Parallel-Correctness and Containment for Conjunctive Queries with Union and Negation

GAETANO GECK, TU Dortmund University
BAS KETSMAN, École Polytechnique Fédérale de Lausanne
FRANK NEVEN, Hasselt University & transnational University of Limburg
THOMAS SCHWENTICK, TU Dortmund University

Single-round multiway join algorithms first reshuffle data over many servers and then evaluate the query at hand in a parallel and communication-free way. A key question is whether a given distribution policy for the reshuffle is adequate for computing a given query, also referred to as parallel-correctness. This article extends the study of the complexity of parallel-correctness and its constituents, parallel-soundness and parallel-completeness, to unions of conjunctive queries with negation. As a by-product, it is shown that the containment problem for conjunctive queries with negation is coNEXPTIME-complete.

CCS Concepts: • Information systems → Parallel and distributed DBMSs; Relational database query languages;

Additional Key Words and Phrases: Conjunctive queries, parallel-correctness, containment

ACM Reference format:
Gaetano Geck, Bas Ketsman, Frank Neven, and Thomas Schwentick. 2019. Parallel-Correctness and Containment for Conjunctive Queries with Union and Negation. ACM Trans. Comput. Logic 20, 3, Article 18 (June 2019), 24 pages.
https://doi.org/10.1145/3329120

1 INTRODUCTION
Motivated by recent in-memory systems such as Spark [1] and Shark [21], Koutris and Suciu introduced the massively parallel communication model (MPC) [14], where computation proceeds in a sequence of parallel steps each followed by global synchronisation of all servers. Of particular interest in the MPC model are queries that can be evaluated in one round of communication [9]. In its most naive setting, a query \( Q \) is evaluated by reshuffling the data over many servers, according to some distribution policy, and then computing \( Q \) at each server in a parallel but communication-free manner. A notable family of distribution policies is formed within the Hypercube algorithm [4, 9, 11]. A property of Hypercube distributions is that for any instance \( I \), the central execution of \( Q(I) \) always equals the union of the evaluations of \( Q \) at every computing node (or server). The
latter guarantees the correctness of the distributed evaluation for any conjunctive query by the Hypercube algorithm.

Ameloot et al. [5] introduced a general framework for reasoning about one-round evaluation algorithms under arbitrary distribution policies. They introduced parallel-correctness as a property of a query w.r.t. a distribution policy, which states that central execution always equals distributed execution; that is, it equals the union of the evaluations of the query at each server under the given distribution policy. One of the main results of Reference [5] is that deciding parallel-correctness for unions of conjunctive queries (UCQs) is \( \Pi^p_2 \)-complete under arbitrary distribution policies. The upper bound follows rather directly from a semantical characterisation of parallel-correctness in terms of properties of minimal valuations. Specifically, it was shown that a union of conjunctive queries is parallel-correct w.r.t. a distribution policy, if the distribution policy sends for every minimal valuation its required facts to at least one node.

In this article, we extend the study of parallel-correctness to conjunctive queries with negation (CQ\(^-\)) and unions of conjunctive queries with negation (UCQ\(^-\)). In fact, we study two additional but related notions: parallel-soundness and parallel-completeness. While parallel-correctness implies equivalence between centralised and distributed execution, parallel-soundness (respectively, parallel-completeness) requires that distributed execution is contained in (respectively, contains) centralised execution. Of course, parallel-soundness and parallel-completeness together are equivalent to parallel-correctness.

Unlike for CQs and UCQs [5], in the presence of negation, parallel-correctness can no longer be characterised in terms of properties of valuations. Instead, our algorithms are based on counter-examples of exponential size, yielding \( \text{coNEXPTIME} \) upper bounds. It turns out that this is optimal, though, as our corresponding lower bounds show. The proof of the lower bounds comes along an unexpected route: We exhibit a reduction from query containment for CQ\(^-\) to parallel-correctness of CQ\(^-\) (and its two variants) and show that query containment for CQ\(^-\) is \( \text{coNEXPTIME} \)-complete. This is considerably different from what we thought was folklore knowledge of the community. Indeed, the \( \Pi^p_2 \)-completeness result for query containment for CQ\(^-\) mentioned in Reference [19] only seems to hold for fixed database schemas (or a fixed arity bound, for that matter). We note that Mugnier et al. [16] provide a \( \Pi^p_2 \) upper bound proof for CQ\(^-\) containment and explicitly mention that it holds under the assumption that the arity of predicates is bounded by a constant. Altogether, parallel-correctness (and its variants) for (unions of) conjunctive queries with negation is thus complete for \( \text{coNEXPTIME} \).

