Rewriting with Acyclic Queries: Mind Your Head

Gaetano Geck
TU Dortmund University, Germany

Jens Keppeler
TU Dortmund University, Germany

Thomas Schwentick
TU Dortmund University, Germany

Christopher Spinrath
TU Dortmund University, Germany

Abstract

The paper studies the *rewriting problem*, that is, the decision problem whether, for a given conjunctive query Q and a set $\mathcal{V}$ of views, there is a conjunctive query Q' over $\mathcal{V}$ that is equivalent to Q, for cases where the query, the views, and/or the desired rewriting are acyclic or even more restricted.

It shows that, if Q itself is acyclic, an acyclic rewriting exists if there is any rewriting. An analogous statement also holds for free-connex acyclic, hierarchical, and q-hierarchical queries.

Regarding the complexity of the rewriting problem, the paper identifies a border between tractable and (presumably) intractable variants of the rewriting problem: for schemas of bounded arity, the *acyclic rewriting problem* is NP-hard, even if both Q and the views in $\mathcal{V}$ are acyclic or hierarchical. However, it becomes tractable, if the views are free-connex acyclic (i.e., in a nutshell, their body is (i) acyclic and (ii) remains acyclic if their head is added as an additional atom).

2012 ACM Subject Classification Theory of computation → Data integration; Theory of computation → Database query processing and optimization (theory)

Keywords and phrases rewriting, acyclic rewriting, acyclic conjunctive queries, free-connex queries, hierarchical queries, NP-hardness

Digital Object Identifier 10.4230/LIPIcs.ICDT.2022.8

Related Version *Full Version*: `https://arxiv.org/abs/2201.05129` [15]

1 Introduction

The *query rewriting problem* asks, for a query Q and a set $\mathcal{V}$ of views, whether there is a query Q' over $\mathcal{V}$ that is equivalent to Q, and to find such a *rewriting* Q'.

We emphasise that in the literature, various notions of rewriting are studied and that the one above is sometimes called *exact* [8] or *equivalent* [2]. Since we are in this paper exclusively interested in exact rewritings, we use the term rewriting in the sense stated above.

Over the last decades, query rewriting has been studied in a wide range of settings with varying database models (relational, XML, graph, ...), query languages, semantics (set, bag, ...) and more. A particular well-known setting is formed by conjunctive queries (CQ) under set semantics, which captures some core aspects of SQL queries. This is also the setting studied in this paper. We refer to [2, 11, 18] for an overview of the extensive literature.

In this paper we are interested in a modified version of the query rewriting problem which does not merely ask for *any* rewriting but rather for rewritings that are structurally simple and allow for efficient evaluation. This is relevant in a setting where access to the database is only possible through the views, in which case the use of an efficient rewriting is obvious. It can also be interesting in a scenario where multiple queries have to be evaluated and it could be useful to evaluate them in some order that allows many of them to be evaluated efficiently.

Table 1 Complexity results for the acyclic rewriting problem. Note that QHCQ $\subseteq$ CCQ $\subseteq$ ACQ and QHCQ $\subseteq$ HCQ $\subseteq$ ACQ hold. The head arity of a view V is the arity of its head atom head(V). The weak head arity of a view is defined in Definition 6.3. In a nutshell, a view with weak head arity ℓ can be "split" into (multiple) views with head arity at most ℓ.

$\mathbb{V}$ Views	$\mathbb{Q}$ Query	$\mathbb{R}$ Rewriting	Restriction of views	REWR$^k(\mathbb{V},\mathbb{Q},\mathbb{R})$ for every $k \in \mathbb{N}$ (bounded arity db-schema)	REWR$(\mathbb{V},\mathbb{Q},\mathbb{R})$ (unbounded arity db-schema)
ACQ	ACQ	ACQ		NP-complete for $k \geq 3$ (Theorem 5.7)	NP-complete for $k \geq 3$ (Theorem 5.7)
ACQ	ACQ	ACQ	head arity $\leq \ell$ $\ell \in \mathbb{N}$	in polynomial time (Corollary 5.5)	in polynomial time (Corollary 5.5)
ACQ	ACQ	ACQ	weak head arity $\leq \ell$ $\ell \in \mathbb{N}$	in polynomial time (Theorem 6.9)	in polynomial time (Theorem 6.9)
CCQ	ACQ	ACQ		in polynomial time (Theorem 6.2)	open
HCQ	HCQ	HCQ		NP-complete for $k \geq 3$ (Corollary 7.5)	NP-complete for $k \geq 3$ (Corollary 7.5)
QHCQ	HCQ	HCQ		in polynomial time (Corollary 7.4)	open
QHCQ	QHCQ	QHCQ		in polynomial time (Corollary 7.4)	open

The forms of structural simplicity that we consider in this paper are acyclic (ACQ) and hierarchical queries (HCQ), and their slightly stronger versions free-connex acyclic (CCQ) and q-hierarchical queries (QHCQ). For a brief discussion of the origin and applications of these classes, we refer to Section 8.

We are interested in two kinds of questions.

(1) Under which circumstances is it guaranteed that a structurally simple rewriting exists, if there exists a rewriting at all?

(2) What is the complexity to decide whether such a rewriting exists and to compute one?

We study these questions depending on the structure of the given views and the given query and we consider the same simplicity notions.

In particular, we study the decision problem REWR$(\mathbb{V},\mathbb{Q},\mathbb{R})$ that asks whether for a given set of views from $\mathbb{V}$ and a query from $\mathbb{Q}$, there is a rewriting from $\mathbb{R}$, for various classes $\mathbb{V}$, $\mathbb{Q}$ and $\mathbb{R}$, with an emphasis on the case where $\mathbb{R}$ is ACQ or a subclass. In the following we refer to this case as the *acyclic rewriting problem*. To the best of our knowledge, there has been no previous work dedicated to the study of questions (1) and (2).

The answer to question (1) turns out to be very simple and also quite encouraging: in the case that $\mathbb{Q} =$ ACQ, whenever a rewriting exists, there is also an acyclic rewriting. And the same is true for the three subclasses of ACQ. That is, for every query in CCQ, HCQ, or QHCQ, whenever a rewriting exists, there is also a rewriting in CCQ, HCQ, or QHCQ, respectively. Thus, in these cases a rewriting with efficient evaluation exists, if there is a rewriting at all. This answer to question (1) simplifies the study of question (2), since it implies that for $\mathbb{Q} \in \{\text{ACQ}, \text{CCQ}, \text{HCQ}, \text{QHCQ}\}$ and $\mathbb{V} \subseteq$ CQ the decision problems REWR$(\mathbb{V},\mathbb{Q},\text{CQ})$ and REWR$(\mathbb{V},\mathbb{Q},\mathbb{Q})$ – and, thus, their complexities – coincide.

The study of question (2) reveals that the complexity of the acyclic rewriting problem, may depend on two parameters: the arity of the underlying schema and the arity of the views. We denote the restriction of REWR$(\mathbb{V},\mathbb{Q},\mathbb{R})$ to database schemas of arity at most k by REWR$^k(\mathbb{V},\mathbb{Q},\mathbb{R})$, and we indicate by $\mathbb{V}^k$ if the arity of views is at most k.

Our main findings regarding the complexity of the acyclic rewriting problem are as follows (see Table 1 for an overview).

- If the query and the views are acyclic and the arity of the views is bounded by some fixed k, then the acyclic rewriting problem is tractable, that is, for every k: REWR(ACQ^k, ACQ, ACQ) is in polynomial time (Corollary 5.5). Furthermore, an acyclic rewriting can be computed in polynomial time (if it exists). This follows easily with the help of the canonical rewriting approach (see [26, Proposition 5.1]) and our answer to question (1).

- If the query and the views are acyclic and the arity of the views can be unbounded, then the acyclic rewriting problem is intractable, even if the database schema has bounded arity, more precisely: $\textsc{Rewr}^3(\mathsf{ACQ}, \mathsf{ACQ}, \mathsf{ACQ})$ is NP-complete (Theorem 5.7).
- If the query is acyclic, the views are free-connex acyclic, and the arity of the database schema is bounded by some fixed k, then the acyclic rewriting problem is tractable, that is, for every k: $\textsc{Rewr}^k(\mathsf{CCQ}, \mathsf{ACQ}, \mathsf{ACQ})$ is in polynomial time (Theorem 6.2).

Many results use a characterisation of rewritability, based on the notion of *cover partitions*, that might be interesting in its own right. Similar notions have been used in earlier work, but ours is particularly suited for the study of exact (equivalent) rewritings.

Our paper is organised as follows: we introduce general basic notions in Section 2 and notions related to rewritings in Section 3. The characterisation of rewritability in terms of cover partitions is given in Section 4. The answer to question (1) and the complexity of the acyclic rewriting problem for acyclic queries and views are studied in Section 5. The feasibility for free-connex acyclic views (and an additional class) is presented in Section 6. Section 7 shows that the answer to question (1) is the same for hierarchical and q-hierarchical queries and views. Section 8 discusses related work and, in particular, the relationship of our characterisation of rewritability with results in the literature. Section 9 concludes.

2 Preliminaries

In this section, we fix some notation and recall the basic concepts from relational databases that are relevant for this paper. Let dom and var be countably infinite sets of *data values* (also called *constants*) and *variables*, respectively. We use the natural extensions of mappings of variables onto tuples, sets without notational distinction. By id we denote the identity mapping (on any set of variables).

Databases and queries are formulated over *schemas* that represent relation names. A *schema* $\mathcal{S}$ is a finite set of relation symbols, where each symbol R is equipped with a fixed arity $\mathrm{ar}(R) \in \mathbb{N}_0$. A *fact* or *$R$-fact* $R(a_1, \ldots, a_r)$ comprises a relation symbol R with some arity r and data values $a_1, \ldots, a_r$. A *database* D over schema $\mathcal{S}$ is a finite subset of R-facts for $R \in \mathcal{S}$.

Queries. In this paper, we consider conjunctive queries and restrictions of them. These queries are conjunctions of relation atoms. An *atom* is of the form $R(x_1, \ldots, x_r)$ with a relation symbol R with arity r, and variable set $\mathrm{vars}(A) = \{x_1, \ldots, x_r\}$.

