (Koutris & Wijsen, 2017, Lemma 3.6), q_2 contains two atoms F_1, F_2 such that $F_1 \overset{q_2}{\rightsquigarrow} F_2$ and $F_2 \overset{q_2}{\rightsquigarrow} F_1$. Since $\Sigma_1 = \Sigma_2$, for $i \in \{1, 2\}$, there is an atom F_i' in q_1 such that $\mathsf{Key}(F_i') = \mathsf{Key}(F_i)$ and $\mathsf{notKey}(F_i') = \mathsf{notKey}(F_i)$. From the preceding paragraph, it follows $F_1' \overset{q_1}{\rightsquigarrow} F_2'$ and $F_2' \overset{q_1}{\rightsquigarrow} F_1'$, which contradicts that the attack graph of q_1 is acyclic. □

D.2. Two Ways for Proving Proposition 7.2.4

We give two proofs of Proposition 7.2.4. The first uses arguments from complexity theory, and the second relies solely on syntactic arguments.

Complexity-Theoretic Proof of Proposition 7.2.4. The only place where Proposition 7.2.4 is used is in the proof of Theorem 7.0.2, to handle the case where $\mathcal{F}_{\texttt{AGG}}(\emptyset)$ is undefined. If we restrict ourselves to aggregate operators that are defined over the empty multiset, such as $\mathcal{F}_{\texttt{SUM}}$, then all our results hold without relying on Proposition 7.2.4.

Assume that $\exists\vec{u}(q(\vec{u}))$ is κ-acyclic. Let $g() := \texttt{SUM}(1) \leftarrow q(\vec{u})$. By Theorem 7.0.1, $\mathsf{LUB\text{-}CQA}(g())$ is expressible in $\mathsf{AGGR[FOL]}$. In (Libkin, 2004, Corollary 8.26 and Exercise 8.16), it is established that every query in a logic called $\mathcal{L}_{\text{aggr}}$ is Hanf-local. Since $\mathsf{AGGR[FOL]}$ is included in $\mathcal{L}_{\text{aggr}}$, every query in $\mathsf{AGGR[FOL]}$ is Hanf-local and hence cannot express $\mathsf{2DM}$. Consequently, there exists no first-order reduction from $\mathsf{2DM}$ to $\mathsf{LUB\text{-}CQA}(g())$. It follows from Lemma 7.4.1 that the attack graph of $\exists\vec{u}(q(\vec{u}))$ is acyclic. □

In the remainder of this section, we provide a proof of Proposition 7.2.4 that relies solely on syntax. We begin by proving some auxiliary lemmas.

Lemma D.2.1. *Let q be a query in* sjfBCQ*. Let $F \in q$ and $v \in \mathsf{vars}(q)$ such that $F \overset{q}{\rightsquigarrow} v$. Let q' be a kernel of q with an acyclic attack graph. There is an atom $F' \in q'$ such that:*

- $F' \overset{q'}{\rightsquigarrow} v$*;*
- $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F) \rightarrow \mathsf{Key}(F')$*; and*
- $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F') \rightarrow \mathsf{Key}(F)$*.*

Proof. Since $F \overset{q}{\rightsquigarrow} v$, there is a path $(v_1, \dots, v_n)$ in the Gaifman graph of q such that $v_1 \in \mathsf{notKey}(F)$, $v_n = v$, and

$$\text{for every } i \in \{1, \dots, n\},\ \mathcal{K}(q \setminus \{F\}) \not\models \mathsf{Key}(F) \rightarrow v_i. \tag{D.3}$$

We will show, by induction on increasing i, that for every $i \in \{1, \dots, n\}$, there is an atom $F'_i \in q'$ such that:

- $F'_i \overset{q'}{\rightsquigarrow} v_i$;
- $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F) \rightarrow \mathsf{Key}(F'_i)$; and
- $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F'_i) \rightarrow \mathsf{Key}(F)$.

Clearly, this suffices to prove Lemma D.2.1 by picking $F' = F'_n$.

$\boxed{\text{Basis } i = 1.}$ From $v_1 \in \mathsf{notKey}(F)$ and $\mathcal{K}(q) \equiv \mathcal{K}(q')$, it follows $\mathcal{K}(q') \models \mathsf{Key}(F) \rightarrow v_1$. Hence, there is a shortest sequence $(H_1, \dots, H_\ell)$ that is a sequential proof of $\mathcal{K}(q') \models \mathsf{Key}(F) \rightarrow v_1$. Note that, since $v_1 \notin \mathsf{Key}(F)$, we have $\ell \geq 1$. For every $j \in \{1, \dots, \ell\}$, let z_j be the unique variable in $\mathsf{notKey}(H_j)$. Let $k \in \{1, \dots, \ell\}$ be the smallest index such that

$$\mathcal{K}(q \setminus \{F\}) \not\models \mathsf{Key}(H_k) \rightarrow z_k. \tag{D.4}$$

Such a k exists, because otherwise we have that $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F) \rightarrow v_1$, which contradicts (D.3). Clearly, $\mathcal{K}(\{H_1, H_2, \dots, H_{k-1}\}) \models \mathsf{Key}(F) \rightarrow \mathsf{Key}(H_k)$. By our choice of k, we have $\mathcal{K}(q \setminus \{F\}) \models \mathcal{K}(\{H_1, H_2, \dots, H_{k-1}\})$. Consequently,

$$\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F) \rightarrow \mathsf{Key}(H_k). \tag{D.5}$$

From $\mathcal{K}(q) \models \mathsf{Key}(H_k) \rightarrow z_k$ (because $\mathcal{K}(q) \equiv \mathcal{K}(q')$) and (D.4), it follows that every sequential proof of $\mathcal{K}(q) \models \mathsf{Key}(H_k) \rightarrow z_k$ (there exists at least one such proof) must use F, and hence

$$\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(H_k) \rightarrow \mathsf{Key}(F). \tag{D.6}$$

By Lemma 7.2.1, along with the definition of a sequential proof, it follows that every atom in the sequential proof $(H_1, \ldots, H_\ell)$, except the last one, attacks some atom that occurs after it in the sequential proof. Since the attack graph of q' is acyclic, it follows from (Koutris & Wijsen, 2017, Lemma 3.5) that it is transitive. It is correct to conclude

$$H_k \overset{q'}{\rightsquigarrow} v_1. \tag{D.7}$$

From (D.5), (D.6), and (D.7), it follows that the desired results holds true by choosing $F'_1 = H_k$.

$\boxed{\text{Step } i-1 \rightarrow i.}$ We now show that the claim holds true when $i > 1$, assuming that it holds true for $i-1$. By induction hypothesis, we have that here is an atom $F'_{i-1} \in q'$ such that:

(a) $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F) \rightarrow \mathsf{Key}(F'_{i-1})$;

(b) $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F'_{i-1}) \rightarrow \mathsf{Key}(F)$; and

(c) $F'_{i-1} \overset{q'}{\rightsquigarrow} v_{i-1}$.

If $F'_{i-1} \overset{q'}{\rightsquigarrow} v_i$, then the desired results obtains by picking $F'_i = F'_{i-1}$. Assume that $F'_{i-1} \overset{q'}{\not\rightsquigarrow} v_i$ from here on. Since $F'_{i-1} \overset{q'}{\rightsquigarrow} v_{i-1}$ and the Gaifman graph of q' contains an edge between v_i and v_{i-1}, it follows that $\mathcal{K}(q' \setminus \{F'_{i-1}\}) \models \mathsf{Key}(F'_{i-1}) \rightarrow v_i$. We can assume a shortest sequence $(H_1, \ldots, H_\ell)$ that is a sequential proof of $\mathcal{K}(q' \setminus \{F'_{i-1}\}) \models \mathsf{Key}(F'_{i-1}) \rightarrow v_i$. Note that $v_i \notin \mathsf{Key}(F'_{i-1})$, because otherwise, by ((a)), $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F) \rightarrow v_i$, which contradicts (D.3). Thus, $\ell \geq 1$. For every $j \in \{1, \ldots, \ell\}$, let z_j be the unique variable in $\mathsf{notKey}(H_j)$. Let $k \in \{1, \ldots, \ell\}$ be the smallest index such that

$$\mathcal{K}(q \setminus \{F\}) \not\models \mathsf{Key}(H_k) \rightarrow z_k. \tag{D.8}$$

Such a k exists, because otherwise we have $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F'_{i-1}) \rightarrow v_i$, and hence, by ((a)), $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F) \rightarrow v_i$, which contradicts (D.3). By the same reasoning as in the Basis of the induction, we obtain $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F'_{i-1}) \rightarrow \mathsf{Key}(H_k)$, hence, by ((a)),

$$\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F) \rightarrow \mathsf{Key}(H_k). \tag{D.9}$$