Finally, a natural question is how the high complexity of parallel-correctness in the presence of negation can be lowered. We identify two cases in which the complexity drops. More specifically, the complexity decreases from \( \text{coNEXPTIME} \) to \( \Pi^p_2 \) if the database schema is fixed or the arity of relations is bounded, and to \( \text{coNP} \) for unions of full conjunctive queries with negation. In the latter case, we again employ a reduction from containment of full conjunctive queries (with negation) and obtain novel results on the containment problem in this setting as well. All upper bounds hold for queries with disequalities.

Outline. This article is further organised as follows: In Section 2, we discuss related work. In Section 3, we introduce the necessary definitions. We address parallel-correctness for unions of conjunctive queries in Section 3.6. We consider containment of conjunctive queries with negation in Section 4 and parallel-correctness together with its variants in Section 5. We discuss the restriction to full conjunctive queries in Section 6. We conclude in Section 7.

2 RELATED WORK

As mentioned in the introduction, Koutris and Suciu introduced the massively parallel communication model (MPC) [14]. A key property is that computation proceeds in a sequence of parallel
steps, each followed by global synchronisation of all computing nodes. In this model, evaluation of conjunctive queries [8, 14] and skyline queries [3] have been considered. Beame et al. [9] proved a matching upper and lower bound for the amount of communication needed to compute a full conjunctive query without self-joins in one communication round. The upper bound is provided by a randomised algorithm called Hypercube, which uses a technique that can be traced back to Ganguly et al. [13] and is described in the context of map-reduce by Afrati and Ullman [4].

Ameloot et al. [5] introduced a general framework for reasoning about one-round evaluation algorithms under arbitrary distribution policies. They introduced the notion of parallel-correctness and proved its associated decision problem to be $\Pi^P_2$-complete for conjunctive queries. In addition, towards optimisation in MPC, they considered parallel-correctness transfer. Here, parallel-correctness transfers from $Q$ to $Q'$ when $Q'$ is parallel-correct under every distribution policy for which $Q$ is parallel-correct. The associated decision problem for conjunctive queries is shown to be $\Pi^P_2$-complete. In addition, some restricted cases (e.g., transferability under Hypercube distributions) are shown to be NP-complete.

Our definition of a distribution policy is borrowed from Ameloot et al. [6] (but already surfaces in the work of Zinn et al. [22]), where distribution policies are used to define the class of policy-aware transducer networks. The work by Ameloot et al. [6, 7] relates coordination-free computation with definability in variants of Datalog. One-round communication algorithms in MPC can be seen as very restrictive coordination-free computation.

The complexity of query containment for conjunctive queries is proved to be NP-complete by Chandra and Merlin [10]. Levy and Sagiv provide a test for query containment of conjunctive queries with negation [15] that involves exploring an exponential number of possible counter-example instances. In the context of information integration, Ullman [19] gives a comprehensive overview of query containment (with and without negation) and states the complexity of query containment for $\text{CQ}^-$ to be $\Pi^P_2$-complete. As mentioned in the introduction, the latter apparently only holds when the database schema is fixed or the arity of relations is considered to be bounded. A proof for the $\Pi^P_2$ lower bound is given by Farré et al. [12, Corollary 4]. Based on Reference [15], Wei and Lausen [20] study a method for testing containment that exploits containment mappings for the positive parts of queries and additionally provide a characterisation for $\text{UCQ}^-$ containment.

In Reference [17, Theorem 1], the containment problem for quantifier-free, first-order queries (with and without negation) is shown to be coNP-complete for fixed schemas. However, the lower bound proof uses queries that are not in disjunctive normal form and therefore does not imply Theorem 4.