We represent a *conjunctive query* (*CQ* for short) over schema $\mathcal{S}$ as a rule of the form $A \leftarrow A_1, \ldots, A_m$ whose *body* $\{A_1, \ldots, A_m\}$ consists of a positive number of atoms and whose *head* is a single atom A such that the following two conditions are satisfied. First, atoms $A_1, \ldots, A_m$ refer to relation symbols from $\mathcal{S}$ and atom A, on the contrary, does not. Second, the query is *safe*, that is, every variable in the head occurs in at least one atom of the body. Let $\mathrm{head}(Q)$ and $\mathrm{body}(Q)$ denote the head and body of a query Q, respectively. Variables that occur in the head of a query are called *head variables*; all other variables are called *quantified variables*. The *arity* of a query Q is the arity of its head atom. Furthermore, a query Q is a Boolean query if the arity of $\mathrm{head}(Q)$ is 0.

A *valuation* is a mapping $\vartheta : \mathsf{var} \rightharpoonup \mathsf{dom}$. A database D *satisfies* a set $\mathcal{A}$ of atoms under a valuation ϑ, if $\vartheta(\mathcal{A}) \subseteq D$, that is for every atom $R(x_1, \ldots, x_r)$ in $\mathcal{A}$ the fact $R(\vartheta(x_1), \ldots, \vartheta(x_r))$ is contained in D. The *result* of query Q on database D is defined as

$$Q(D) = \{\vartheta(\mathrm{head}(Q)) \mid D \text{ satifies } \mathrm{body}(Q) \text{ under } \vartheta\}.$$

Relationships between queries. Queries over the same schema can be compared with respect to the results they define. Let Q_1 and Q_2 be queries. We say that Q_1 is *contained* in Q_2 (notation: $Q_1 \sqsubseteq Q_2$) if $Q_1(D) \subseteq Q_2(D)$ for every database D. We say that Q_1 and Q_2 are *equivalent* (notation: $Q_1 \equiv Q_2$) if $Q_1 \sqsubseteq Q_2$ and $Q_2 \sqsubseteq Q_1$.

It is well-known that a conjunctive query Q_1 is contained in a conjunctive query Q_2 if and only if there is a homomorphism from the latter query into the first [9]. Such a *homomorphism* is a mapping $h : \text{vars}(Q_2) \to \text{vars}(Q_1)$ such that (1) $h(\text{body}(Q_2)) \subseteq \text{body}(Q_1)$ and (2) $h(\text{head}(Q_2)) = \text{head}(Q_1)$. We call h a *body homomorphism* if it fulfils condition (1).

A conjunctive query Q_1 is *minimal* if there is no conjunctive query Q_2 such that $Q_2 \equiv Q_1$ and $|\text{body}(Q_2)| < |\text{body}(Q_1)|$ holds.

Structurally simple queries. As mentioned in the introduction, we are particularly interested in the following classes of structurally simple queries. A *join tree* for a query Q is a tree whose vertices are the atoms in the query's body that satisfies the following path property: for two atoms $A, A' \in \text{body}(Q)$ with a common variable x, all atoms on the path from A to A' contain x. A query Q is *acyclic* if it has a join tree. It is *free-connex acyclic* if Q is acyclic and the Boolean query whose body is $\text{body}(Q) \cup \{\text{head}(Q)\}$ is acyclic as well [3, 7]. For a fixed query Q and some variable x in Q, let $\text{atoms}(x)$ denote the set of atoms in $\text{body}(Q)$ in which x appears.

▶ **Definition 2.1** ([12], [6, Definition 3.1]). A conjunctive query Q is *hierarchical* if, for all variables x, y in Q, one of the following conditions is satisfied.
(1) $\text{atoms}(x) \subseteq \text{atoms}(y)$
(2) $\text{atoms}(x) \supseteq \text{atoms}(y)$
(3) $\text{atoms}(x) \cap \text{atoms}(y) = \emptyset$

A conjunctive query Q is *q-hierarchical* if it is hierarchical and for all variables $x, y \in \text{vars}(Q)$ the following is satisfied: if $\text{atoms}(x) \subsetneq \text{atoms}(y)$ holds and x is in the head of Q, then y is also in the head of Q.

For brevity, we denote by CQ, ACQ, CCQ, HCQ, and QHCQ the classes of conjunctive queries in general and those conjunctive queries that are acyclic, free-connex acyclic, hierarchical or q-hierarchical, respectively. We note that each q-hierarchical query is free-connex acyclic [20, Proposition 4.25] and each hierarchical query is acyclic.[1]

Views and rewritings. A view V over a schema $\mathcal{S}$ is just a query over schema $\mathcal{S}$.[2] A set $\mathcal{V} = \{V_1, \dots, V_k\}$ of views induces, for each database D over schema $\mathcal{S}$, the $\mathcal{V}$-defined *database* $\mathcal{V}(D) = V_1(D) \cup \cdots \cup V_k(D)$.

In this paper, we consider only views that are conjunctive queries. We furthermore assume that every view defines its own relation in the derived database. We denote the set of relation names of the heads of the views in $\mathcal{V}$ by $\mathcal{S}_{\mathcal{V}}$.

In a nutshell, a rewriting of a query Q over a set $\mathcal{V}$ of views is a query over $\mathcal{S}_{\mathcal{V}}$ that is meant to yield, for each database D, the same result over $\mathcal{V}(D)$ as Q does over D.

[1] The latter inclusion is mentioned in, e.g. [19, 21]; it also follows readily from the former inclusion. For acyclicity the head of a query is of no concern, and the Boolean variant of a hierarchical query is always q-hierarchical by definition, and, thus, free-connex acyclic which implies the existence of a join tree for the body of the original query.
[2] They are called views due to their special role which distinguishes them from "normal" queries.

▶ **Definition 2.2.** Let Q be a query, $\mathcal{V}$ a set of views, and Q' a query over $\mathcal{S}_{\mathcal{V}}$. We call Q' a *$\mathcal{V}$-rewriting of Q* if, for every database D, it holds that $Q'(\mathcal{V}(D)) = Q(D)$.

3 The Rewriting Problem

In this section, we define the rewriting problem, recall some known results about its complexity and discuss some relevant concepts that were used to tackle it.

▶ **Definition 3.1.** Let $\mathbb{V}$, $\mathbb{Q}$, and $\mathbb{R}$ be classes of conjunctive queries. The *rewriting problem* for $\mathbb{V}$, $\mathbb{Q}$, and $\mathbb{R}$, denoted $\text{REWR}(\mathbb{V}, \mathbb{Q}, \mathbb{R})$ asks, upon input of a query $Q \in \mathbb{Q}$ and a (finite) set $\mathcal{V} \subseteq \mathbb{V}$ of views, whether there is a $\mathcal{V}$-rewriting of Q in the class $\mathbb{R}$. We write $\text{REWR}^k(\mathbb{V}, \mathbb{Q}, \mathbb{R})$ for the restriction, where the arity of the database schema is bounded by k.

In general, the rewriting problem for conjunctive queries is NP-complete.

▶ **Theorem 3.2** ([25, Theorem 3.10]). *$\text{REWR}(\mathsf{CQ}, \mathsf{CQ}, \mathsf{CQ})$ is NP-complete.*

There is a straightforward, albeit in general inefficient, way of finding a rewriting for a conjunctive query and views. In fact, it can be shown that, for given Q and $\mathcal{V}$, a rewriting Q' over $\mathcal{V}$ exists, if and only if a certain canonical query $\mathsf{canon}(Q, \mathcal{V})$ exists and is a rewriting.

This *canonical candidate* $\mathsf{canon}(Q, \mathcal{V})$ can be obtained by evaluating $\mathcal{V}$ on the canonical database $\mathsf{canon}(Q)$, which in turn is defined as the set of body atoms of Q viewed as facts, in which the variables are considered as fresh constants. The canonical candidate has the same head as Q and its body is just $\mathcal{V}(\mathsf{canon}(Q))$.[3] If the canonical candidate turns out to be a rewriting, then we often call it *canonical rewriting*.

An important detail should be mentioned here: the canonical candidate does not always exist, e.g., if $\mathcal{V}(\mathsf{canon}(Q))$ does not contain all head variables of Q or it is outright empty. In that case, there is no rewriting.

▶ **Proposition 3.3** ([26, Proposition 5.1]). *Let Q be a CQ and $\mathcal{V}$ a set of views. If there is a $\mathcal{V}$-rewriting of Q, then the canonical candidate $\mathsf{canon}(Q, \mathcal{V})$ is such a rewriting.*

▶ **Example 3.4.** Let us consider the query Q defined by

$$H(x, y, z) \leftarrow C(x, y, z),\ R(x, y),\ S(y, z),\ T(z, x)$$

and the views

$$V_1(u, v, w) \leftarrow C(u, v, w) \quad \text{and} \quad V_2(x, y, z, u) \leftarrow R(x, y),\ S(y, z),\ T(z, u).$$

Evaluating the views V_1 and V_2 on the canonical database $\mathsf{canon}(Q)$ of Q yields the result $\{V_1(x, y, z), V_2(x, y, z, x)\}$. Thus, the query Q' defined by

$$H(x, y, z) \leftarrow V_1(x, y, z), V_2(x, y, z, x)$$

is the canonical candidate, which happens to be an actual $\mathcal{V}$-rewriting.

We will need the following notions throughout the paper.

▶ **Definition 3.5** (View Application). An *application* of a view V is a substitution $\alpha\colon \text{vars}(V) \to \mathsf{var}$ that does not unify any quantified variable x of V with another variable of V.

[3] But here the variables are again considered as variables.