Like in the Basis of the induction, from (D.8) and $\mathcal{K}(q) \models \mathsf{Key}(H_k) \rightarrow z_k$, it follows

$$\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(H_k) \rightarrow \mathsf{Key}(F). \tag{D.10}$$

By the same reasoning as in the Basis of the induction,

$$H_k \overset{q'}{\rightsquigarrow} v_1. \tag{D.11}$$

From (D.9), (D.10), and (D.11), it follows that the desired results holds true by choosing $F_i' = H_k$. □

Lemma D.2.2. *Let q be a query in* sjfBCQ. *Let F, G be atoms of q such that $F \overset{q}{\rightsquigarrow} G$ and $G \overset{q}{\rightsquigarrow} F$. Let q' be a kernel of q with an acyclic attack graph. For every $v \in \mathsf{vars}(q)$, if $F \overset{q}{\rightsquigarrow} v$, then there is an atom $F' \in q'$ such that*

- *$F' \overset{q'}{\rightsquigarrow} v$; and*
- *$G \overset{q}{\rightsquigarrow} w$ for some $w \in \mathsf{Key}(F')$.*

Proof. Let $v \in \mathsf{vars}(q)$ such that $F \overset{q}{\rightsquigarrow} v$. By Lemma D.2.1, there is an atom $F' \in q'$ such that :

(a) $F' \overset{q'}{\rightsquigarrow} v$;

(b) $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F) \rightarrow \mathsf{Key}(F')$; and

(c) $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F') \rightarrow \mathsf{Key}(F)$.

It remains to show that $G \overset{q}{\rightsquigarrow} w$ for some $w \in \mathsf{Key}(F')$. From $G \overset{q}{\rightsquigarrow} F$, there is $z \in \mathsf{Key}(F)$ such that $G \overset{q}{\rightsquigarrow} z$. By ((c)), $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F') \rightarrow z$. There is a shortest sequence $(H_1, \dots, H_n)$ that is a sequential proof for $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F') \rightarrow z$. Assume, for the sake of contradiction, that $G \in (H_1, \dots, H_n)$. Then $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F') \rightarrow \mathsf{Key}(G)$, hence, by ((b)), $\mathcal{K}(q \setminus \{F\}) \models \mathsf{Key}(F) \rightarrow \mathsf{Key}(G)$, which contradicts $F \overset{q}{\rightsquigarrow} G$. We conclude by contradiction that $G \notin (H_1, \dots, H_n)$. Since $G \overset{q}{\rightsquigarrow} z$, by (Amezian El Khalfioui & Wijsen, 2024b, Lemma A.3), we have that $G \overset{q}{\rightsquigarrow} w$ for some $w \in \mathsf{Key}(F')$. □

We can now proceed with the proof of Proposition 7.2.4.

Syntax-Based Proof of Proposition 7.2.4. We prove the contrapositive: if an sjfBCQ query has a cyclic attack graph, then it is not κ-acyclic. To this end, assume that the attack graph of q contains a cycle. From (Koutris & Wijsen, 2017, Lemma 3.6), there are two distinct atoms $F, G \in q$ such that $F \overset{q}{\rightsquigarrow} G$ and $G \overset{q}{\rightsquigarrow} F$. Assume, for the sake of contradiction, that the attack graph of q' is acyclic. We will show that we can build an infinite sequence of atoms $(H_1, H_2, \dots)$ such that for every $i \in \{1, 2, \dots\}$, $H_i \neq H_{i+1}$ and $H_{i+1} \overset{q'}{\rightsquigarrow} H_i$.

Since $F \overset{q}{\rightsquigarrow} G$, there is a variable $v \in \mathsf{Key}(G)$ such that $F \overset{q}{\rightsquigarrow} v$. By Lemma D.2.2, there is an atom $H_1 \in q'$ such that $G \overset{q}{\rightsquigarrow} w_1$ for some $w_1 \in \mathsf{Key}(H_1)$.

- From $G \overset{q}{\rightsquigarrow} w_1$, by Lemma D.2.2 (with the roles of F and G switched), there is an atom $H_2 \in q'$ such that $H_2 \overset{q'}{\rightsquigarrow} w_1$ (hence $H_2 \neq H_1$ and $H_2 \overset{q'}{\rightsquigarrow} H_1$) and $F \overset{q}{\rightsquigarrow} w_2$ for some $w_2 \in \mathsf{Key}(H_2)$.
- From $F \overset{q}{\rightsquigarrow} w_2$, by Lemma D.2.2, there is an atom $H_3 \in q'$ such that $H_3 \overset{q'}{\rightsquigarrow} w_2$ (hence $H_3 \neq H_2$ and $H_3 \overset{q'}{\rightsquigarrow} H_2$) and $G \overset{q}{\rightsquigarrow} w_3$ for some $w_3 \in \mathsf{Key}(H_3)$.
- And so on.

Since q' is finite, eventually the sequence will contain a repeated atom, which by construction of our sequence induces a cycle in the attack graph of q', a contradiction. This concludes the proof of Lemma 7.2.4. □

D.3. Decomposition and Consistent Extension Lemmas for LUB-CQA

In this section, after some preliminaries in Section D.3.1, we provide formal proofs for Lemma 7.3.2 (*Decomposition Lemma for* LUB-CQA) in Section D.3.2 and Lemma 7.3.3 (*Consistent Extension Lemma for* LUB-CQA) in Section D.3.3.

D.3.1 Preliminaries

Definition D.3.1 (Pre-attacks)**.** Let q be a query in sjfBCQ, and $\leq$ be a linear order on the atoms of q. We write $F < G$ if $F \leq G$ and $F \neq G$. For every atom G in q, we define $G^{+,(q,<)}$ as the closure of $\mathsf{Key}(G)$ with respect to $\{\mathsf{Key}(F) \rightarrow \mathsf{vars}(F) \mid F < G\}$. A variable u is said to $<$*-precede* G, denoted $u < G$, if there is an atom F such that $F < G$ and $u \in \mathsf{vars}(F)$.

Let G be an atom of q, and $u \in \mathsf{vars}(q)$. We say that G *pre-attacks* u (with respect to $(q, \leq)$), denoted $G \overset{(q,\leq)}{\looparrowright} u$, if $\mathcal{G}aifman(q)$ contains a path $(v_0, v_1, \ldots, v_n)$ with $n \geq 0$, $v_0 \in \mathsf{notKey}(G)$, and $v_n = u$ such that no v_i belongs to $G^{+,(q,\leq)}$. For an atom H in q, we write $G \overset{(q,\leq)}{\looparrowright} H$ if $H \neq G$ and $G \overset{(q,\leq)}{\looparrowright} u$ for some $u \in \mathsf{vars}(H)$. □

Lemma D.3.1. *Let q be an irreducible* sjfBCQ *query with an acyclic attack graph. Let $\leq$ be a topological sort of q's attack graph. For every atom G in q, for every variable u such that $u < G$, we have $G \overset{(q,\leq)}{\not\rightsquigarrow} u$.*

Proof. Assume for the sake of a contradiction that there exists an atom H in q and a variable u such that $u < H$ and $H \overset{(q,\leq)}{\rightsquigarrow} u$. For ease of notation, let $\Sigma_{<H} = \mathcal{K}(\{F \in q \mid F < H\})$. We can assume a path $(u_0, u_1, \ldots, u_s)$ in $\mathcal{G}aifman(q)$ such that $\mathsf{notKey}(H) = \{u_0\}$, $u_s = u$, and no u_i is in $H^{+,(q,<)}$, which implies the following:

$$\text{for every } i \in \{0, 1, \ldots, s\}, \text{ we have } \Sigma_{<H} \not\models \mathsf{Key}(H) \rightarrow u_i. \tag{D.12}$$

We show that for some $\ell \geq 1$ there is a sequence

$$(K_1, u_{i_1}), (K_2, u_{i_2}), \ldots, (K_\ell, u_{i_\ell}), \tag{D.13}$$

such that $0 \leq i_1 < i_2 < \cdots < i_\ell = s$ and for each $j \in \{1, 2, \ldots, \ell\}$,

(a) $H \leq K_j$;

(b) $K_j \overset{q}{\rightsquigarrow} u_{i_j}$; and

(c) $\Sigma_{<H} \models \mathsf{Key}(H) \rightarrow \mathsf{Key}(K_j)$.

The proof runs by induction:

Induction basis: We will show that we can choose (K_1, u_{i_1}) with the desired properties.