3 DEFINITIONS

3.1 Queries and Instances

We assume an infinite set $\text{dom}$ of data values that can be represented by strings over some fixed alphabet. By $\text{dom}_n$, we denote the set of data values represented by strings of length at most $n$. A database schema $\mathcal{D}$ is a finite set of relation names $R$, each with some arity $\text{ar}(R)$. We also write $R^{(k)}$ as a shorthand to denote that $R$ is a relation of arity $k$. We call $R(t)$ a fact when $R$ is a relation name and $t$ a tuple over $\text{dom}$ of appropriate arity. We say that a fact $R(t)$ is over a database schema $\mathcal{D}$ if $R \in \mathcal{D}$. For a subset $U \subseteq \text{dom}$, we write $\text{facts}(\mathcal{D}, U)$ for the set of possible facts over schema $\mathcal{D}$ and $U$ and by $\text{facts}(\mathcal{D})$, we denote $\text{facts}(\mathcal{D}, \text{dom})$. A (database) instance $I$ over $\mathcal{D}$ is a finite set of facts over $\mathcal{D}$. By $\text{adom}(I)$, we denote the set of data values occurring in $I$. A query $Q$ over input schema $\mathcal{D}_1$ and output schema $\mathcal{D}_2$ is a generic mapping from instances over $\mathcal{D}_1$ to instances over $\mathcal{D}_2$. Genericity means that for every permutation $\pi$ of $\text{dom}$ and every instance $I$, it holds $Q(\pi(I)) = \pi(Q(I))$. We say that $Q$ is contained in $Q'$, denoted $Q \subseteq Q'$ iff, for all instances $I$, it holds $Q(I) \subseteq Q'(I)$.
3.2 Unions of Conjunctive Queries with Negation

Let \( \text{var} \) be an infinite set of variables, disjoint from \( \text{dom} \). An \textit{atom} over schema \( D \) is of the form \( R(x) \), where \( R \) is a relation name from \( D \) and \( x = (x_1, \ldots, x_k) \) is a tuple of variables in \( \text{var} \) with \( k = ar(R) \). A \textit{conjunctive query} \( Q \) with negation and disequalities over input schema \( D \) is an expression of the form

\[
T(x) \leftarrow R_1(y_1), \ldots, R_m(y_m), \neg S_1(z_1), \ldots, \neg S_n(z_n), \beta_1, \ldots, \beta_p,
\]

where all \( R_i(y_i) \) and \( S_j(z_j) \) are atoms over \( D \), every \( \beta_i \) is a disequality of the form \( s \neq s' \) where \( s, s' \) are distinct variables occurring in some \( y_i \) or \( z_j \), and \( T(x) \) is an atom for which \( T \notin D \). Multiple occurrences of the same variable in the head are allowed. Additionally, for safety, we require that every variable in \( x \) occurs in some \( y_i \) and that every variable occurring in a negated atom has to occur in a positive atom as well (safe negation). We refer to the head atom \( T(x) \) as \( \text{head}_Q \), to the set \( \{R_1(y_1), \ldots, R_m(y_m), S_1(z_1), \ldots, S_n(z_n)\} \) as \( \text{body}_Q \), and to the set \( \{\beta_1, \ldots, \beta_p\} \) as \( \text{diseq}_Q \).

Specifically, we refer to \( \{R_1(y_1), \ldots, R_m(y_m)\} \) as the positive atoms in \( Q \), denoted \( \text{pos}_Q \), and to \( \{S_1(z_1), \ldots, S_n(z_n)\} \) as the \textit{negated} atoms of \( Q \), denoted \( \text{neg}_Q \). We denote by \( \text{vars}(Q) \) the set of all variables occurring in \( Q \). We refer to the class of conjunctive queries with negation and disequalities by \( \text{CQ}^{\neg,*} \), its restriction to queries without disequalities, without negated atoms, and without both by \( \text{CQ}^\neg, \text{CQ}^* \), and \( \text{CQ} \), respectively. As a shorthand, we refer to queries from \( \text{CQ}^\neg,* \) as \( \text{CQ}^{\neg,*} \)s and similarly for the other classes.