We say that a collection $\alpha_1, \ldots, \alpha_m$ of applications for views $V_1, \ldots, V_m$ fulfils *quantified variable disjointness*, if for all $i, j \in \{1, \ldots, m\}$, each quantified variable x of V_i and each variable y of V_j with $i \neq j$, it holds $\alpha_i(x) \neq \alpha_j(y)$. This ensures that, for all $i, j \in \{1, \ldots, m\}$ with $i \neq j$, the bodies $\alpha_i(\text{body}(V_i))$ and $\alpha_j(\text{body}(V_j))$ do not share any variables that do not occur in $\text{vars}(\alpha_i(\text{head}(V_i))) \cap \text{vars}(\alpha_j(\text{head}(V_j)))$.

▶ **Definition 3.6** (Expansion)**.** Let $\mathcal{V}$ be a set of views over a schema $\mathcal{S}$ and let Q' be a query with $\text{body}(Q') = \{A'_1, \ldots, A'_m\}$ over the schema $\mathcal{S}_{\mathcal{V}}$. Furthermore, let, for each $i \in \{1, \ldots, m\}$, α_i be an application and V_i be a view such that $A'_i = \alpha_i(\text{head}(V_i))$, such that $\alpha_1, \ldots, \alpha_m$ fulfil quantified variable disjointness.

The *expansion of Q' w.r.t. $\mathcal{V}$ and $\alpha_1, \ldots, \alpha_m$* is the query that has the same head as Q' and body $\bigcup_{i=1}^{m} \alpha_i(\text{body}(V_i))$.

Since the applications $\alpha_1, \ldots, \alpha_m$ in Definition 3.6 are uniquely determined up to renaming of quantified variables, we usually do not mention them explicitly and just speak of an expansion of Q' w.r.t. $\mathcal{V}$.

We note that the expansion of Q' can be directly compared with Q since it is over the same schema. In fact, Q' is a $\mathcal{V}$-rewriting of Q if and only if the expansion is equivalent to Q. In Example 3.4, this is obviously the case, since (the only) expansion of Q' w.r.t. $\mathcal{V}$ even coincides with Q.

Even though the canonical candidate can be of exponential size in $|Q| + |\mathcal{V}|$, there is always a rewriting of polynomial size if a rewriting exists. In fact, at most n view applications are needed if the body of query Q contains n atoms [25, Lemma 3.5]. Therefore, and because equivalence of CQ can be tested in NP, $\text{Rewr}(\mathsf{CQ}, \mathsf{CQ}, \mathsf{CQ})$ is in NP.

However, for acyclic views of bounded arity and acyclic queries, the situation is much better (cf. the full version [15] for a short proof). For a class $\mathbb{V}$ of views, we write $\mathbb{V}^k$ for the restriction to views of arity at most k.

▶ **Proposition 3.7.** *For every $k \geq 0$, $\text{Rewr}(\mathsf{ACQ}^k, \mathsf{ACQ}, \mathsf{CQ})$ is in polynomial time.*

Chekuri and Rajaraman have shown that the rewriting problem is in P for acyclic queries without self-joins and arbitrary views using a similar idea [10, Theorem 5].

However, even for Boolean views and databases over a small fixed schema, it does not suffice to restrict only the query or only the views to acyclic queries.

▶ **Proposition 3.8.** $\text{Rewr}^k(\mathsf{CQ}^0, \mathsf{ACQ}, \mathsf{CQ})$ *and* $\text{Rewr}^k(\mathsf{ACQ}^0, \mathsf{CQ}, \mathsf{CQ})$ *are* NP*-hard, for every $k \geq 3$. This even holds for Boolean queries, view sets with only one Boolean view and a schema with two relations of maximum arity 3.*

The proof is given in the full version [15].

4 A Characterisation

In this section we give a characterisation of rewritability of a query Q with respect to a set $\mathcal{V}$ of views. It is in terms of a partition of the atoms of $\text{body}(Q)$, where each set of the partition can be matched with a view in a specific way. We refer to such partitions as *cover partitions* and the matches as *cover descriptions*. The characterisation is very similar to other such characterisations in the literature [17, 2], in particular to "MiniCon Descriptions" [28]. However, in its specific form and notation it is tailored for our needs in the subsequent sections. We will discuss the relationship of our characterisation with others further below.

Next we define the notions of *cover descriptions* and *cover partitions*. Let Q be a conjunctive query. For a set $\mathcal{A} \subseteq \text{body}(Q)$ we define $\text{bvars}_Q(\mathcal{A})$ as the set of *bridge variables* of $\mathcal{A}$, that occur in $\mathcal{A}$ as well as in the head of Q or in some atom of Q not in $\mathcal{A}$. More formally, $\text{bvars}_Q(\mathcal{A}) = \text{vars}(\mathcal{A}) \cap (\text{vars}(\text{head}(Q)) \cup \text{vars}(\text{body}(Q) \setminus \mathcal{A}))$.

▶ **Definition 4.1** (Cover Description). A *cover description* C for a query Q is a tuple $(\mathcal{A}, V, \alpha, \psi)$ where

- $\mathcal{A}$ is a subset of $\text{body}(Q)$,
- V is a view,
- α is an application of V, and
- ψ is a mapping from $\text{vars}(\alpha(V))$ to $\text{vars}(Q)$,

such that

(1) $\mathcal{A} \subseteq \alpha(\text{body}(V))$,
(2) $\text{bvars}_Q(\mathcal{A}) \subseteq \alpha(\text{vars}(\text{head}(V)))$,
(3) ψ is a body homomorphism from $\alpha(V)$ to Q, and
(4) ψ is the identity on $\text{vars}(\mathcal{A})$.

▶ **Example 4.2.** Consider the query Q given by the rule

$$H(x, y, z) \leftarrow R(x, y, z), T(x, v), F(v), E(w), S(w, z)$$

and the following views.

$$\begin{aligned} V_1(x_1, y_1, w_1) &\leftarrow R(x_1, y_1, u_1), T(x_1, v_1), F(v_1), E(w_1), S(w_1, u_1) \\ V_2(x_2, y_2, z_2) &\leftarrow R(x_2, y_2, z_2), F(v_2) \\ V_3(w_3, z_3) &\leftarrow S(w_3, z_3), E(w_3) \end{aligned}$$

The tuple $C_1 = (\mathcal{A}_1, V_1, \alpha_1, \psi_1)$ with $\mathcal{A}_1 = \{T(x, v), F(v)\}$,

$$\begin{aligned} &\alpha_1 = \{x_1 \mapsto x, y_1 \mapsto y', u_1 \mapsto u', v_1 \mapsto v, w_1 \mapsto w'\}, \text{ and} \\ &\psi_1 = \{x \mapsto x, y' \mapsto y, u' \mapsto z, v \mapsto v, w' \mapsto w\} \end{aligned}$$

is a cover description for Q with $\text{bvars}_Q(\mathcal{A}_1) = \{x\}$. Although ψ_1 could be chosen as id (by adapting α_1 accordingly), we will see in Example 4.4, that this is not always desirable.

Now we can simply characterise rewritability of a query Q by the existence of a partition of $\text{body}(Q)$ whose subsets have cover descriptions with views from $\mathcal{V}$.

▶ **Definition 4.3.** Let Q be a query and $\mathcal{V}$ be a set of views. A collection $\mathcal{C} = C_1, \ldots, C_m$ of cover descriptions $C_i = (\mathcal{A}_i, V_i, \alpha_i, \psi_i)$ for Q with $V_i \in \mathcal{V}$ is a *cover partition* for Q over $\mathcal{V}$ if the sets $\mathcal{A}_1, \ldots, \mathcal{A}_m$ constitute a partition of $\text{body}(Q)$.

We call a cover partition *consistent* if variables of any $\alpha_j(V_j)$ are in the range of any other α_i, only if they also appear in $\text{bvars}_Q(\mathcal{A}_j)$. We note that, since each α_i is a view application, in a consistent cover partition, the applications obey quantified variable disjointness.[4]

▶ **Example 4.4** (Continuation of Example 4.2). Let $C_1 = (\mathcal{A}_1, V_1, \alpha_1, \psi_1)$ be the cover description defined in Example 4.2. In addition, we consider the cover descriptions C_2 and C_3 with $C_i = (\mathcal{A}_i, V_i, \alpha_i, \psi_i)$ for $i \in \{2, 3\}$ where

$$\begin{aligned} &\mathcal{A}_2 = \{R(x, y, z)\}, && \mathcal{A}_3 = \{E(w), S(w, z)\}, \\ &\alpha_2 = \{x_2 \mapsto x, y_2 \mapsto y, z_2 \mapsto z, v_2 \mapsto v\}, && \alpha_3 = \{w_3 \mapsto w, z_3 \mapsto z\}, \end{aligned}$$

[4] For the sake of contradiction, assume two quantified variables x and y are mapped to the same variable z by two different view applications α_i and α_j, respectively. Then, due to consistency, z is in $\text{bvars}(\mathcal{A}_i)$ and, thus, in $\alpha_i(\text{head}(V_i))$ due to Definition 4.1(2). It follows that α_i unifies x with a head variable. But this is a contradiction to α_i being a view application.

and $\psi_2 = \psi_3 = \text{id}$. The cover descriptions C_1, C_2, C_3 constitute a cover partition for Q over $\{V_1, V_2, V_3\}$. It is, however, not consistent, since v is in the range of α_1 and α_2, but not in $\text{bvars}(\mathcal{A}_1)$. Replacing α_2 and ψ_2 by mappings $\hat{\alpha}_2$ and $\hat{\psi}_2$ with $\hat{\alpha}_2(v_2) = \hat{v}$ and $\hat{\psi}_2(\hat{v}) = v$ that agree with α_2 and ψ_2 on all other variables, respectively, yields a consistent cover partition. Note that there is no consistent cover partition of the form $(\mathcal{A}_1, V_1, \alpha'_1, \text{id})$ because necessarily $\alpha'_1(w_1) = w$ would hold and thus w would be in the range of α'_1 and α_3.