Induction step: Suppose (K_j, u_{i_j}) exists with the desired properties for some $j \geq 1$. We will show that if $i_j \neq s$, then we can choose $(K_{j+1}, u_{i_{j+1}})$ with the desired properties.

For the induction basis, $j = 1$, we choose $K_1 := H$ and $i_1 := 0$. It is easily verified that with this choice, items ((a))–((c)) are satisfied for $j = 1$. For the induction step, $j \rightarrow j + 1$, the hypothesis is that items ((a))–((c)) hold true for some (K_j, u_{i_j}) with $j \geq 1$. Assume $i_j \neq s$ (otherwise the proof of the construction of the sequence (D.13) is concluded). Since $K_j \overset{q}{\rightsquigarrow} u_{i_j}$ but $K_j \overset{q}{\not\rightsquigarrow} u_s$ (because $\leq$ is a topological sort of q's attack graph, and $u_s < H \leq K_j$), there is $g \in \{i_j{+}1, i_j{+}2, \ldots, s\}$ such that $u_g \in {K_j}^{+,q}$. Then, for some $m \geq 1$, there is a shortest sequence $(L_1, L_2, \ldots, L_m)$ that is a sequential proof of $\mathcal{K}(q \setminus \{K_j\}) \models \mathsf{Key}(K_j) \rightarrow u_g$. Assume for the sake of a contradiction that each atom in the sequential proof $<$-precedes H. Then, $\Sigma_{<H} \models \mathsf{Key}(K_j) \rightarrow u_g$. By ((c)), it follows $\Sigma_{<H} \models \mathsf{Key}(H) \rightarrow u_g$, which contradicts (D.12). We conclude by contradiction that there is a smallest $f \in \{1, \ldots, m\}$ such that $H \leq L_f$. We choose $K_{j+1} := L_f$ (thereby satisfying ((a)) for $j+1$) and $i_{j+1} := g$ (hence $i_j < i_{j+1}$). By our choice of f, the (possibly empty) sequence $(L_1, \ldots, L_{f-1})$ is a sequential proof of $\Sigma_{<H} \models \mathsf{Key}(K_j) \rightarrow \mathsf{Key}(K_{j+1})$. Since ((c)) holds for j, it follows $\Sigma_{<H} \models \mathsf{Key}(H) \rightarrow \mathsf{Key}(K_{j+1})$, hence ((c)) holds for $j+1$ as well. Since the sequential proof is shortest possible, $\mathsf{notKey}(L_m) = \{u_g\}$. Since no atom is redundant in the sequential proof, it follows from Lemma 7.2.1 that for every $i \in \{1, \ldots, m-1\}$, there is $i' \in \{i+1, i+2, \ldots, m\}$ such that $L_i \overset{q}{\rightsquigarrow} L_{i'}$. Since acyclic attacks graphs are known to be transitive (Koutris & Wijsen, 2017, Lemma 3.5), it follows $K_{j+1} \overset{q}{\rightsquigarrow} u_g$ with $g = i_{j+1}$, thereby satisfying ((b)) for $j+1$.

The sequence (D.13) shows that there is an atom K_ℓ in q such that $K_\ell \overset{q}{\rightsquigarrow} u$ with $u < H \leq K_\ell$, contradicting that $\leq$ is a topological sort of the attack graph of q. This concludes the proof. $\square$

Lemma D.3.2. *Let q be an irreducible* sjfBCQ *query with an acyclic attack graph, topologically sorted as $(F_1, \ldots, F_n)$. Let*

$$q_\alpha := \{F_\ell\} \cup \{F_i \mid 1 \leq i \leq n \text{ and } F_\ell \overset{(q,\leq)}{\not\mapsto} F_i\}; \text{ and}$$
$$q_\beta := \{F_{\ell+1}, F_{\ell+2}, \ldots, F_n\} \setminus q_\alpha.$$

Then,

- $q_\alpha \cap \{F_1, F_2, \ldots, F_{\ell-1}\} = \emptyset$*; and*
- *for every $u \in \mathsf{vars}(q_\alpha) \cap \mathsf{vars}(q_\beta)$, we have $u \in F_\ell^{+,(q,<)}$.*

Proof. The first item is an immediate consequence of Lemma D.3.1. To prove the second item, let $u \in \mathsf{vars}(q_\alpha) \cap \mathsf{vars}(q_\beta)$. We can assume $G \in q_\beta$ such that $u \in \mathsf{vars}(G)$. From $G \notin q_\alpha$, it follows $F_\ell \overset{(q,\leq)}{\not\rightarrow} G$, hence $F_\ell \overset{(q,\leq)}{\not\rightarrow} u$. Since $u \in \mathsf{vars}(F)$ for some $F \in q_\alpha$, it must be the case that $u \in F_\ell^{+,(q,<)}$. $\square$

We are now ready to present the *Decomposition Lemma* (Lemma 7.3.2) and the *Consistent Extension Lemma* (Lemma 7.3.3). Together, these lemmas establish that the diagram in Fig. D.1 commutes under their hypotheses on q and $\mathcal{F}_{\mathsf{AGG}}$, which is key to the proof of Lemma 7.3.1.

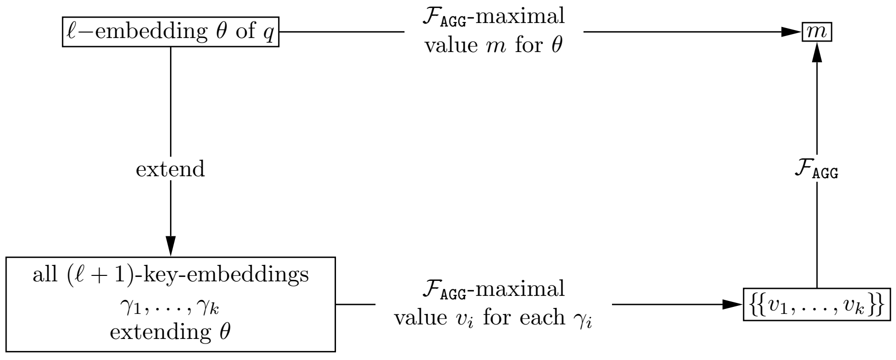
ℓ–embedding θ of q
$\mathcal{F}_{\texttt{AGG}}$-maximal value m for θ
m
extend
$\mathcal{F}_{\texttt{AGG}}$
all $(\ell+1)$-key-embeddings $\gamma_1, \dots, \gamma_k$ extending θ
$\mathcal{F}_{\texttt{AGG}}$-maximal value v_i for each γ_i
$\{\{v_1, \dots, v_k\}\}$

Figure D.1: Commutative diagram established by the *Decomposition Lemma* (Lemma 7.3.2) and the *Consistent Extension Lemma* (Lemma 7.3.3), which forms the crux of the proof of Lemma 7.3.1.

D.3.2 The *Decomposition Lemma* (Lemma 7.3.2)

The following helping lemma states that for every ℓ-embedding θ and $(\ell+1)$-key-embedding γ extending θ, if an MCS of $\mathsf{Ext}(\theta)$ is restricted to those valuations that extend γ, the result is an MCS of $\mathsf{Ext}(\gamma)$.

Lemma D.3.3. *Let q be an irreducible* sjfBCQ *query with an acyclic attack graph, topologically sorted as $(F_1, \dots, F_n)$. Let* $\mathbf{db}$ *be a database instance. Let $\ell \in \{0, \dots, n-1\}$. Let θ be an ℓ-embedding of q in* $\mathbf{db}$*, and let N be an MCS of* $\mathsf{Ext}(\theta)$*. Let γ be an (ℓ+1)-key-embedding of q in* $\mathbf{db}$ *that extends θ. Let N_γ be the subset of N containing all (and only) embeddings that extend γ. Then, N_γ is an MCS of* $\mathsf{Ext}(\gamma)$.

Proof. Assume, for the sake of a contradiction, that N_γ is not an MCS of $\mathsf{Ext}(\gamma)$. Since $N_\gamma \models \mathcal{K}(q)$, there is an MCS N^* of $\mathsf{Ext}(\gamma)$ such that $N_\gamma \subsetneq N^*$. We can assume $\mu \in N^* \setminus N_\gamma$. From $\mu \notin N_\gamma$, it follows $\mu \notin N$. Let k be the greatest integer in $\{\ell, \dots, n\}$ such that for some $\mu' \in N$, we have $\mu(\{F_1, \dots, F_k\}) = \mu'(\{F_1, \dots, F_k\})$. It is easily verified that such k exists. If $k = n$, then $\mu \in N$, a contradiction. Assume from here on that $k < n$. Let μ_k be the restriction of μ to $\mathsf{vars}(\{F_1, \dots, F_k\}) \cup \mathsf{Key}(F_{k+1})$.