A \textit{pre-valuation} for a \( \text{CQ}^\neg,* \) \( Q \) is a total function \( \text{vars}(Q) \rightarrow \text{dom} \), which naturally extends to atoms and sets of atoms. It is \textit{consistent} for \( Q \), if \( V(\text{pos}_Q) \cap V(\text{neg}_Q) = \emptyset \), and \( V(s) \neq V(s') \), for every disequality \( s \neq s' \) of \( Q \), in which case it is called a valuation. Of course, for a conjunctive query without negated atoms and without disequalities, every pre-valuation is also a valuation. We refer to \( V(\text{pos}_Q) \) as the facts \textit{required} by \( V \), and to \( V(\text{neg}_Q) \) as the facts \textit{prohibited} by \( V \).

A valuation \( V \) \textit{satisfies} \( Q \) on instance \( I \) if all facts required by \( V \) are in \( I \) while no fact prohibited by \( V \) is in \( I \), that is, if \( V(\text{pos}_Q) \subseteq I \) and \( V(\text{neg}_Q) \cap I = \emptyset \). In that case, \( V \) \textit{derives} the fact \( V(\text{head}_Q) \).

The \textit{result} of \( Q \) on instance \( I \), denoted \( Q(I) \), is defined as the set of facts that can be derived by satisfying valuations for \( Q \) on \( I \).

A \textit{union} of conjunctive queries with negation and disequalities is a finite union of \( \text{CQ}^{\neg,*} \)s. That is, \( Q \) is of the form \( \bigcup_{i=1}^n Q_i \), where all subqueries \( Q_1, \ldots, Q_n \) have the same relation name in their head atoms. We assume disjoint variable sets among different disjuncts in \( Q \). That is, \( \text{vars}(Q_i) \cap \text{vars}(Q_j) = \emptyset \) for \( i \neq j \) and, in particular, \( \text{vars}(\text{head}_Q) \neq \text{vars}(\text{head}_{Q_i}) \). By \( \text{varmax}(Q) \), we denote the maximum number of variables that occurs in any disjunct of \( Q \). By \( \text{UCQ}^{\neg,*} \), we denote the class of unions of conjunctive queries with negation and disequalities and its fragments are denoted correspondingly.

A \( \text{CQ}^{\neg,*} \) is called \textit{full} if all of its variables occur in its head. A \( \text{UCQ}^{\neg,*} \) is \textit{full} if all its subqueries are full.

The \textit{result} of \( Q \) on instance \( I \) is \( Q(I) = \bigcup_{i=1}^n Q_i(I) \). Accordingly, a mapping from variables to data values is a \textit{valuation} for a \( \text{UCQ}^{\neg,*} \) \( Q \) if it is a valuation for one of its subqueries.

3.3 Networks, Data Distribution, and Policies

A \textit{network} \( N \) is a nonempty finite set of values from \( \text{dom} \), which we call \textit{(computing) nodes} (or servers). A \textit{distribution policy} \( P = (U, rfacts_p) \) for a database schema \( D \) and a network \( N \) consists of a universe \( U \) and a total function \( rfacts_p \) that maps each node of \( N \) to a set of facts from \( facts(D, U) \). A node \( \kappa \) is \textbf{responsible for} fact \( f \) (under policy \( P \)) if \( f \in rfacts_p(\kappa) \). As a shorthand (and slight abuse of notation), we denote the set of nodes \( \kappa \) that are responsible for some given fact \( f \) by \( P(f) \). For a distribution policy \( P \) and an instance \( I \) over \( D \), let \( \text{loc-inst}_{P,I} \) denote the function...
that maps each $\kappa \in \mathcal{N}$ to $I \cap rfacts_p(\kappa)$, that is, the set of facts in $I$ for which $\kappa$ is responsible. We sometimes refer to a given instance $I$ as the global instance and to $loc-inst_p,I(\kappa)$ as the local instance at node $\kappa$.