A consistent cover partition $\mathcal{C}$ induces a query $Q_\mathcal{C}$ and an expansion $Q'_\mathcal{C}$ as follows. We note first that each variable in $\text{head}(Q)$ occurs in at least one of the sets $\mathcal{A}_i$ and thus in some set $\text{bvars}_Q(\mathcal{A}_i)$. Therefore, Condition (2) guarantees that each head variable of Q occurs in some set $\alpha_i(\text{head}(V_i))$ and thus a cover partition $\mathcal{C} = C_1, \dots, C_m$ with $C_i = (\mathcal{A}_i, V_i, \alpha_i, \psi_i)$ induces a query $Q_\mathcal{C}$ with

$$\text{head}(Q_\mathcal{C}) = \text{head}(Q) \text{ and } \text{body}(Q_\mathcal{C}) = \{\alpha_i(\text{head}(V_i)) \mid 1 \leq i \leq m\}.$$

Likewise, and thanks to quantified variable disjointness, it induces an expansion $Q'_\mathcal{C}$ with

$$\text{head}(Q'_\mathcal{C}) = \text{head}(Q) \text{ and } \text{body}(Q'_\mathcal{C}) = \{\alpha_i(\text{body}(V_i)) \mid 1 \leq i \leq m\}.$$

Next, we show that the existence of a cover partition indeed characterises rewritability.

▶ **Theorem 4.5.** *Let Q be a minimal query and $\mathcal{V}$ be a set of views. The following three statements are equivalent.*
(a) *Q is $\mathcal{V}$-rewritable.*
(b) *There is a cover partition $\mathcal{C}$ for Q over $\mathcal{V}$.*
(c) *There is a consistent cover partition $\mathcal{C}$ for Q over $\mathcal{V}$.*
If $\mathcal{C}$ is a consistent cover partition, then $Q_\mathcal{C}$ is a $\mathcal{V}$-rewriting of Q.

Proof ideas. To show that (c) implies (a) we consider a consistent cover partition $\mathcal{C} = C_1, \dots, C_m$ for Q over $\mathcal{V}$, where, for each i, $C_i = (\mathcal{A}_i, V_i, \alpha_i, \psi_i)$. We prove that the query $Q_\mathcal{C}$ induced by $\mathcal{C}$ is a $\mathcal{V}$-rewriting of Q: first, since the sets $\mathcal{A}_1, \dots, \mathcal{A}_m$ partition $\text{body}(Q)$ and thanks to Condition (1), the identity mapping is a homomorphism from Q into Q'_C. Therefore, it suffices to show that the mapping h' from $Q'_\mathcal{C}$ into Q, defined as the union of the mappings ψ_i, is a homomorphism.

To show that (a) implies (b) we assume that Q has a $\mathcal{V}$-rewriting Q_R with an expansion Q_E. We show that we can assume that the body of Q is a subset of Q_R and that there is a reverse homomorphism h' which is the identity on all variables of Q. That is, we have $\text{head}(Q) = \text{head}(Q_E) = \text{head}(Q_R)$ as well as $\text{body}(Q) \subseteq \text{body}(Q_E)$.

Let $\alpha_1, \dots, \alpha_m$ be applications of views $V_1, \dots, V_m$ that result in expansion Q_E, that is, $\text{body}(Q_E) = \bigcup_{i=1}^m \text{body}(\alpha_i(V_i))$, where the applications fulfil the quantified variable disjointness property. These applications induce a partition $\mathcal{A}_1, \dots, \mathcal{A}_m$ of $\text{body}(Q)$ via $\mathcal{A}_i = \{A \in \text{body}(Q) \mid i \text{ is minimal such that } A \in \alpha_i(\text{body}(V_i))\}$. We show then that this partition induces a cover partition.

Finally, we show that (b) implies (c) by renaming some variables. We note that this step does not change the underlying partition of $\text{body}(Q)$. ◀

The full proof is given in the full version [15].

▶ Remark 4.6. We note that the requirement in Theorem 4.5 for Q to be minimal is only seemingly a restriction in our setting, since we apply it only to acyclic queries. If Q is acyclic but not minimal, an equivalent minimal query Q' can be computed in polynomial time by iteratively removing atoms from its body [9, 10]. Moreover, it is guaranteed that the minimal query Q' is also acyclic (we believe this to be folklore, it follows readily from more general results, cf. for instance [5]).

The same is true for free-connex acyclic queries: every homomorphism from Q to Q' is also a homomorphism from $\text{body}(Q) \cup \{\text{head}(Q)\}$ to $\text{body}(Q') \cup \{\text{head}(Q')\}$ and vice versa, since the relation symbol of $\text{head}(Q) = \text{head}(Q')$ does not occur in $\text{body}(Q')$. Thus, Q' is minimal if and only if the Boolean query whose body is $\text{body}(Q') \cup \{\text{head}(Q')\}$ is minimal. Therefore, if $\text{body}(Q) \cup \{\text{head}(Q)\}$ is acyclic, so is $\text{body}(Q') \cup \{\text{head}(Q')\}$. In other words, if Q is free-connex acyclic, then Q' is free-connex acyclic as well.

For hierarchical and q-hierarchical queries the same holds: it is easy to see that removing atoms does not change the conditions in their respective definitions.

5 Towards Acyclic Rewritings

In this section, we turn our focus to the main topic of this paper: acyclic rewritings and the decision problem that asks whether such a rewriting exists. We study the complexity of $\textsc{Rewr}(\mathbb{V}, \mathbb{Q}, \mathsf{ACQ})$ for the case that $\mathbb{V}$ and $\mathbb{Q}$ are the class of conjunctive queries and for various subclasses. It will be helpful to analyse the case that $\mathbb{Q} = \mathsf{ACQ}$ and $\mathbb{V} = \mathsf{CQ}$ first.

5.1 A Characterisation of Acyclic Rewritability for Acyclic Queries

The following example illustrates that, even if an acyclic rewriting exists, the canonical rewriting need not be acyclic. Furthermore, it may be that each "sub-rewriting" of the canonical candidate is cyclic or not a rewriting, and thus none of them is an acyclic rewriting.

▶ **Example 5.1.** Consider the query Q given by the rule

$$H() \leftarrow R_1(x), R_2(y), S(x,z), T_1(z), T_2(y)$$

and the following views $\mathcal{V} = \{V_1, V_2, V_3\}$.

$$V_1(u,v) \leftarrow R_1(u), R_2(v) \qquad V_2(u,v) \leftarrow S(u,v) \qquad V_3(u,v) \leftarrow T_1(u), T_2(v)$$

The canonical candidate $Q_R = H() \leftarrow V_1(x,y), V_2(x,z), V_3(z,y)$ is a $\mathcal{V}$-rewriting, but it is *cyclic*. Each query whose body is a proper subset of the body of Q_R is not a $\mathcal{V}$-rewriting for Q. However, an *acyclic* $\mathcal{V}$-rewriting for Q exists. One such rewriting is the query Q'_R.

$$H() \leftarrow V_1(x,y), V_2(x,z), V_3(z,y'), V_3(z',y)$$

Even though the example suggests that general and acyclic rewritings can be seemingly unrelated, it turns out that there is a close connection between them. In fact, we will show next that an acyclic $\mathcal{V}$-rewriting exists if and only if an arbitrary $\mathcal{V}$-rewriting exists. Furthermore, from an arbitrary rewriting Q_R, an acyclic one Q_C can be constructed.

Towards a proof, we study the relationship between the target query Q and the view applications that can occur in any rewriting of Q more closely, to determine the circumstances under which a decomposition of view atoms is possible.

▶ **Example 5.2** (Continuation of Example 5.1). In the SPJ-algebra[5], the V_3-atom in Q_R can be expressed as the SPJ-expression $\pi_{z,y}(V_3(z,y))$, whereas the two V_3-atoms in Q'_R can be expressed as $\pi_{z,y}(V_3(z,y') \bowtie V_3(z',y))$. When expanded by the SPJ-expressions that define

[5] SPJ-algebra refers to the select-project-join-algebra.

V_3, these two expressions are equivalent. That is, the acyclic rewriting can be obtained from the canonical rewriting by replacing an atom by a set of atoms that is equivalent with respect to view expansions.[6]

In Example 5.1 the acyclic rewriting was obtained from the canonical rewriting by replacing a view atom by a set that contained one view atom for each connected component of V_3. The following example shows that the required modifications can be more involved.

▶ **Example 5.3.** Consider the query Q given by the rule

$$H(x, y, z) \leftarrow R(x, y, z), E_1(x), E_2(y), E_3(w), S(w, z)$$

and the following views.

$$V_1(x, y, w) \leftarrow R(x, y, v), E_1(x), E_3(w), S(w, v) \qquad V_2(x, y, z) \leftarrow R(x, y, z), E_2(y)$$
$$V_3(w, z) \leftarrow S(w, z), E_3(w)$$

The canonical candidate $H(x, y, z) \leftarrow V_1(x, y, w), V_2(x, y, z), V_3(w, z)$ is a cyclic $\{V_1, V_2, V_3\}$-rewriting of Q. In contrast to Example 5.1, all view bodies are connected. Therefore, replacing a view atom by a set of representatives of connected components of some views does not yield an acyclic rewriting. However, there is an acyclic $\{V_1, V_2, V_3\}$-rewriting of Q, namely $H(x, y, z) \leftarrow V_1(x, y', w'), V_2(x, y, z), V_3(w, z)$.

Now we turn to the main result of this section. Its proof is given in the full version [15].

▶ **Theorem 5.4.** *Let Q be a (free-connex) acyclic conjunctive query and $\mathcal{V}$ a set of* CQ*-views. If Q is $\mathcal{V}$-rewritable, then there is an (free-connex) acyclic $\mathcal{V}$-rewriting of Q.*

The proof shows that a given cover partition for Q can be modified such that each of its cover descriptions induces a connected sub-tree of a join tree of Q. Then it is shown that by collapsing these sub-trees into nodes, a join tree for the rewriting induced by the modified cover partition is obtained.

Theorem 5.4 delivers good news as well as bad news. The good news is that, since the proofs of Theorem 5.4 and Theorem 4.5 are constructive, we altogether have a procedure to construct an acyclic rewriting from an arbitrary rewriting Q_R if the query Q is acyclic. Furthermore, if a homomorphism from an expansion of Q_R to Q can be computed in polynomial time, the overall procedure can be performed in polynomial time. In particular, we get the following collary.