Claim 14. For every $i \in \{k+1, \dots, n\}$, there is an embedding $\mu_i^* \in \mathsf{Ext}(\theta)$ such that μ_i^* extends μ_k and $N \cup \{\mu_i^*\} \models \mathcal{K}(\{F_1, \dots, F_i\})$.

Proof of Claim 14. The proof is by induction on increasing i.

Induction basis $i = k + 1$. The desired result is immediate if $N \cup \{\mu\} \models \mathcal{K}(\{F_1, \dots, F_{k+1}\})$. Assume $N \cup \{\mu\} \not\models \mathcal{K}(\{F_1, \dots, F_{k+1}\})$ from here on. Then, there is an embedding $\mu_c \in N$ such that $\mu(F_{k+1})$ and $\mu_c(F_{k+1})$ are key-equal but distinct, with $\{\mu, \mu_c\} \models \mathcal{K}(\{F_1, \dots, F_k\})$, hence μ and μ_c agree on all variables in $F_{k+1}^{+,(q,<)}$. Let

$$q_\alpha := \{F_{k+1}\} \cup \{F_i \mid 1 \leq i \leq n \text{ and } F_{k+1} \overset{(q,\leq)}{\hookrightarrow} F_i\};$$
$$q_\beta := \{F_{k+2}, F_{k+3}, \dots, F_n\} \setminus q_\alpha.$$

By Lemma D.3.2, there exists an embedding μ_i^* such that for every atom F of q,

$$\mu_i^*(F) = \begin{cases} \mu(F) & \text{if } F \in \{F_1, \dots, F_k\} \cup q_\beta\text{; and} \\ \mu_c(F) & \text{if } F \in q_\alpha. \end{cases}$$

Clearly, $N \cup \{\mu_i^*\} \models \mathcal{K}(\{F_1, \dots, F_{k+1}\})$.

Induction step $i - 1 \to i$. The argumentation is the same as in the base case of the induction, with μ_{i-1}^* replacing μ.

This concludes the proof of Claim 14 □

Let $\mu^* = \mu_n^*$. Since μ^* extends μ_k, it follows that $\mu^* \notin N$, which contradicts that N is an MCS of $\mathsf{Ext}(\theta)$. We conclude by contradiction that N_γ is an MCS of $\mathsf{Ext}(\gamma)$. This concludes the proof of Lemma D.3.3. □

Proof of Lemma 7.3.2. Assume that $\bigcup_{i=1}^k N_i \models \mathcal{K}(q)$, which implies that the set $\bigcup_{i=1}^k N_i$ is an MCS of $\mathsf{Ext}(\theta)$. Let N be an $\mathcal{F}_{\mathtt{AGG}}$-maximal MCS of $\mathsf{Ext}(\theta)$, i.e,

$$m = \mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N\}\}\right).$$

For ease of notation, we define

$$\widehat{m} := \mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in \bigcup_{i=1}^{k} N_i\}\}\right). \tag{D.14}$$

For each $i \in \{1, \dots, k\}$, let N_i^- be the subset of N containing all (and only) embeddings that extend γ_i, and define $v_i^- := \mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N_i^-\}\}\right)$. Note that $\{N_1^-, N_2^-, \dots, N_k^-\}$ is a partition of N. Since $\mathcal{F}_{\mathtt{AGG}}$ is associative, it follows that

$$m = \mathcal{F}_{\mathtt{AGG}}\left(\{\{v_1^-, v_2^-, \dots, v_k^-\}\}\right), \tag{D.15}$$

and

$$\widehat{m} = \mathcal{F}_{\texttt{AGG}}\left(\{\{v_1, v_2, \ldots, v_k\}\}\right). \tag{D.16}$$

Since q is irreducible, by Lemma D.3.3, for every $i \in \{1, \ldots, k\}$, N_i^- is an MCS of $\mathsf{Ext}(\gamma_i)$. By the definition of $\mathcal{F}_{\texttt{AGG}}$-maximal MCS, we have that for every $i \in \{1, \ldots, k\}$, $v_i \geq v_i^-$. Thus, since $\mathcal{F}_{\texttt{AGG}}$ is monotone, we have that

$$\mathcal{F}_{\texttt{AGG}}\left(\{\{v_1, v_2, \ldots, v_k\}\}\right) \geq \mathcal{F}_{\texttt{AGG}}\left(\{\{v_1^-, v_2^-, \ldots, v_k^-\}\}\right). \tag{D.17}$$

By (D.15), (D.16) and (D.17), it follows that

$$\widehat{m} \geq m. \tag{D.18}$$

Since m is the $\mathcal{F}_{\texttt{AGG}}$-maximal value for θ, it follows that

$$\widehat{m} = m. \tag{D.19}$$

By (D.16) and (D.19), we can conclude that

$$m = \mathcal{F}_{\texttt{AGG}}\left(\{\{v_1, v_2, \ldots, v_k\}\}\right).$$

This concludes the proof. □

D.3.3 The *Consistent Extension Lemma* (Lemma 7.3.3)

We first show the following helping lemma.

Lemma D.3.4. *Let $\exists\vec{u}(q(\vec{u}))$ be an irreducible* sjfBCQ *query with an acyclic attack graph, topologically sorted as $(F_1, \ldots, F_n)$. Let $g()$ be the* AGGR[sjfBCQ] *query defined as $g() := \texttt{AGG}(r) \leftarrow q(\vec{u})$, where $\mathcal{F}_{\texttt{AGG}}$ is monotone and associative. Let $\mathbf{db}$ be a database instance. Let $\ell \in \{1, \ldots, n\}$. Let γ be an ℓ-key-embedding of q in $\mathbf{db}$. Let*

$$\begin{aligned} q_\alpha &:= \{F_\ell\} \cup \{F_i \mid i > \ell \text{ and } F_\ell \overset{(q,\leq)}{\nrightarrow} F_i\}; \text{ and} \\ q_\beta &:= \{F_{\ell+1}, F_{\ell+2}, \ldots, F_n\} \setminus q_\alpha. \end{aligned}$$

For $i \in \{\alpha, \beta\}$, let M_i be the set of embeddings of $\gamma(q_i)$ in $\mathbf{db}$, and let N_i be an MCS of M_i such that

- *N_i is $\mathcal{F}_{\texttt{AGG}}$-maximal if r is a variable in $\gamma(q_i)$; and*
- *N_i is $\mathcal{F}_{\texttt{COUNT}}$-maximal otherwise (i.e., if r is a variable not in $\gamma(q_i)$ or a constant).*

Then, $\gamma(q_\alpha)$ and $\gamma(q_\beta)$ have no variables in common, and $\{\gamma \cdot \theta_1 \cdot \theta_2 \mid \theta_1 \in N_\alpha \text{ and } \theta_2 \in N_\beta\}$ is an $\mathcal{F}_{\mathtt{AGG}}$-maximal MCS of $\mathsf{Ext}(\gamma)$.

Proof. It follows from Lemma D.3.2 that $\mathsf{vars}(\gamma(q_\alpha)) \cap \mathsf{vars}(\gamma(q_\beta)) = \emptyset$, and thus, for every $\theta_1 \in N_\alpha$ and $\theta_2 \in N_\beta$, we have that $\gamma \cdot \theta_1 \cdot \theta_2$ is a valid embedding of q in **db**. Let $N = \{\gamma \cdot \theta_1 \cdot \theta_2 \mid \theta_1 \in N_\alpha \text{ and } \theta_2 \in N_\beta\}$. Clearly, N is a subset of $\mathsf{Ext}(\gamma)$. Moreover, since N_α and N_β are MCSs of M_α and M_β respectively, it follows that N is an MCS of $\mathsf{Ext}(\gamma)$.

Let N^* be an $\mathcal{F}_{\mathtt{AGG}}$-maximal MCS of $\mathsf{Ext}(\gamma)$. For $i \in \{\alpha, \beta\}$, let N_i^* be the set of valuations obtained from N^* by restricting each valuation in N^* to the variables in $\mathsf{vars}(\gamma(q_i))$. Then, $N^* = \{\gamma \cdot \theta_1 \cdot \theta_2 \mid \theta_1 \in N_\alpha^* \text{ and } \theta_2 \in N_\beta^*\}$. Since $\mathsf{vars}(\gamma(q_\alpha)) \cap \mathsf{vars}(\gamma(q_\beta)) = \emptyset$ and N^* is an MCS of of $\mathsf{Ext}(\gamma)$, it is easily seen that each N_i^* is an MCS of M_i ($i \in \{\alpha, \beta\}$). By the definition of $\mathcal{F}_{\mathtt{AGG}}$-maximal MCS, it follows that

$$\mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N^*\}\}\right) \geq \mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N\}\}\right). \tag{D.20}$$

To show that N is $\mathcal{F}_{\mathtt{AGG}}$-maximal, we distinguish two cases. To ease the notation, for $i \in \{\alpha, \beta\}$, we define $c_i := |N_i|$ and $c_i^* := |N_i^*|$.