We note that for some facts from $\text{facts}(\mathcal{D}, U)$ there are no responsible nodes. This gives our framework some additional flexibility. However, it does not affect our results: In the lower bound proofs, we only use distributions for which all facts from $\text{facts}(\mathcal{D}, U)$ have some responsible nodes. Each distribution policy implicitly induces a network and each query implicitly defines a database (sub-) schema. Therefore, we often omit the explicit notation for networks and schemas. Although there is no technical necessity, we usually assume that instances refer only to constants in the policy’s universe, $\text{dom}(l) \subseteq U$, because otherwise parallel-completeness and parallel-correctness (introduced below) are violated trivially due to skipped facts. For other problems, such as parallel-soundness and transfer of parallel-correctness, this assumption is irrelevant.

Given some policy $P$ that is defined over a network $\mathcal{N}$, the result $[Q, P](I)$ of the distributed evaluation of a query $Q$ on an instance $I$ in one round is defined as the union of the results of the query evaluated on each node’s local instance. Formally,

$$[Q, P](I) \overset{\text{def}}{=} \bigcup_{\kappa \in \mathcal{N}} Q(\text{loc-inst}_p,I(\kappa)).$$

In the decision problem for parallel correctness (to be formalised later), the input consists of a query $Q$ and a distribution policy $P$. However, it is not obvious how distribution policies should be specified. In principle, they could be defined in an arbitrary fashion, but it is reasonable to assume that given a potential fact $f$, a node $\kappa$, and a policy $P$, it is not too hard to find out whether $\kappa$ is responsible for $f$ under $P$.

For UCQs, which are monotone, our complexity results are remarkably robust with respect to the choice of the representation of distribution policies. In fact, the complexity results coincide for the two extreme possible choices that we consider in this article. In the first case, distribution policies are specified by an explicit list of tuple-node-pairs; whereas in the second case, the test whether a given node is responsible for a given tuple can be carried out by a non-deterministic polynomial-time algorithm. However, we do require that some bound $n$ on the length of strings that represent node names and data values is given. Without such a restriction, no upper complexity bounds would be possible as nodes with names of super-polynomial length in the size of the input would not be accessible.

Considering queries with negated atoms, however, these two settings (seem to) differ, complexity-wise. Thus, we consider a third option, $P_{\text{rule}}$, in which the universe $U$ of a policy is explicitly enumerated and the responsibilities are defined by simple constraints (described below). The latter representation enjoys the same complexity properties as the full NP-test based case.

Now, we give more precise definitions of classes of policies and their representations as inputs of algorithmic problems. As said before, policies $P = (U, rfacts_p)$ from $P_{\text{fin}}$ are specified by an explicit enumeration of $U$ and of all pairs $(\kappa, f)$ where $\kappa \in P(f)$. A policy $P = (U, rfacts_p)$ from $P_{\text{rule}}$ is given by an explicit enumeration of $U$ and a list of rules of the form $\rho = (A, \kappa)$, where $A$ is an atom with variables and/or constants from $U$, and a network node $\kappa$. The semantics of such a rule is as follows: For every substitution $\mu : \text{var} \cup \text{dom} \rightarrow \text{dom}$ that maps variables to values from $U$ and leaves constants from $U$ unchanged, the node $\kappa$ is responsible for the fact $\mu(A)$. A rule is a fact rule if its atom does not contain any variables, that is, $A = R(a_1, \ldots, a_n)$, where $a_1, \ldots, a_n \in U$. In particular, $P_{\text{fin}} \subseteq P_{\text{rule}}$. 

ACM Transactions on Computational Logic, Vol. 20, No. 3, Article 18. Publication date: June 2019.
Example 3.1. Let distribution policy \( P \) over schema \( \{ \text{Rel}^{(3)} \} \) and network \( \{ \kappa_1, \kappa_2 \} \) be given by \( U = \{ 1, \ldots, 10 \} \) and the rules \( \{ \text{Rel}(1, x), \kappa_1 \}, \{ \text{Rel}(2, x, y), \kappa_2 \} \). On global instance
\[
I = \{ \text{Rel}(1, 7, 7), \text{Rel}(1, 7, 8), \text{Rel}(2, 9, 8), \text{Rel}(2, 9, 9) \},
\]
policy \( P \) induces local instances
\[
\text{loc-inst}_{P, I}(\kappa_1) = \{ \text{Rel}(1, 7, 7) \} \text{ and } \text{loc-inst}_{P, I}(\kappa_2) = \{ \text{Rel}(2, 9, 8), \text{Rel}(2, 9, 9) \}. \]