▶ **Corollary 5.5.** *For every k,* Rewr(ACQ^k, ACQ, ACQ) *is in polynomial time and an acyclic rewriting can be computed in polynomial time (if it exists).*

That the decision problem is in polynomial time follows immediately from Proposition 3.7 and Theorem 5.4. Moreover, the canonical candidate for an acyclic query Q and a set $\mathcal{V}$ of acyclic views with bounded arity can be computed in polynomial time, since the size of $\mathcal{V}(\mathsf{canon}(Q))$ is bounded by a polynomial and query evaluation for acyclic queries is in polynomial time [30, 1]. In case the canonical candidate is a rewriting, valuations witnessing $\mathcal{V}(\mathsf{canon}(Q))$ can be computed in polynomial time [23] and combined into a homomorphism from an expansion of the rewriting to Q. Thus, an acyclic rewriting can then be efficiently constructed by the above procedure.

[6] We note that one could also replace the V_1-atom in Q_R.

However, this leaves open the complexity of $\textsc{Rewr}(\mathsf{ACQ}, \mathsf{ACQ}, \mathsf{ACQ})$ (as well as that of $\textsc{Rewr}(\mathsf{ACQ}, \mathsf{ACQ}, \mathsf{CQ})$), and of their restrictions to schemas of bounded arity. The bad news is that, since $\textsc{Rewr}(\mathbb{V}, \mathsf{ACQ}, \mathsf{CQ})$ and $\textsc{Rewr}(\mathbb{V}, \mathsf{ACQ}, \mathsf{ACQ})$ are basically the same problem, lower bounds on $\textsc{Rewr}(\mathbb{V}, \mathsf{ACQ}, \mathsf{CQ})$ transfer to $\textsc{Rewr}(\mathbb{V}, \mathsf{ACQ}, \mathsf{ACQ})$. In fact, we get the following, due to NP-hardness of $\textsc{Rewr}^k(\mathsf{CQ}, \mathsf{ACQ}, \mathsf{CQ})$ in the general case (Proposition 3.8).[7]

▶ **Corollary 5.6.** *For every $k \geq 3$, the problem $\textsc{Rewr}^k(\mathsf{CQ}, \mathsf{ACQ}, \mathsf{ACQ})$ and, therefore, also $\textsc{Rewr}^k(\mathsf{CQ}, \mathsf{CQ}, \mathsf{ACQ})$ is* NP*-hard.*

In the next part of this section, we resolve the complexity of $\textsc{Rewr}(\mathsf{ACQ}, \mathsf{ACQ}, \mathsf{CQ})$ and $\textsc{Rewr}(\mathsf{ACQ}, \mathsf{ACQ}, \mathsf{ACQ})$ and their restrictions to schemas of bounded arity.

5.2 The Complexity of Acyclic Rewritability for Acyclic Queries

It may be tempting to assume that, since acyclic queries are so well-behaved in general, it should be tractable to decide whether for an acyclic query and a set of acyclic views there exists an acyclic rewriting. However, as we show next, this is (probably) not the case, and this surprising finding even holds for the even better behaved hierarchical queries as well.

▶ **Theorem 5.7.** *The problems $\textsc{Rewr}^k(\mathsf{ACQ}, \mathsf{ACQ}, \mathsf{CQ})$ and $\textsc{Rewr}^k(\mathsf{ACQ}, \mathsf{ACQ}, \mathsf{ACQ})$, as well as $\textsc{Rewr}^k(\mathsf{HCQ}, \mathsf{HCQ}, \mathsf{CQ})$ are* NP*-complete, for $k \geq 3$.*[8] *The lower bounds even hold for instances with a single view.*

We will see in Section 7 that Theorem 5.4 has an analogue for hierarchical queries from which it can be concluded that deciding the existence of a hierarchical rewriting, given a hierarchical query and hierarchical views, is still NP-complete.

Theorem 5.7 will easily follow from NP-hardness of a seemingly simpler problem. From the characterisation in Theorem 4.5, we already know that deciding the existence of a rewriting is the same as deciding the existence of a cover partition. We show next that it is even NP-hard to decide the existence of a *cover description*, given a query, a set of atoms, and a single view.

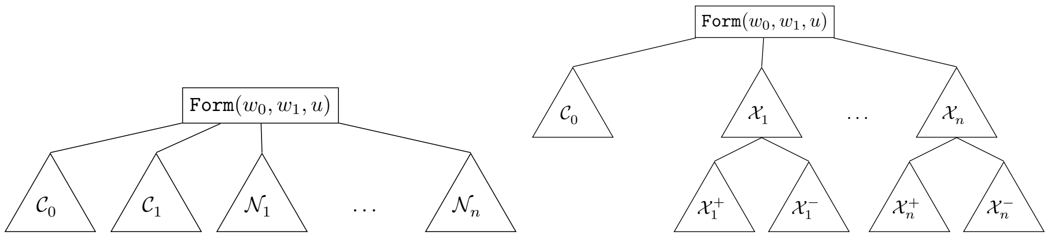
▶ **Definition 5.8.** Let $\mathbb{V} \subseteq \mathsf{CQ}$ and $\mathbb{Q} \subseteq \mathsf{CQ}$ be classes of conjunctive queries. The *cover description problem* for $\mathbb{V}$ and $\mathbb{Q}$, denoted $\textsc{CovDesc}(\mathbb{V}, \mathbb{Q})$ asks, upon input of a query $Q \in \mathbb{Q}$, a subset $\mathcal{A} \subseteq \text{body}(Q)$ and a view $V \in \mathbb{V}$, whether mappings α and ψ exist such that $(\mathcal{A}, V, \alpha, \psi)$ is a cover description.

▶ **Theorem 5.9.** $\textsc{CovDesc}(\mathsf{ACQ}, \mathsf{ACQ})$ *is* NP*-hard. Indeed, even* $\textsc{CovDesc}(\mathsf{HCQ}, \mathsf{HCQ})$ *is* NP*-hard and even if the input is restricted to $\mathcal{A} = \text{body}(Q)$.*

Proof. We reduce problem 3SAT to $\textsc{CovDesc}(\mathsf{HCQ}, \mathsf{HCQ})$. This lower bound directly translates to $\textsc{CovDesc}(\mathsf{ACQ}, \mathsf{ACQ})$ since every hierarchical query is acyclic. For a formula f in 3CNF, we describe how a query Q and a view V can be derived in polynomial time such that both Q and V are hierarchical and such that f is satisfiable if and only if there are mappings α and ψ such that $(\text{body}(Q), V, \alpha, \psi)$ is a cover description for Q.

[7] We note that Corollary 5.6 also follows from the proof of Proposition 3.8, since the canonical candidate constructed in that proof is trivially acyclic. Indeed, NP-hardness of $\textsc{Rewr}(\mathsf{ACQ}, \mathsf{CQ}, \mathsf{ACQ})$ is also implied.

[8] These results hold for schemas of unbounded arity, as well.



(a) Structure of query $\boldsymbol{Q}$, induced by formula $\boldsymbol{f}$. **(b)** Structure of view $\boldsymbol{V}$, induced by formula $\boldsymbol{f}$.

Figure 1 Structure of the query $\boldsymbol{Q}$ and the view $\boldsymbol{V}$, induced by formula $\boldsymbol{f}$.

Construction. Let $f = f_1 \wedge \cdots \wedge f_k$ be a propositional formula in 3CNF over variables $x_1, \ldots, x_n$ in clauses $f_1, \ldots, f_k$, where $f_j = (\ell_{j,1} \vee \ell_{j,2} \vee \ell_{j,3})$ for each $j \in \{1, \ldots, k\}$.

We start with the construction of query Q. This query is Boolean, with head $H()$, and uses only three variables w_0, w_1 and u, where the first two are intended to represent the truth values false and true. The structure of Q is depicted in Figure 1a. The body of Q is defined as the union

$$\{\mathtt{Form}(w_0, w_1, u)\} \uplus \mathcal{C}_0 \uplus \mathcal{C}_1 \uplus \mathcal{N}_1 \uplus \cdots \uplus \mathcal{N}_n$$

of the following sets of atoms.

- Set $\mathcal{C}_0$ contains an atom $\mathtt{C}_j(w_0, w_1, u)$ for each clause f_j in f.

  Intuitively, these atoms represent *unsatisfied* clauses as opposed to the next atoms that represent *satisfied* clauses. Note that these atoms differ in the order of variables w_0 and w_1 only.
- Set $\mathcal{C}_1$ contains an atom $\mathtt{C}_j(w_1, w_0, u)$ for each clause f_j in f.
- There is a set $\mathcal{N}_i$ for each variable x_i in formula f. The sets differ only in the name of the relation they address and contain two atoms $\mathtt{Neg}_i(w_0, w_1, u)$ and $\mathtt{Neg}_i(w_1, w_0, u)$ each.

Query Q is hierarchical since every atom in each of the sets above refers to all three variables w_0, w_1 and u. Thus, query Q is acyclic in particular.

We now proceed with the construction of view V, which refers to the same relations as query Q. Like query Q, the view uses variables w_0, w_1 and u but also additional variables for the propositions $x_1, \ldots, x_n$ in formula f. For each proposition x_i, there are two variables x_i and $\bar{x}_i$, intended to represent the positive and negated literal over the proposition, respectively. With the exception of variable u, all other variables are in the head $H(w_0, w_1, x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n)$ of view V. The body of V, whose structure is depicted in Figure 1b, is defined as the union

$$\{\mathtt{Form}(w_0, w_1, u)\} \uplus \mathcal{C}_0 \uplus \left(\mathcal{X}_1 \uplus \mathcal{X}_1^+ \uplus \mathcal{X}_1^-\right) \uplus \cdots \uplus \left(\mathcal{X}_n \uplus \mathcal{X}_n^+ \uplus \mathcal{X}_n^-\right)$$

of sets of atoms, where the first two sets $\{\mathtt{Form}(w_0, w_1, u)\}$ and $\mathcal{C}_0$ are the same as above. The bodies of query Q and view V thus differ in the sets $\mathcal{C}_1$ and $\mathcal{N}_1, \ldots, \mathcal{N}_n$ of atoms on the one hand as well as $\mathcal{X}_1, \ldots, \mathcal{X}_n$ and $\mathcal{X}_1^+, \ldots, \mathcal{X}_n^+$ and $\mathcal{X}_1^-, \ldots, \mathcal{X}_n^-$ on the other hand. Sets $\mathcal{N}_1, \ldots, \mathcal{N}_n$ and $\mathcal{X}_1, \ldots, \mathcal{X}_n$ are intended to guide the assignment of propositions while sets $\mathcal{X}_1^+, \ldots, \mathcal{X}_n^+$ and $\mathcal{X}_1^-, \ldots, \mathcal{X}_n^-$ represent the occurrences of literals in the clauses of formula f and set $\mathcal{C}_1$ represents the aim to satisfy all of them. The new sets are defined as follows.