Case that $r \in \mathsf{dom}(\gamma)$ or r is a constant. Then, $\gamma(r)$ is a constant. Indeed, since γ is the identity on constants, if r is a constant, then $r = \gamma(r)$. We have that for every $i \in \{\alpha, \beta\}$, N_i is an $\mathcal{F}_{\mathtt{COUNT}}$-maximal MCS of M_i. With the construct $i\#t$ as defined in Definition 8.1.1, it follows

$$\mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N\}\}\right) = \mathcal{F}_{\mathtt{AGG}}\left(\{\{(c_\alpha * c_\beta)\#\gamma(r)\}\}\right), \tag{D.21}$$

and

$$\mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N^*\}\}\right) = \mathcal{F}_{\mathtt{AGG}}\left(\{\{(c_\alpha^* * c_\beta^*)\#\gamma(r)\}\}\right). \tag{D.22}$$

By the definition of $\mathcal{F}_{\mathtt{COUNT}}$-maximal MCS, for every $i \in \{\alpha, \beta\}$, $c_i \geq c_i^*$. Since $\mathcal{F}_{\mathtt{AGG}}$ is monotone,

$$\mathcal{F}_{\mathtt{AGG}}\left(\{\{(c_\alpha * c_\beta)\#\gamma(r)\}\}\right) \geq \mathcal{F}_{\mathtt{AGG}}\left(\{\{(c_\alpha^* * c_\beta^*)\#\gamma(r)\}\}\right). \tag{D.23}$$

From (D.21), (D.22), and (D.23), it follows

$$\mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N\}\}\right) \geq \mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N^*\}\}\right). \tag{D.24}$$

From (D.20) and (D.24), it follows

$$\mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N\}\}\right) = \mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N^*\}\}\right). \tag{D.25}$$

It follows that N is an $\mathcal{F}_{\mathtt{AGG}}$-maximal MCS of $\mathsf{Ext}(\gamma)$.

Case that r is a variable not in $\mathsf{dom}(\gamma)$. Assume, without loss of generality, that $r \in \mathsf{vars}(\gamma(q_\alpha))$. It follows that $r \notin \mathsf{vars}(\gamma(q_\beta))$. Thus, N_α is an $\mathcal{F}_{\mathtt{AGG}}$-maximal MCS of M_α, and N_β is an $\mathcal{F}_{\mathtt{COUNT}}$-maximal MCS of M_β. It follows that

$$\mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N\}\}\right) = \mathcal{F}_{\mathtt{AGG}}\left(\{\{c_\beta \# \mu(r) \mid \mu \in N_\alpha\}\}\right), \tag{D.26}$$

and

$$\mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N^*\}\}\right) = \mathcal{F}_{\mathtt{AGG}}\left(\{\{c^*_\beta \# \mu(r) \mid \mu \in N^*_\alpha\}\}\right). \tag{D.27}$$

Since $\mathcal{F}_{\mathtt{AGG}}$ is associative,

$$\mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N\}\}\right) = \mathcal{F}_{\mathtt{AGG}}\left(\{\{c_\beta \# \mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N_\alpha\}\}\right)\}\}\right), \tag{D.28}$$

and

$$\mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N^*\}\}\right) = \mathcal{F}_{\mathtt{AGG}}\left(\{\{c^*_\beta \# \mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N^*_\alpha\}\}\right)\}\}\right). \tag{D.29}$$

By the definition of $\mathcal{F}_{\mathtt{COUNT}}$-maximal MCS, $c_\beta \geq c^*_\beta$. By definition of an $\mathcal{F}_{\mathtt{AGG}}$-maximal MCS, $\mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N_\alpha\}\}\right) \geq \mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N^*_\alpha\}\}\right)$. Since $\mathcal{F}_{\mathtt{AGG}}$ is monotone, we obtain that

$$\begin{array}{c} \mathcal{F}_{\mathtt{AGG}}\left(\{\{c_\beta \# \mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N_\alpha\}\}\right)\}\}\right) \\ \rotatebox{90}{$\leq$} \\ \mathcal{F}_{\mathtt{AGG}}\left(\{\{c^*_\beta \# \mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N^*_\alpha\}\}\right)\}\}\right). \end{array} \tag{D.30}$$

From (D.28), (D.29) and (D.30), it follows

$$\mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N\}\}\right) \geq \mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N^*\}\}\right). \tag{D.31}$$

From (D.20) and (D.31), it follows

$$\mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N\}\}\right) = \mathcal{F}_{\mathtt{AGG}}\left(\{\{\mu(r) \mid \mu \in N^*\}\}\right). \tag{D.32}$$

It follows that N is an $\mathcal{F}_{\mathtt{AGG}}$-maximal MCS of $\mathsf{Ext}(\gamma)$.

The proof is now concluded. □

Proof of Lemma 7.3.3. For readability, we show the lemma for $k = 2$. The proof can easily be generalized for $k > 2$.

In the first part of the proof, we show that for $i \in \{1, 2\}$, there is an ℓ-embedding θ_i extending γ_i such that:

- every $\mathcal{F}_{\texttt{AGG}}$-maximal MCS of $\mathsf{Ext}(\theta_i)$ is also an $\mathcal{F}_{\texttt{AGG}}$-maximal MCS of $\mathsf{Ext}(\gamma_i)$; and
- $\{\theta_1, \theta_2\} \models \mathcal{K}(\{F_1, \dots, F_\ell\})$.

Note that for $i \in \{1,2\}$, $\mathsf{dom}(\gamma_i) = \left(\bigcup_{j=1}^{\ell-1} \mathsf{vars}(F_j)\right) \cup \mathsf{Key}(F_\ell)$, and $\mathsf{dom}(\theta_i) = \mathsf{dom}(\gamma_i) \cup \mathsf{notKey}(F_\ell)$. We distinguish two cases:

Case that γ_1 and γ_2 disagree on a variable of $\mathsf{Key}(F_\ell)$. Let $i \in \{1,2\}$. Let N_i^* be an $\mathcal{F}_{\texttt{AGG}}$-maximal MCS of $\mathsf{Ext}(\gamma_i)$. Let θ_i be the (unique) ℓ-embedding such that every embedding in N_i^* extends θ_i. Clearly, every $\mathcal{F}_{\texttt{AGG}}$-maximal MCS of $\mathsf{Ext}(\theta_i)$ is also an $\mathcal{F}_{\texttt{AGG}}$-maximal MCS of $\mathsf{Ext}(\gamma_i)$. Since γ_1 and γ_2 disagree on some variable of $\mathsf{Key}(F_\ell)$, it follows that $\{\theta_1, \theta_2\} \models \mathcal{K}(\{F_1, \dots, F_\ell\})$.

Case that γ_1 and γ_2 agree on all variables of $\mathsf{Key}(F_\ell)$. Let

$$q_\alpha := \{F_\ell\} \cup \{F_i \mid i > \ell \text{ and } F_\ell \overset{(q,\leq)}{\rightsquigarrow} F_i\}; \text{ and}$$
$$q_\beta := \{F_{\ell+1}, F_{\ell+2}, \dots, F_n\} \setminus q_\alpha.$$

From Lemma D.3.2 and $\{\gamma_1, \gamma_2\} \models \mathcal{K}(\{F_1, \dots, F_{\ell-1}\})$, it follows $\gamma_1(q_\alpha) = \gamma_2(q_\alpha)$. Let $N^{(\alpha)}$ be an MCS of $\mathsf{Ext}(\gamma_1 \mid \gamma_1(q_\alpha), \mathbf{db})$ such that $N^{(\alpha)}$ is $\mathcal{F}_{\texttt{AGG}}$-maximal if $r \in \mathsf{vars}(\gamma_1(q_\alpha))$, and $\mathcal{F}_{\texttt{COUNT}}$-maximal otherwise. For $j \in \{1,2\}$, let $N_j^{(\beta)}$ be an MCS of $\mathsf{Ext}(\gamma_j \mid \gamma_j(q_\beta), \mathbf{db})$ such that $N_j^{(\beta)}$ is $\mathcal{F}_{\texttt{AGG}}$-maximal if $r \in \mathsf{vars}(\gamma_j(q_\beta))$, and $\mathcal{F}_{\texttt{COUNT}}$-maximal otherwise. By Lemma D.3.4, the set $N_i^* := \{\gamma_i \cdot \delta \cdot \epsilon \mid \delta \in N^{(\alpha)} \text{ and } \epsilon \in N_i^{(\beta)}\}$ is an $\mathcal{F}_{\texttt{AGG}}$-maximal MCS of $\mathsf{Ext}(\gamma_i)$. Let θ_i be the (unique) ℓ-embedding such that every valuation in N_i^* extends θ_i. Clearly, every $\mathcal{F}_{\texttt{AGG}}$-maximal MCS of $\mathsf{Ext}(\theta_i)$ is also an $\mathcal{F}_{\texttt{AGG}}$-maximal MCS of $\mathsf{Ext}(\gamma_i)$. Since $N^{(\alpha)}$ is the same for $i = 1$ and $i = 2$, and since $F_\ell \in q_\alpha$, it follows that $\theta_1(F_\ell) = \theta_2(F_\ell)$. Consequently, $\{\theta_1, \theta_2\} \models \mathcal{K}(\{F_1, \dots, F_\ell\})$.