The most general classes of policies allow to specify policies by means of a “test algorithm” with time bound \( \ell^k \), where \( \ell \) is the length of the input and \( k \) some constant. Such an algorithm decides, for an input consisting of a node \( \kappa \) and fact \( f \), whether \( \kappa \) is responsible for \( f \). \(^1\) A policy \( P = (U, \text{rfacts}_p) \) from \( \mathcal{P}_{\text{npoly}}^k \) is specified by a pair \( (n, \mathcal{A}_P) \), where \( n \) is a natural number in unary representation and \( \mathcal{A}_P \) is a non-deterministic algorithm.\(^2\) The universe \( U \) of \( P \) is the set of all data values that can be represented by strings of length at most \( n \) (for some given fixed alphabet) and the underlying network consists of all nodes that are represented by strings of length at most \( n \), that is, \( \mathcal{N} = \text{dom}_\kappa. \) A node \( \kappa \) is responsible for a fact \( f \) if \( \mathcal{A}_P \), on input \( (\kappa, f) \), has an accepting run of at most \( |(\kappa, f)|^k \) steps. Clearly, each policy of \( \mathcal{P}_{\text{fin}} \) can be described in \( \mathcal{P}_{\text{npoly}}^2 \). Let \( \Psi_{\text{npoly}} \) denote the set\(^3\) \( \{ \mathcal{P}_{\text{npoly}}^k \mid k \geq 2 \} \) of distribution policies and by \( \Psi \) the set \( \{ \mathcal{P}_{\text{fin}}, \mathcal{P}_{\text{rule}} \} \cup \Psi_{\text{npoly}}. \)

### 3.4 Parallel-correctness, Soundness, and Completeness

In this article, we mainly consider the one-round evaluation algorithm for a query \( Q \) that first distributes (reshuffles) the data over the computing nodes according to \( P \), then evaluates \( Q \) in a parallel step at every computing node, and finally outputs all facts that are obtained in this way.\(^1\) As formalised next, the one-round evaluation algorithm is correct (sound, complete) if the query \( Q \) is parallel-correct (parallel-sound, parallel-complete) under \( P \).

**Definition 3.2.** Let \( Q \) be a query, \( I \) an instance, and \( P \) a distribution policy.

- \( Q \) is **parallel-sound** on \( I \) under \( P \) if \( Q(I) \supseteq [Q, P](I) \).
- \( Q \) is **parallel-complete** on \( I \) under \( P \) if \( Q(I) \subseteq [Q, P](I) \).
- \( Q \) is **parallel-correct** on \( I \) under \( P \) if \( Q(I) = [Q, P](I) \), that is, if it is parallel-sound and parallel-complete.

**Definition 3.3.** A query \( Q \) is parallel-correct (respectively, parallel-sound and parallel-complete) under distribution policy \( P = (U, \text{rfacts}_p) \), if \( Q \) is parallel-correct (respectively, parallel-sound and parallel-complete) on all instances \( I \subseteq \text{facts}(\mathcal{D}, U) \).

In Reference \([5]\), parallel-correctness is characterised in terms of minimal valuations as defined next:

**Definition 3.4.** Let \( Q \) be a CQ. A valuation \( V \) for \( Q \) is **minimal** for \( Q \) if there exists no valuation \( V' \) for \( Q \) such that \( V(\text{head}_Q) = V'(\text{head}_Q) \) and \( V'(\text{body}_Q) \subsetneq V(\text{body}_Q) \).

The following lemma is key in obtaining the \( \Pi_2^p \) upper bound on the complexity of testing parallel-correctness for conjunctive queries:

\(^1\)We note that it is important that for each class of policies there is a fixed \( k \) that bounds the exponent in the test algorithm as otherwise we could not expect a polynomial bound for all policies of that class.

\(^2\)For concreteness, say, a non-deterministic Turing machine.

\(^3\)Since “linear time” is a subtle notion, we rather not consider \( \mathcal{P}_{\text{npoly}}^1 \).