- For each proposition x_i, set $\mathcal{X}_i$ contains atoms $\mathtt{Neg}_i(x_i, \bar{x}_i, u)$ and $\mathtt{Neg}_i(\bar{x}_i, x_i, u)$, intended to enforce the mapping of $(x_i, \bar{x}_i)$ to either (w_0, w_1) or (w_1, w_0), that is, to "complementary truth values".

- For each positive literal x_i and each clause f_j that contains literal x_i, set $\mathcal{X}_i^+$ contains an atom $\mathtt{C}_j(x_i, \bar{x}_i, u)$.
- For each negated literal $\bar{x}_i$ and each clause f_j that contains literal $\neg x_i$, set $\mathcal{X}_i^-$ contains an atom $\mathtt{C}_j(\bar{x}_i, x_i, u)$.

Thus defined, the view is also hierarchical as shown in the following. Every atom in every set contains variable u. Therefore, it follows that $\text{atoms}(x) \subseteq \text{atoms}(u)$ for all $x \in \text{vars}(V)$. Furthermore, all atoms in $\{\mathtt{Form}(w_0, w_1, u)\}$ and $\mathcal{C}_0$ additionally contain both variables w_0 and w_1 and they are the only atoms to contain these variables. Thus, we have that $\text{atoms}(w_0) = \text{atoms}(w_1)$ and $\text{atoms}(y) \cap \text{atoms}(w_0) = \emptyset$ and $\text{atoms}(y) \cap \text{atoms}(w_1) = \emptyset$ for all $y \in \text{vars}(V) \setminus \{u, w_0, w_1\}$. Similarly, for each $i \in \{1, \dots, n\}$, all atoms in $\mathcal{X}_i \cup \mathcal{X}_i^+ \cup \mathcal{X}_i^-$ contain both variables x_i and $\bar{x}_i$, which do not occur in any other atom. It follows that $\text{atoms}(x_i) = \text{atoms}(\bar{x}_i)$ and $\text{atoms}(\bar{x}_i) \cap \text{atoms}(y) = \emptyset$ and $\text{atoms}(x_i) \cap \text{atoms}(y) = \emptyset$ for all $y \in \text{vars}(V) \setminus \{x_i, \bar{x}_i, u\}$ and $i \in \{1, \dots, n\}$. Therefore, we obtain that each pair of variables in V satisfies at least one of the conditions in Definition 2.1 and thus V is hierarchical – and acyclic in particular.

For the correctness argument, however, the structure of the join tree of view V, depicted in Figure 1b is probably more helpful. We close the description of the construction by remarking that query and view can be computed in polynomial time for any given propositional formula in 3CNF. The correctness of this construction is shown in the full version [15]. ◀

The proof of Theorem 5.7 can now be stated easily.

Proof of Theorem 5.7. The upper bound follows from Theorem 3.2. For the lower bound, we observe that the proof of Theorem 5.9 shows that query Q has a V-rewriting *if* the formula f is satisfiable. Moreover, the query cannot be covered (non-redundantly) by multiple applications of this view because variable u is no head variable of view V. Thus, we can conclude that Q has a V-rewriting *only if* formula f is satisfiable. Hence, NP-hardness transfers to the rewriting problem. ◀

Notably, the complexity does not result from the restriction to rewritings that are acyclic since the last argument does not depend on the acyclicity.

The fact that, for Theorem 5.7, a single view application is sufficient if there is any rewriting, allows to draw yet another conclusion for a related problem, defined next. In the following, a *selection* is a query that can be expressed by a composition of select-operators.

▶ **Definition 5.10.** Given a query class $\mathbb{Q} \subseteq \mathsf{CQ}$, the *select-full-project equivalence* problem for $\mathbb{Q}$, denoted SelProjEquiv($\mathbb{Q}$), asks, upon input of a Boolean query $Q \in \mathbb{Q}$ and a query $Q' \in \mathbb{Q}$ whether there is a selection σ such that Q is equivalent to $(\pi_\emptyset \circ \sigma)(Q')$ where $\pi_\emptyset$ denotes the projection to the empty set.

The selection σ is reflected[9] in application α of the hardness proof and the projection in the fact that set $\mathcal{A} = \text{body}(Q)$ has no bridge variables. Hence, the following holds.

▶ **Corollary 5.11.** SelProjEquiv(ACQ) *and* SelProjEquiv(HCQ) *are* NP*-hard.*[10]

[9] An application can be decomposed into a selection which unifies head variables and an injective renaming of variables.

[10] We note that the restriction of the select-full-project equivalence problem to Boolean queries Q' is just the equivalence problem for Boolean queries which is in P for acyclic queries [10].

Surprisingly, the select-full-project equivalence problem is NP-hard not only for hierarchical but even for q-hierarchical queries – while the rewriting problem for q-hierarchical views is in P (Corollary 7.4).

▶ **Corollary 5.12.** SelProjEquiv(QHCQ) *is* NP*-hard, even over database schemas with fixed arity.*

The proof idea is explained in the full version [15].

6 A Tractable Case

In this section, we first show that the acyclic rewriting problem becomes tractable for free-connex acyclic views and acyclic queries over database schemas of bounded arity. We then define a slightly larger class of views, for which this statement holds as well. For the first result, we mainly show that rewritability with respect to a set $\mathcal{V}$ of free-connex acyclic views can be reduced to rewritability with respect to a set $\mathcal{W}$ of views of bounded arity.

▶ **Proposition 6.1.** *There is a polynomial-time algorithm that computes from each set $\mathcal{V}$ of free-connex acyclic views a set $\mathcal{W}$ of acyclic*[11] *views such that*

(a) *the arity of the views of $\mathcal{W}$ is bounded by the arity of the underlying schema, and*

(b) *every conjunctive query Q is $\mathcal{V}$-rewritable if and only if it is $\mathcal{W}$-rewritable.*

Furthermore, given a $\mathcal{W}$-rewriting of Q, a $\mathcal{V}$-rewriting of Q can be computed in polynomial time.

The proof splits each view V into a set of views obtained by the subtrees of the root head(V) of a join tree of body$(V) \cup \{$head$(V)\}$. The arities of the children of head(V) yield the desired arity bound. It can be found in the full version [15]. The main result of this section is now a simple corollary to Proposition 6.1.

▶ **Theorem 6.2.** *For every fixed k,* Rewr^k(CCQ, ACQ, ACQ) *is in polynomial time and an acyclic rewriting can be computed in polynomial time, if it exists.*

Proof. Let $Q \in$ ACQ be an acyclic query and $\mathcal{V} \subseteq$ CCQ be a set of free-connex acyclic views over a schema $\mathcal{S}$ with arity at most k. Thanks to Proposition 6.1, from $\mathcal{V}$ an equivalent set $\mathcal{W}$ of acyclic views of arity at most k can be computed in polynomial time. The statements of the theorem thus follow immediately from Corollary 5.5. ◀

We leave the complexity of Rewr(CCQ, ACQ, ACQ) as an open problem.

A closer inspection of the proof of Proposition 6.1 reveals that it does not exactly require that the views are free-connex acyclic. In fact, it suffices that each view has a partition $\mathcal{B}_1 \uplus \cdots \uplus \mathcal{B}_m$ of its body that obeys Conditions (1) and (2) below. Therefore, we turn these two requirements into a new notion. We formulate and study this notion for conjunctive queries Q, but we emphasise that we will use it for queries that define views only.

▶ **Definition 6.3.** The *weak head arity* of a query Q is the smallest k for which there is a partition $\mathcal{B}_1 \uplus \cdots \uplus \mathcal{B}_n$ of body(Q) such that, for every $i \in \{1, \dots, n\}$ and every $j < i$,

(1) $|\text{vars}(\mathcal{B}_i) \cap \text{vars}(\text{head}(Q))| \leq k$, and

(2) $\text{vars}(\mathcal{B}_i) \cap \text{vars}(\mathcal{B}_j) \subseteq \text{head}(Q)$.

[11] In fact, the views in $\mathcal{W}$ are even free-connex acyclic again. However, we do not claim it here, since it is not needed in the following, and it does not need to hold for the subsequent generalisation.

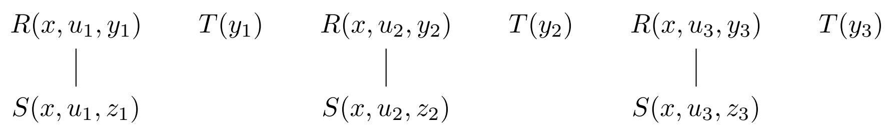


Figure 2 The cover graph $G(V_3)$ of the query V_n for $n = 3$ defined in Example 6.4.

From the proof of Proposition 6.1 it follows that every free-connex acyclic query over a fixed schema has bounded weak head arity. The following example illustrates that there are indeed views over a fixed schema that have bounded weak head arity but are not free-connex acyclic.

▶ **Example 6.4.** Let us consider the family $(V_n)_{n\in\mathbb{N}}$ of views with
- head $V_n(x, y_1, \ldots, y_n, z_1, \ldots, z_n)$, and
- body $\{R(x, u_i, y_i), S(x, u_i, z_i), T(y_i) \mid 1 \leq i \leq n\}$.