So it is correct to conclude that θ_1, θ_2 with the desired properties exist, which concludes the first part of the proof.

We are now ready to prove that the lemma holds for every choice of ℓ in the statement of the lemma. The proof is by induction on decreasing ℓ. It is straightforward to see that the lemma holds true when $\ell = n$. We next show that the lemma holds true for $\ell = g$, assuming that it holds true for $\ell = g+1$. Let $i \in \{1,2\}$. Let $\gamma_1^{+,(i)}, \dots, \gamma_{k_i}^{+,(i)}$ enumerate all extensions of θ_i that are $(g+1)$-key-embeddings of q in $\mathbf{db}$, where θ_i is the g-embedding whose existence was proved in the first part of the proof. By the induction hypothesis, for every $j \in \{1, \dots, k_i\}$, there is an $\mathcal{F}_{\mathsf{AGG}}$-maximal MCS $N_j^{+,(i)}$ of $\mathsf{Ext}(\gamma_j^{+,(i)})$ such that

$$\overbrace{\left(\bigcup_{j=1}^{k_1} N_j^{+,(1)}\right)}^{N_1} \cup \overbrace{\left(\bigcup_{j=1}^{k_2} N_j^{+,(2)}\right)}^{N_2} \models \mathcal{K}(q),$$

in which we define N_1 and N_2 as shown above. It remains to show that N_i is an $\mathcal{F}_{\mathsf{AGG}}$-maximal MCS of $\mathsf{Ext}(\gamma_i)$. Clearly, it suffices to show that N_i is an $\mathcal{F}_{\mathsf{AGG}}$-maximal MCS of $\mathsf{Ext}(\theta_i)$. To this end, let m_i denote the $\mathcal{F}_{\mathsf{AGG}}$-maximal value for θ_i in $\mathbf{db}$, as defined in Definition 5.0.2. Since $N_i \models \mathcal{K}(q)$, it follows from Lemma 7.3.2 that

$$m_i = \mathcal{F}_{\mathsf{AGG}}\left(\{\{v_1, v_2, \dots, v_{k_i}\}\}\right),$$

where for every $j \in \{1, \dots, k_i\}$, $v_j := \mathcal{F}_{\mathsf{AGG}}\left(\{\{\mu(r) \mid \mu \in N_j^{+,(i)}\}\}\right)$. From this, it is correct to conclude that N_i is an $\mathcal{F}_{\mathsf{AGG}}$-maximal MCS of $\mathsf{Ext}(\gamma_i)$, which concludes the proof of Lemma 7.3.3. $\square$

D.4. Proof of Lemma 7.4.1

Proof of Lemma 7.4.1. An embedding of $q(\vec{u})$ into a database instance is a valuation over $\vec{u}$ that maps every atom of $q(\vec{u})$ to a fact in the database instance. In the first part of the proof, we assume that $r = 1$, i.e., $\mathsf{LUB\text{-}CQA}(g())$ returns the number of embeddings of $q(\vec{u})$ in a repair that has the greatest number of embeddings. In the second part of the proof, we will adapt the proof to the case where $r \neq 1$. Assume that the attack graph of $\exists\vec{u}(q(\vec{u}))$ has a cycle. From (Koutris & Wijsen, 2017, Lemma 3.6), there exist two atoms F and G such that $F \overset{q}{\rightsquigarrow} G$ and $G \overset{q}{\rightsquigarrow} F$. Let $M \subseteq A \times B$ be an instance of $\mathsf{2DM}$ with $|A| = |B| = n$. Assume without loss of generality that $1 \notin A \cup B$. For every $(a, b) \in M$, let θ_b^a be the valuation over $\mathsf{vars}(q)$ such for every variable $v \in \mathsf{vars}(q)$,

$$\theta_b^a(v) := \begin{cases} a & \text{if } F \overset{q}{\not\rightsquigarrow} v \text{ and } G \overset{q}{\rightsquigarrow} v \\ b & \text{if } F \overset{q}{\rightsquigarrow} v \text{ and } G \overset{q}{\not\rightsquigarrow} v \\ (a,b) & \text{if } F \overset{q}{\rightsquigarrow} v \text{ and } G \overset{q}{\rightsquigarrow} v \\ 1 & \text{if } F \overset{q}{\not\rightsquigarrow} v \text{ and } G \overset{q}{\not\rightsquigarrow} v \end{cases}$$

Let $\mathbf{db}_M := \bigcup\{\theta_b^a(q) \mid (a,b) \in M\}$. Clearly, $\mathbf{db}_M$ is first-order computable from M. We now show the following:

(i) for every repair $\mathbf{r}$ of $\mathbf{db}_M$, the number of embeddings of q in $\mathbf{r}$ is at most n; and

(ii) $\mathbf{db}_M$ has a repair with n embeddings if and only if M contains a matching.

Claim 15. Let H be an atom of q. If $\theta_b^a(x) = 1$ for every variable $x \in \mathsf{Key}(H)$, then $\theta_b^a(y) = 1$ for every variable $y \in \mathsf{notKey}(H)$.

Proof. Assume $\theta_b^a(x) = 1$ for every variable $x \in \mathsf{Key}(H)$. Assume for the sake of a contradiction that $\theta_b^a(y) \neq 1$ for some variable $y \in \mathsf{notKey}(H)$. Then, then for some $E \in \{F, G\}$, we have $E \overset{q}{\rightsquigarrow} y$. It can be easily verified that $E \overset{q}{\rightsquigarrow} x$ for some variable $x \in \mathsf{Key}(H)$, contradicting that $\theta_b^a(x) = 1$. □

Claim 16. Let $(a,b) \in M$. Then,

- for every $x \in \mathsf{Key}(F)$, $\theta_b^a(x) \in \{a, 1\}$;
- for every $x \in \mathsf{Key}(G)$, $\theta_b^a(x) \in \{b, 1\}$.

Proof. Easy. □

Claim 17. Let μ be a valuation over $\mathsf{vars}(q)$ such that $(\mathbf{db}_M, \mu) \models q$. Let $(a,b) \in M$ such that $\mu(F) = \theta_b^a(F)$. For every $H \in q \setminus \{F\}$ such that $F \overset{q}{\rightsquigarrow} H$, there exists $a' \in A$ such that $\mu(H) = \theta_b^{a'}(H)$.

Proof. Let $H \in q \setminus \{F\}$ such that $F \overset{q}{\rightsquigarrow} H$. Then there is a sequence

$$(H_0, x_1, H_1, x_2, H_2, \ldots, H_i, x_{i+1}, H_{i+1}, \ldots, x_\ell, H_\ell)$$

where for every $i \in \{1, 2, \ldots, \ell\}$,

- H_i is an atom of q, with $H_0 = F$ and $H_\ell = H$;
- $F \overset{q}{\rightsquigarrow} x_i$; and
- $x_i \in \mathsf{vars}(H_{i-1}) \cap \mathsf{vars}(H_i)$.