\(^4\)We note that, since \( P \) is defined on the granularity of a fact, the reshuffling does not depend on the current distribution of the data and can be done in parallel as well.
Lemma 3.5 (Characterisation of parallel-correctness for CQs [5]). A CQ $Q$ is parallel-correct under distribution policy $P = (U, r\text{facts}_P)$ if and only if the following holds:

For every minimal valuation $V$ for $Q$ over $U$, there is a node $\kappa \in N$ such that $V(body_Q) \subseteq r\text{facts}_P(\kappa)$.  

(C1)

Remark 3.6. Informally, Condition (C1) states that there is a node in the network where all facts required for $V$ meet.

3.5 Algorithmic Problems

We consider the following decision problems for various sub-classes $C$ and $C'$ of UCQ$^{\neg \neq}$ and classes $\mathcal{P}$ of distribution policies from $\{\mathcal{P}_{\text{fin}}, \mathcal{P}_{\text{rule}}\} \cup \frak{P}_{\text{npoly}}$:

| Problem                                    | Input:                                      | Question:                               |
|--------------------------------------------|---------------------------------------------|-----------------------------------------|
| Containment$(C, C')$:                       | $Q \in C$ and $Q' \in C'$                  | Is $Q \subseteq Q'$?                   |
| Parallel-Sound$(C, \mathcal{P})$:          | $Q \in C$, $P \in \mathcal{P}$             | Is $Q$ parallel-sound under $P$?        |
| Parallel-Complete$(C, \mathcal{P})$:       | $Q \in C$, $P \in \mathcal{P}$             | Is $Q$ parallel-complete under $P$?     |

3.6 Parallel-correctness: Unions of Conjunctive Queries

Parallel-correctness of unions of conjunctive queries (without negation) reduces to parallel-completeness for the simple reason that these queries are monotone and therefore parallel-sound for every distribution policy. We have shown that Condition (C1) can be extended to UCQ$^\neq$s, such that testing parallel-completeness for them remains in $\Pi^p_2$, even for policies from $\frak{P}_{\text{npoly}}$ [5]. Hardness already follows from $\Pi^p_2$-hardness of Parallel-Correct$(\text{CQ}, \mathcal{P}_{\text{fin}})$ [5].

Theorem 1 (Complexity of Parallel-Correctness for UCQ$^\neq$s [5]). For every $\mathcal{P} \in \frak{P}$, problem Parallel-Correct$(\text{UCQ}^{\neq}, \mathcal{P})$ is $\Pi^p_2$-complete.

4 CONTENTMENT OF CQ$^\neg$ AND UCQ$^\neg$

In this section, we establish the complexity of containment for CQ$^\neg$ and UCQ$^\neg$. We use these results to establish lower bounds on parallel-correctness and its constituents in the next section. Whereas containment for CQ has been intensively studied in the literature, the analogous problems for CQ$^\neg$ and UCQ$^\neg$ have hardly been addressed and seem to belong to folklore. In fact, we only found a reference of a complexity result for containment of CQ$^\neg$ in Reference [19], where a $\Pi^p_2$-algorithm for the problem is given, based on observations in Reference [15], and the existence of a matching lower bound is mentioned. However, as we show below, although the problem is indeed in $\Pi^p_2$ for queries defined over a fixed schema (or when the arity of relations is bounded), it is coNEXPTIME-complete in the general case.

We first show the lower bounds. They actually already hold for Boolean queries. We show that Containment$(\text{BCQ}^{\neg}, \text{UBCQ}^{\neg})$ is coNEXPTIME-hard by a reduction from the succinct 3-colorability problem. In a second step, we show that this hardness result does not depend on the use of unions. To this end, we reduce Containment$(\text{BCQ}^{\neg}, \text{UBCQ}^{\neg})$ to Containment$(\text{BCQ}^{\neg}, \text{BCQ}^{\neg})$. Here, BCQ$^{\neg}$ and UBCQ$^{\neg}$ denote the class of Boolean CQ$^{\neg}$s and unions of Boolean CQ$^{\neg}$s, respectively. Together, this establishes that Containment$(\text{BCQ}^{\neg}, \text{BCQ}^{\neg})$ and therefore in particular Containment$(\text{CQ}^{\neg}, \text{CQ}^{\neg})$ are coNEXPTIME-hard.
Proposition 4.1. Containment $\text{BCQ}^\sim, \text{UBCQ}^\sim$ is coNEXPTIME-hard.