For $n \geq 1$ the view V_n is acyclic but *not* free-connex acyclic. It has, however, weak head arity 3. This is witnessed by the sets $\mathcal{B}_i = \{R(x, u_i, y_i), S(x, u_i, z_i), T(y_i)\}$ for $1 \leq i \leq n$ which form a partition satisfying the conditions of Definition 6.3.

To generalise Theorem 6.2 for views of bounded weak head arity, we thus only need to show that partitions obeying Conditions (1) and (2) of Definition 6.3 can be efficiently computed. In the remainder of this section we design an algorithm that determines the weak head arity of a given conjunctive query Q and computes a corresponding partition.

The algorithm relies on the concept of a cover graph for Q.

▶ **Definition 6.5.** The cover graph $G(Q)$ of a conjunctive query Q is the undirected graph with node set body(Q) and edges (A, A') for atoms A and A' that share a variable that does not belong to head(Q).

▶ **Example 6.6.** The cover graph of the view V_3 from Example 6.4 is depicted in Figure 2.

The following lemma states the relationship between weak head arity and cover graph.

▶ **Lemma 6.7.** *Let Q be a conjunctive query and $\mathcal{B}_1, \ldots, \mathcal{B}_n$ the connected components of its cover graph. The weak head arity of Q is the maximal number ℓ of head variables of Q in a set $\mathcal{B}_i$. Moreover, the connected components $\mathcal{B}_1, \ldots, \mathcal{B}_n$ witness that Q has weak head arity ℓ.*

Proof. Let Q be a conjunctive query and let $\mathcal{B}'_1 \uplus \cdots \uplus \mathcal{B}'_m$ be a partition of body(Q) witnessing that Q has weak head arity k.

Since by definition of the cover graph, two different sets $\mathcal{B}_i$ and $\mathcal{B}_j$ only share head variables, $\mathcal{B}_1 \uplus \cdots \uplus \mathcal{B}_n$ is a partition of body(Q) witnessing that Q has weak head arity at most ℓ, hence it holds $k \leq \ell$.

To show that $\ell \leq k$, it suffices to show that every connected component $\mathcal{B}$ in $G(Q)$ is contained in some atom set $\mathcal{B}'_i$. Towards a contradiction we assume that there is a connected component $\mathcal{B}$ with atoms from two different subsets of the partition. Since $\mathcal{B}$ is connected there must be some $A_1, A_2 \in \mathcal{B}$, connected by an edge, such that $A_1 \in \mathcal{B}'_i$ and $A_2 \in \mathcal{B}'_j$, for some $i \neq j$. By definition of $G(Q)$, A_1 and A_2 share a variable that is not part of the head of Q. But that contradicts Condition (2) in Definition 6.3. Thus, we can conclude $\ell \leq k$. ◀

Lemma 6.7 offers an algorithm to compute the weak head arity of a conjunctive query Q and a witness partition for it. It simply computes the cover graph $G(Q)$ of Q and its connected components. Then the weak head arity is the maximum number of head variables that occur in any connected component. Furthermore, the connected components form the desired partition satisfying the conditions of Definition 6.3. We thus have the following corollary.

▶ **Corollary 6.8.** *There is an algorithm that, upon input of a conjunctive query Q, computes in polynomial time the weak head arity of Q and a partition of* $\text{body}(Q)$ *that witnesses it.*

By combining Corollary 6.8 with the proofs of Proposition 6.1 and Theorem 6.2, we obtain the following generalisation of Theorem 6.2.

▶ **Theorem 6.9.** *For each fixed $k \in \mathbb{N}$, the acyclic rewriting problem for acyclic queries and acyclic views with weak head arity k is in polynomial time.*

7 The Existence of Hierarchical and q-hierarchical Rewritings

In Section 5, we have shown that every *acyclic* query has an *acyclic* rewriting if it has a rewriting at all. This gives us a guarantee that there is a rewriting that has the same complexity benefits for query evaluation as the original query.

It is natural to ask whether other, stronger properties transfer in the same fashion. In this section, we consider this question for hierarchical and q-hierarchical queries.

The following example illustrates that, as for acyclic rewritings, even if a hierarchical rewriting for a hierarchical query exists, the canonical rewriting is not necessarily hierarchical.

▶ **Example 7.1.** Consider the hierarchical query $H(x,y) \leftarrow R(x), S(y), T(x), T(y)$ and the views $V_1(x,y) \leftarrow R(x), S(y)$ and $V_2(z) \leftarrow T(z)$. The canonical rewriting leads us to the non-hierarchical rewriting $H(x,y) \leftarrow V_1(x,y), V_2(x), V_2(y)$, since $\text{atoms}(y)$ and $\text{atoms}(z)$ are neither disjoint nor subsets of one another. However, a hierarchical rewriting exists, for instance $H(x,y) \leftarrow V_1(x,y'), V_1(x',y), V_2(x), V_2(y)$ is a hierarchical rewriting.

It turns out that Theorem 5.4 also holds for hierarchical and q-hierarchical queries.

▶ **Theorem 7.2.** *Let $\mathcal{V}$ be a set of views.*
(a) *Every hierarchical query that has some $\mathcal{V}$-rewriting also has a hierarchical $\mathcal{V}$-rewriting.*
(b) *Every q-hierarchical query that has some $\mathcal{V}$-rewriting also has a q-hierarchical $\mathcal{V}$-rewriting.*

Similarly as for Theorem 5.4, the proof of Theorem 7.2 partitions cover descriptions. However, the strategy for doing so is different here: instead of defining the partition as the connected components of an atom set with respect to a join tree of the query at hand, here the partition guaranteed by the following lemma, which is proved in the full version [15], is used.

▶ **Lemma 7.3.** *Let Q be a hierarchical query and $(\mathcal{A}, V, \alpha, \psi)$ a cover description for Q. There is a partition $\mathcal{A}_1 \uplus \cdots \uplus \mathcal{A}_n = \mathcal{A}$ such that the following conditions hold.*
(A) *Each variable $y \notin \text{vars}(\alpha(\text{head}(V)))$ appears in at most one set $\mathcal{A}_i$.*
(B) *Each $\mathcal{A}_i$ is*
  (Ba) *a singleton set or*
  (Bb) *there is a variable $x \notin \text{vars}(\alpha(\text{head}(V)))$ that appears in every atom in $\mathcal{A}_i$.*

Similarly to Theorem 5.4, Theorem 7.2 delivers good news as well as bad news. The good news is that, since $\mathsf{QHCQ} \subseteq \mathsf{CCQ}$ and $\mathsf{QHCQ} \subseteq \mathsf{HCQ} \subseteq \mathsf{ACQ}$ hold, the rewriting problem for q-hierarchical views and hierarchical queries over a fixed schema is tractable thanks to Theorems 6.2 and 7.2. The bad news is that Theorems 5.7 and 7.2 imply NP-completeness for hierarchical queries and views.

▶ **Corollary 7.4.** $\textsc{Rewr}^k(\mathsf{QHCQ}, \mathsf{HCQ}, \mathsf{HCQ})$ *and* $\textsc{Rewr}^k(\mathsf{QHCQ}, \mathsf{QHCQ}, \mathsf{QHCQ})$ *are in polynomial time for every* $k \in \mathbb{N}$.

▶ **Corollary 7.5.** $\textsc{Rewr}^k(\mathsf{HCQ}, \mathsf{HCQ}, \mathsf{HCQ})$ *is* NP*-complete for every* $k \geq 3$.

8 Related Work

We already mentioned that our notion of cover partitions is similar to various notions from the literature. We mention three of them here. A detailed discussion for them is given in the full version [15].

- In [2], algorithms for finding rewritings with a minimal number of atoms and (maximally) contained rewritings[12] are presented. For this purpose triples (S, S', h) are considered which are comparable to cover descriptions. However, the context there is maximal rewritings and no full characterisation of exact rewritability is given.
- In [17], a characterisation for $\mathcal{V}$-rewritability is employed to find rewritings efficiently. It is in terms of *tuple coverages* and a *partition condition* corresponding to cover descriptions and partitions, respectively [17, Theorem 5, Theorem 6]. However, similar to [2], only rewritings whose body is contained in the canonical rewriting are considered.
- In [28], *MiniCon descriptions (MCDs)* are used to compute (maximally) contained rewritings (which are unions of conjunctive queries, in general). They are similar to cover descriptions, but there are several differences in the technical details, as discussed the full version [15].

Minimal cover descriptions

In [2], "minimal" triples are used to construct maximally contained rewritings. In terms of cover descriptions, an analogous definition of minimality is as follows.

▶ **Definition 8.1** (Minimal Cover Descriptions). A cover description $(\mathcal{A}, V, \alpha, \psi)$ for a query Q is called *minimal* if there is no partition $\mathcal{A}_1 \uplus \cdots \uplus \mathcal{A}_k = \mathcal{A}$ for $k \geq 2$ with nonempty subsets $\mathcal{A}_1, \ldots, \mathcal{A}_k$ such that cover descriptions $C_1, \ldots, C_k$ with $C_i = (\mathcal{A}_i, V, \alpha_i, \psi_i)$ for Q exist.

It seems to be obvious to exploit such minimal cover description for the constructions in Theorem 5.4 and 7.2, instead of partitioning the set $\mathcal{A}$ of a cover description "manually". However, in contrast to our constructions, this would (possibly) not result in efficient algorithms to compute an acyclic or hierarchical rewriting from an arbitrary one, since it turns out that deciding whether a cover description is minimal is coNP-hard. A proof for the following proposition is given in the full version [15].

▶ **Proposition 8.2.** *Let Q be a hierarchical conjunctive query. It is* coNP*-hard to decide whether a cover description* $(\mathcal{A}, Q, \alpha, \psi)$ *is minimal.*

Applications of acyclic, free-connex acyclic, hierarchical and q-hierarchical queries

It is well known that many problems are tractable for acyclic conjunctive queries but (presumably) not for conjunctive queries in general. Notably, the evaluation, minimisation, and the containment problem are tractable for acyclic queries [30, 10, 16] but NP-complete for the class of conjunctive queries [9].