We show, by induction on increasing $i = 0, \ldots, \ell$, that for every $i \in \{0, 1, \ldots, \ell\}$, there exists $a_i \in A$ such that $\mu(H_i) = \theta_b^{a_i}(H_i)$. The basis of the induction, $i = 0$, holds vacuously true by choosing $a_0 = a$. For the induction step,

$i \to i+1$, the induction hypothesis is that there there exists $a_i \in A$ such that $\mu(H_i) = \theta_b^{a_i}(H_i)$. Since $F \overset{q}{\rightsquigarrow} x_{i+1}$, we have $\mu(x_{i+1}) \in \{b, (a_i, b)\}$. Since $x_{i+1} \in \mathsf{vars}(H_{i+1})$ and $\mu(H_{i+1}) \in \mathbf{db}_M$, it follows that there is $a_{i+1} \in A$ such that $\mu(H_{i+1}) = \theta_b^{a_{i+1}}(H_{i+1})$. This concludes the proof of Claim 17 $\square$

Claim 18. Let μ be a valuation over $\mathsf{vars}(q)$ such that $(\mathbf{db}_M, \mu) \models q$. Let $(a, b) \in M$ such that $\mu(G) = \theta_b^a(G)$. For every $H \in q \setminus \{G\}$ such that $G \overset{q}{\rightsquigarrow} H$, there exists $b' \in B$ such that $\mu(H) = \theta_{b'}^a(H)$.

Proof. Symmetrical to the proof of Claim 17. $\square$

Claim 19. Let μ be a valuation over $\mathsf{vars}(q)$ such that $(\mathbf{db}_M, \mu) \models q$. Then, $\mu \in \{\theta_b^a \mid (a, b) \in M\}$.

Proof. There is $(a, b) \in M$ such that $\mu(F) = \theta_b^a(F)$. We next argue that (a, b) is unique. Indeed, there is a variable $x \in \mathsf{Key}(F)$ such that $G \overset{q}{\rightsquigarrow} x$, hence $\mu(x) = a$ or $\mu(x) = (a, b)$, implying that a is uniquely determined. Furthermore, since $F \overset{q}{\rightsquigarrow} G$, there is a variable $y \in \mathsf{notKey}(F)$ such that $F \overset{q}{\rightsquigarrow} y$, hence $\mu(y) = b$ or $\mu(y) = (a, b)$, implying that b is uniquely determined.

By symmetrical reasoning, there is a unique $(a', b') \in M$ such that $\mu(G) = \theta_{b'}^{a'}(G)$. Since $F \overset{q}{\rightsquigarrow} G$, it follows by Claim 17 that there exists a' such that $\mu(G) = \theta_b^{a'}(G)$. Consequently, $b = b'$. By symmetrical reasoning $a = a'$. Consequently, $\mu(G) = \theta_b^a(G)$.

Finally, let $H \in q \setminus \{F, G\}$. Then,

- if $F \overset{q}{\rightsquigarrow} H$ and $G \overset{q}{\rightsquigarrow} H$, then $\mu(H) = \theta_b^a(H)$. This follows from Claims 17 and 18.
- if $F \overset{q}{\rightsquigarrow} H$ and $G \overset{q}{\not\rightsquigarrow} H$, then $\mu(H) = \theta_b^{a_0}$ for every $(a_0, b) \in M$. This follows from Claims 17 and the observation that no H-fact contains constants from A;
- if $F \overset{q}{\not\rightsquigarrow} H$ and $G \overset{q}{\rightsquigarrow} H$, then $\mu(H) = \theta_{b_0}^a$ for every $(a, b_0) \in M$. This follows from Claims 18 and the observation that no H-fact contains constants from B; and
- if $F \overset{q}{\not\rightsquigarrow} H$ and $G \overset{q}{\not\rightsquigarrow} H$, then $\mu(H) = \theta_{b_0}^{a_0}$ for every $(a_0, b_0) \in M$. This follows from the observation that that no H-fact contains constants other than 1.

It follows $\mu(H) = \theta_b^a(H)$. This concludes the proof of Claim 19. $\square$

Claim 20. Let $(a_1, b_1), (a_2, b_2) \in M$. Let $\mathbf{r}$ be a repair of $\mathbf{db}_M$ such that $(\mathbf{r}, \theta_{b_1}^{a_1}) \models q$ and $(\mathbf{r}, \theta_{b_2}^{a_2}) \models q$. Then,

- if $a_1 = a_2$, then $b_1 = b_2$; and
- if $b_1 = b_2$, then $a_1 = a_2$.

Proof. We show the first item (the proof of the second item is identical). Let $a_1 = a_2$. Let $a := a_1$. Let $\mathbf{r}$ be a repair of $\mathbf{db}_M$ such that $\theta_{b_1}^{a}(q) \subseteq \mathbf{r}$ and $\theta_{b_2}^{a}(q) \subseteq \mathbf{r}$, hence $\theta_{b_1}^{a}(F) \in \mathbf{r}$ and $\theta_{b_2}^{a}(F) \in \mathbf{r}$. From Claim 16, it follows that the facts $\theta_{b_1}^{a}(F)$ and $\theta_{b_2}^{a}(F)$ are key-equal, and hence are equal because they belong to the same repair. Since $F \overset{q}{\rightsquigarrow} G$, there exists a variable $y \in \mathsf{notKey}(F)$ such that $F \overset{q}{\rightsquigarrow} y$, hence $\theta_{b_1}^{a}(y) \in \{b_1, (a, b_1)\}$ and $\theta_{b_2}^{a}(y) \in \{b_2, (a, b_2)\}$. From $\theta_{b_1}^{a}(F) = \theta_{b_2}^{a}(F)$, it follows $\theta_{b_1}^{a}(y) = \theta_{b_2}^{a}(y)$, hence $b_1 = b_2$. □

$\boxed{\text{Proof of (i).}}$ Immediate from Claims 19 and 20.

$\boxed{\text{Proof of (ii).}}$ Assume that $\mathbf{db}_M$ has n distinct embeddings. By Claim 19, there exists $M' \subseteq M$ with $|M'| = n$ such that the set of embeddings in $\mathbf{db}_M$ is $\{\theta_b^a \mid (a, b) \in M'\}$. By Claim 20, M' is a matching. Conversely, assume that $M' \subseteq M$ is a matching, and let $\mathbf{r} := \bigcup\{\theta_b^a(q) \mid (a, b) \in M'\}$. It suffices now to show that $\mathbf{r}$ is consistent. To this end, let H be an atom of q with relation name R_H. We consider all possibilities:

- If neither F nor G attacks H, then the R_H-relation of $\mathbf{r}$ is consistent by Claim 15.
- If both F and G attack H, then there is $v \in \mathsf{Key}(H)$ such that for every $(a, b) \in M'$, $\theta_b^a(v) = (a, b)$, hence no two distinct R_H-facts of $\mathbf{r}$ are key-equal.
- If $F \overset{q}{\rightsquigarrow} H$ but $G \overset{q}{\not\rightsquigarrow} H$, which includes the case $H = G$, then for all $v \in \mathsf{vars}(H)$ and $(a, b) \in M'$, $\theta_b^a(v) \in \{b, 1\}$. Moreover, there is $v \in \mathsf{Key}(H)$ such that for every $(a, b) \in M'$, $\theta_b^a(v) = b$. It follows that no two distinct R_H-facts of $\mathbf{r}$ are key-equal.
- The case $G \overset{q}{\rightsquigarrow} H$ but $F \overset{q}{\not\rightsquigarrow} H$ is symmetrical to the previous one.

Consequently, $\mathbf{r} \models \mathcal{K}(q)$.

We now show how to treat the case where $r \neq 1$. The easier case is where r is a constant distinct from 0, or r is a variable such that $F \overset{q}{\not\rightsquigarrow} v$ and $G \overset{q}{\not\rightsquigarrow} v$. We treat the difficult case where r is a numerical variable such that either $F \overset{q}{\not\rightsquigarrow} r$ or $G \overset{q}{\not\rightsquigarrow} r$ (or both). Assume $F \overset{q}{\rightsquigarrow} r$ and $G \overset{q}{\rightsquigarrow} r$ (the other cases are similar). In this case, let $m_1, m_2, \ldots, m_k$ enumerate all elements of M. We encode every m_i by $1 + \frac{1}{2^i}$. A repair with j embeddings (but not $j+1$ embeddings) will yield a sum s such that $j < s \leq j + (\frac{1}{2} + \frac{1}{4} + \ldots + \frac{1}{2^j}) < j + 1$. It follows that the number of embeddings is $\lfloor s \rfloor$. This concludes the proof of Lemma 7.4.1. $\Box$

Bibliography

Abiteboul, S., Hull, R., & Vianu, V. (1995). *Foundations of Databases*. Addison-Wesley.

Amezian El Khalfioui, A., Joertz, J., Labeeuw, D., Staquet, G., & Wijsen, J. (2020). Optimization of answer set programs for consistent query answering by means of first-order rewriting. In *CIKM*, (pp. 25–34). ACM.

Amezian El Khalfioui, A., & Wijsen, J. (2023). Consistent query answering for primary keys and conjunctive queries with counting. In *ICDT*, vol. 255 of *LIPIcs*, (pp. 23:1–23:19). Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Amezian El Khalfioui, A., & Wijsen, J. (2024a). Computing range consistent answers to aggregation queries via rewriting. *Proc. ACM Manag. Data*, *2*(5), 218:1–218:19.

Amezian El Khalfioui, A., & Wijsen, J. (2024b). Computing range consistent answers to aggregation queries via rewriting. *CoRR*, *abs/2409.01648*.

Amezian El Khalfioui, A., & Wijsen, J. (2026). Computing consistent least upper bounds in aggregate logic. In *ICDT, to appear*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Arenas, M., Bertossi, L. E., & Chomicki, J. (1999). Consistent query answers in inconsistent databases. In *PODS*, (pp. 68–79). ACM Press.

Arenas, M., Bertossi, L. E., & Chomicki, J. (2001). Scalar aggregation in FD-inconsistent databases. In *ICDT*, vol. 1973 of *Lecture Notes in Computer Science*, (pp. 39–53). Springer.

Bertossi, L. E. (2019). Database repairs and consistent query answering: Origins and further developments. In *PODS*, (pp. 48–58). ACM.

Bulatov, A. A. (2011). Complexity of conservative constraint satisfaction problems. *ACM Trans. Comput. Log.*, *12*(4), 24:1–24:66.

Calautti, M., Console, M., & Pieris, A. (2019). Counting database repairs under primary keys revisited. In *PODS*, (pp. 104–118). ACM.

Calautti, M., Console, M., & Pieris, A. (2021). Benchmarking approximate consistent query answering. In *PODS*, (pp. 233–246). ACM.

Calautti, M., Libkin, L., & Pieris, A. (2018). An operational approach to consistent query answering. In *PODS*, (pp. 239–251). ACM.

Calautti, M., Livshits, E., Pieris, A., & Schneider, M. (2022a). Counting database repairs entailing a query: The case of functional dependencies. In *PODS*, (pp. 403–412). ACM.

Calautti, M., Livshits, E., Pieris, A., & Schneider, M. (2022b). Uniform operational consistent query answering. In *PODS*, (pp. 393–402). ACM.

Chandra, A. K., Stockmeyer, L. J., & Vishkin, U. (1984). Constant depth reducibility. *SIAM J. Comput.*, *13*(2), 423–439.

Cohen, S., Nutt, W., & Sagiv, Y. (2006). Rewriting queries with arbitrary aggregation functions using views. *ACM Trans. Database Syst.*, *31*(2), 672–715.

Cohen, S., Nutt, W., & Serebrenik, A. (1999). Algorithms for rewriting aggregate queries using views. In *DMDW*, vol. 19 of *CEUR Workshop Proceedings*, (p. 9). CEUR-WS.org.

Dixit, A. A., & Kolaitis, P. G. (2019). A SAT-based system for consistent query answering. In *SAT*, vol. 11628 of *Lecture Notes in Computer Science*, (pp. 117–135). Springer.

Dixit, A. A., & Kolaitis, P. G. (2022). Consistent answers of aggregation queries via SAT. In *ICDE*, (pp. 924–937). IEEE.

Fan, Z., Koutris, P., Ouyang, X., & Wijsen, J. (2023). Lincqa: Faster consistent query answering with linear time guarantees. *Proc. ACM Manag. Data*, *1*(1), 38:1–38:25.

Figueira, D., Padmanabha, A., Segoufin, L., & Sirangelo, C. (2023). A simple algorithm for consistent query answering under primary keys. In *ICDT*, vol. 255 of *LIPIcs*, (pp. 24:1–24:18). Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Figueira, D., Padmanabha, A., Segoufin, L., & Sirangelo, C. (2025). A simple algorithm for consistent query answering under primary keys. *Logical Methods in Computer Science*, *Volume 21, Issue 1*.
URL `https://lmcs.episciences.org/12679`

Fontaine, G. (2015). Why is it hard to obtain a dichotomy for consistent query answering? *ACM Trans. Comput. Log.*, *16*(1), 7:1–7:24.

Fuxman, A. (2007). *Efficient query processing over inconsistent databases.*. Ph.D. thesis, University of Toronto.

Fuxman, A., Fazli, E., & Miller, R. J. (2005a). ConQuer: Efficient management of inconsistent databases. In *SIGMOD Conference*, (pp. 155–166). ACM.

Fuxman, A., Fuxman, D., & Miller, R. J. (2005b). ConQuer: A system for efficient querying over inconsistent databases. In *VLDB*, (pp. 1354–1357). ACM.

Fuxman, A., & Miller, R. J. (2005). First-order query rewriting for inconsistent databases. In *ICDT*, vol. 3363 of *Lecture Notes in Computer Science*, (pp. 337–351). Springer.

Garey, M. R., Graham, R. L., & Johnson, D. S. (1976). Some np-complete geometric problems. In *STOC*, (pp. 10–22). ACM.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Hannula, M., & Wijsen, J. (2022). A dichotomy in consistent query answering for primary keys and unary foreign keys. In *PODS*, (pp. 437–449). ACM.

Hella, L., Libkin, L., Nurmonen, J., & Wong, L. (2001). Logics with aggregate operators. *J. ACM*, *48*(4), 880–907.

Kimelfeld, B., & Kolaitis, P. G. (2024). A unifying framework for incompleteness, inconsistency, and uncertainty in databases. *Commun. ACM*, *67*(3), 74–83.

Kimelfeld, B., Livshits, E., & Peterfreund, L. (2020). Counting and enumerating preferred database repairs. *Theor. Comput. Sci.*, *837*, 115–157.

Kolaitis, P. G., Pardal, N., Virtema, J., & Wijsen, J. (2025). Rewriting consistent answers on annotated data. *Proc. ACM Manag. Data*, *3*(2), 110:1–110:26.

Kolaitis, P. G., Pema, E., & Tan, W. (2013). Efficient querying of inconsistent databases with binary integer programming. *Proc. VLDB Endow.*, *6*(6), 397–408.

Koutris, P., Ouyang, X., & Wijsen, J. (2021). Consistent query answering for primary keys on path queries. In *PODS*, (pp. 215–232). ACM.

Koutris, P., Ouyang, X., & Wijsen, J. (2024). Consistent query answering for primary keys on rooted tree queries. *Proc. ACM Manag. Data*, *2*(2). URL `https://doi.org/10.1145/3651139`

Koutris, P., & Wijsen, J. (2017). Consistent query answering for self-join-free conjunctive queries under primary key constraints. *ACM Trans. Database Syst.*, *42*(2), 9:1–9:45.

Koutris, P., & Wijsen, J. (2018). Consistent query answering for primary keys and conjunctive queries with negated atoms. In *PODS*, (pp. 209–224). ACM.

Koutris, P., & Wijsen, J. (2020). First-order rewritability in consistent query answering with respect to multiple keys. In *PODS*, (pp. 113–129). ACM.

Koutris, P., & Wijsen, J. (2021). Consistent query answering for primary keys in datalog. *Theory Comput. Syst.*, *65*(1), 122–178.

Lechtenbörger, J. (2004). Computing unique canonical covers for simple FDs via transitive reduction. *Inf. Process. Lett.*, *92*(4), 169–174.

Libkin, L. (2004). *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer.

Maier, D. (1980). Minimum covers in relational database model. *J. ACM*, *27*(4), 664–674.

Maier, D. (1983). *The Theory of Relational Databases*. Computer Science Press.

Maslowski, D., & Wijsen, J. (2013). A dichotomy in the complexity of counting database repairs. *J. Comput. Syst. Sci.*, *79*(6), 958–983.

Maslowski, D., & Wijsen, J. (2014). Counting database repairs that satisfy conjunctive queries with self-joins. In *ICDT*, (pp. 155–164). OpenProceedings.org.

Padmanabha, A., Segoufin, L., & Sirangelo, C. (2024). A dichotomy in the

complexity of consistent query answering for two atom queries with self-join. *Proc. ACM Manag. Data*, *2*(2).
URL `https://doi.org/10.1145/3651137`

Staworko, S., Chomicki, J., & Marcinkowski, J. (2012). Prioritized repairing and consistent query answering in relational databases. *Ann. Math. Artif. Intell.*, *64*(2-3), 209–246.

Wijsen, J. (2012). Certain conjunctive query answering in first-order logic. *ACM Trans. Database Syst.*, *37*(2), 9:1–9:35.

Wijsen, J. (2019). Foundations of query answering on inconsistent databases. *SIGMOD Rec.*, *48*(3), 6–16.