[12] In a nutshell, query Q' is a *contained rewriting* of query Q if $Q'(\mathcal{V}(D)) \subseteq Q(D)$ for any database D.

The class of free-connex conjunctive queries played a central role in the enumeration complexity of conjunctive queries. In [3], Bagan, Durand and Grandjean showed that the result of a free-connex acyclic conjunctive query can be enumerated with constant delay after a linear time preprocessing phase. Moreover, they also showed that the result of a self-join free acyclic conjunctive query that is not free-connex can not be enumerated with constant delay after a linear time preprocessing, unless $n \times n$ matrices can be multiplied in time $O(n^2)$.

Hierarchical queries have played a central role in different contexts. On the one hand, Dalvi and Suciu [12] showed that the class of hierarchical Boolean conjunctive queries characterises precisely the Boolean CQs that can be answered in polynomial time on probabilistic databases. This has been extended by Fink and Olteanu [13] to the notion of non-Boolean queries and queries with negation. On the other hand, Koutris and Suciu [24] studied hierarchical join queries in the context of query evaluation on massively parallel architectures. We refer to [14] for further applications of hierarchical queries.

The notion of q-hierarchical queries has played a central role in the evaluation of conjunctive queries under single tuple updates [6]. It is also related to factorised databases [22]. The notion of factorised databases has already been considered in various contexts [27, 4, 29]. A further recent source of information on structurally simple queries is [21].

9 Conclusion

We studied rewritability by acyclic queries or queries from CCQ, HCQ, or QHCQ. Based on a new characterisation of (exact) rewritability, we showed that acyclic queries have acyclic rewritings, if they have any CQ rewriting. The same holds for the other three query classes.

We showed that for acyclic queries and views the decision problem, whether an acyclic rewriting exists, is intractable, even for schemas with bounded arity, but becomes tractable if views have a bounded arity (even with unbounded schema arity) or are free-connex acyclic.

We leave the case of free-connex acyclic views and unbounded schemas open. Another interesting open question is the complexity of rewriting problems $\textsc{Rewr}(\mathbb{V}, \mathbb{Q}, \mathbb{R})$ with $\mathbb{R} \subsetneq \mathbb{Q}$, e.g. $\textsc{Rewr}(\mathsf{ACQ}, \mathsf{ACQ}, \mathsf{HCQ})$. Finally, it would be interesting to study whether our results can be extended to other classes of queries that can be evaluated efficiently like conjunctive queries of bounded treewidth.

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995. URL: `http://webdam.inria.fr/Alice/`.

2 Foto N. Afrati and Rada Chirkova. *Answering Queries Using Views, Second Edition*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2019. `doi:10.2200/S00884ED2V01Y201811DTM054`.

3 Guillaume Bagan, Arnaud Durand, and Etienne Grandjean. On acyclic conjunctive queries and constant delay enumeration. In Jacques Duparc and Thomas A. Henzinger, editors, *Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL, Lausanne, Switzerland, September 11-15, 2007, Proceedings*, volume 4646 of *Lecture Notes in Computer Science*, pages 208–222. Springer, 2007. `doi:10.1007/978-3-540-74915-8_18`.

4 Nurzhan Bakibayev, Tomás Kociský, Dan Olteanu, and Jakub Zavodny. Aggregation and ordering in factorised databases. *Proc. VLDB Endow.*, 6(14):1990–2001, 2013. `doi:10.14778/2556549.2556579`.

5 Pablo Barceló, Andreas Pieris, and Miguel Romero. Semantic optimization in tractable classes of conjunctive queries. *SIGMOD Rec.*, 46(2):5–17, 2017. `doi:10.1145/3137586.3137588`.

6 Christoph Berkholz, Jens Keppeler, and Nicole Schweikardt. Answering conjunctive queries under updates. In Emanuel Sallinger, Jan Van den Bussche, and Floris Geerts, editors, *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, pages 303–318. ACM, 2017. doi:10.1145/3034786.3034789.

7 Johann Brault-Baron. *De la pertinence de l'énumération : complexité en logiques propositionnelle et du premier ordre.* Theses, Université de Caen, April 2013. URL: https://hal.archives-ouvertes.fr/tel-01081392.

8 Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. View-based query processing: On the relationship between rewriting, answering and losslessness. In Thomas Eiter and Leonid Libkin, editors, *Database Theory - ICDT 2005, 10th International Conference, Edinburgh, UK, January 5-7, 2005, Proceedings*, volume 3363 of *Lecture Notes in Computer Science*, pages 321–336. Springer, 2005. doi:10.1007/978-3-540-30570-5_22.

9 Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In John E. Hopcroft, Emily P. Friedman, and Michael A. Harrison, editors, *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*, pages 77–90. ACM, 1977. doi:10.1145/800105.803397.

10 Chandra Chekuri and Anand Rajaraman. Conjunctive query containment revisited. *Theor. Comput. Sci.*, 239(2):211–229, 2000. doi:10.1016/S0304-3975(99)00220-0.

11 Rada Chirkova and Jun Yang. Materialized views. *Found. Trends Databases*, 4(4):295–405, 2012. doi:10.1561/1900000020.

12 Nilesh N. Dalvi and Dan Suciu. The dichotomy of conjunctive queries on probabilistic structures. In Leonid Libkin, editor, *Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China*, pages 293–302. ACM, 2007. doi:10.1145/1265530.1265571.

13 Robert Fink and Dan Olteanu. A dichotomy for non-repeating queries with negation in probabilistic databases. In Richard Hull and Martin Grohe, editors, *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, pages 144–155. ACM, 2014. doi:10.1145/2594538.2594549.

14 Robert Fink and Dan Olteanu. Dichotomies for queries with negation in probabilistic databases. *ACM Trans. Database Syst.*, 41(1):4:1–4:47, 2016. doi:10.1145/2877203.

15 Gaetano Geck, Jens Keppeler, Thomas Schwentick, and Christopher Spinrath. Rewriting with acyclic queries: Mind your head, 2022. arXiv:2201.05129.

16 Georg Gottlob, Nicola Leone, and Francesco Scarcello. The complexity of acyclic conjunctive queries. *J. ACM*, 48(3):431–498, 2001. doi:10.1145/382780.382783.

17 Gang Gou, Maxim Kormilitsin, and Rada Chirkova. Query evaluation using overlapping views: completeness and efficiency. In Surajit Chaudhuri, Vagelis Hristidis, and Neoklis Polyzotis, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, June 27-29, 2006*, pages 37–48. ACM, 2006. doi:10.1145/1142473.1142479.

18 Alon Y. Halevy. Answering queries using views: A survey. *VLDB J.*, 10(4):270–294, 2001. doi:10.1007/s007780100054.

19 Xiao Hu and Ke Yi. Instance and output optimal parallel algorithms for acyclic joins. In Dan Suciu, Sebastian Skritek, and Christoph Koch, editors, *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 450–463. ACM, 2019. doi:10.1145/3294052.3319698.

20 Muhammad Idris, Martín Ugarte, and Stijn Vansummeren. The dynamic yannakakis algorithm: Compact and efficient query processing under updates. In Semih Salihoglu, Wenchao Zhou, Rada Chirkova, Jun Yang, and Dan Suciu, editors, *Proceedings of the 2017 ACM International*

Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017, pages 1259–1274. ACM, 2017. doi:10.1145/3035918.3064027.

21 Ahmet Kara, Milos Nikolic, Dan Olteanu, and Haozhe Zhang. Trade-offs in static and dynamic evaluation of hierarchical queries. In Dan Suciu, Yufei Tao, and Zhewei Wei, editors, *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2020, Portland, OR, USA, June 14-19, 2020*, pages 375–392. ACM, 2020. doi:10.1145/3375395.3387646.

22 Jens Keppeler. *Answering Conjunctive Queries and FO+MOD Queries under Updates.* PhD thesis, Humboldt University of Berlin, Germany, 2020. URL: http://edoc.hu-berlin.de/18452/22264.

23 Benny Kimelfeld and Yehoshua Sagiv. Incrementally computing ordered answers of acyclic conjunctive queries. In Opher Etzion, Tsvi Kuflik, and Amihai Motro, editors, *Next Generation Information Technologies and Systems, 6th International Workshop, NGITS 2006, Kibbutz Shefayim, Israel, July 4-6, 2006, Proceedings*, volume 4032 of *Lecture Notes in Computer Science*, pages 141–152. Springer, 2006. doi:10.1007/11780991_13.

24 Paraschos Koutris and Dan Suciu. Parallel evaluation of conjunctive queries. In Maurizio Lenzerini and Thomas Schwentick, editors, *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, pages 223–234. ACM, 2011. doi:10.1145/1989284.1989310.

25 Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. Answering queries using views. In Mihalis Yannakakis and Serge Abiteboul, editors, *Proceedings of the Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 22-25, 1995, San Jose, California, USA*, pages 95–104. ACM Press, 1995. doi:10.1145/212433.220198.

26 Alan Nash, Luc Segoufin, and Victor Vianu. Views and queries: Determinacy and rewriting. *ACM Trans. Database Syst.*, 35(3):21:1–21:41, 2010. doi:10.1145/1806907.1806913.

27 Dan Olteanu and Jakub Závodný. Size bounds for factorised representations of query results. *ACM Trans. Database Syst.*, 40(1):2:1–2:44, 2015. doi:10.1145/2656335.

28 Rachel Pottinger and Alon Y. Halevy. Minicon: A scalable algorithm for answering queries using views. *VLDB J.*, 10(2-3):182–198, 2001. doi:10.1007/s007780100048.

29 Maximilian Schleich, Dan Olteanu, and Radu Ciucanu. Learning linear regression models over factorized joins. In Fatma Özcan, Georgia Koutrika, and Sam Madden, editors, *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 3–18. ACM, 2016. doi:10.1145/2882903.2882939.

30 Mihalis Yannakakis. Algorithms for acyclic database schemes. In *Very Large Data Bases, 7th International Conference, September 9-11, 1981, Cannes, France, Proceedings*, pages 82–94. IEEE Computer Society, 1981.