Proof. The proof is by a reduction from the succinct 3-colorability problem, which asks whether a graph $G$, which is implicitly given by a circuit with binary AND- and OR- and unary NEG-gates, is 3-colorable. The latter problem is known to be NEXPTIME-complete [18]. We say that a circuit $C$, with $2\ell$ Boolean inputs, describes a graph $G = (N, E)$, when $N = \{0, 1\}^\ell$, and there is an edge $(n_1, n_2) \in N^2$ if and only if $C$ outputs true on input $n_1, n_2$.

Let $C$ be an input for the succinct 3-colorability problem with $2\ell$ Boolean inputs. We construct queries $Q_1$ and $Q_2$ such that $Q_1 \not\subseteq Q_2$ if and only if the graph described by $C$ is 3-colorable.

Both queries are over schema $D$, which consists of relation names DomainValues$^{(3)}$, Bool$^{(1)}$, And$^{(3)}$, Or$^{(3)}$, Neg$^{(2)}$, and Label$^{(\ell+1)}$. Intuitively, satisfaction of $Q_1$ will guarantee that there is a tuple $(a_0, a_1, a_2)$ with three different values in relation DomainValues. We will use, for some such tuple, $a_0, a_1, a_2$ as colors and $a_0, a_1$ as truth values. We will often assume without loss of generality that $(a_0, a_1, a_2) = (0, 1, 2)$. In particular, for such a tuple, $a_0$ is interpreted as false while $a_1$ is interpreted as true. The unary relation Bool will be forced by $Q_1$ to contain at least $a_0$ and $a_1$.

Relations And, Or, and Neg are intended to represent the respective logical functions. The first two attributes represent input values, and the last attribute represents the output. Again, $Q_1$ will guarantee that at least all triples of Boolean values that are consistent with the semantics of AND, OR, and NEG are present in these relations. Tuples in relation Label represent nodes together with their respective color (one can think of the representation of a node by $\ell$-ary addresses over a ternary alphabet).

We define query $Q_1$ as follows:

$$
T() \leftarrow \text{DomainValues}(w_0, w_1, w_2), \neg\text{DomainValues}(w_1, w_0, w_2), \neg\text{DomainValues}(w_2, w_1, w_0), \text{DomainValues}(w_0, w_2, w_1), \text{Bool}(w_0), \text{Bool}(w_1), \text{Neg}(w_1, w_0), \text{Neg}(w_0, w_1), \text{And}(w_0, w_0, w_0), \text{And}(w_0, w_1, w_0), \text{And}(w_1, w_0, w_0), \text{And}(w_1, w_1, w_1), \text{Or}(w_0, w_0, w_0), \text{Or}(w_0, w_1, w_1), \text{Or}(w_1, w_0, w_1), \text{Or}(w_1, w_1, w_1).
$$

It is easy to see that $Q_1$ enforces the conditions mentioned above.

In the following, we denote sequences $x_1, \ldots, x_\ell$ of $\ell$ variables by $x$.

We define $Q_2$ as the union of the queries $Q_2^1$ and $Q_2^2$, where subquery $Q_2^1$ is defined as:

$$
T() \leftarrow \text{Bool}(x_1), \text{Bool}(x_2), \ldots, \text{Bool}(x_\ell), \text{DomainValues}(y_r, y_g, y_b), \neg\text{Label}(x, y_r), \neg\text{Label}(x, y_g), \neg\text{Label}(x, y_b).
$$

Intuitively, $Q_2^1$ can be satisfied in a database if for some node, represented by $x$, there is no color.

Subquery $Q_2^2$ deals with the correctness of a coloring and uses a set circuit of atoms that is intended to check whether for two nodes $u$ and $v$, represented by $y$ and $z$, respectively, there is an edge between $u$ and $v$.

To this end, circuit uses the variables $y_1, \ldots, y_\ell, z_1, \ldots, z_\ell$, representing the input and, at the same time, the $2\ell$ input gates of $C$, and an additional variable $u_i$, for each gate of $C$, with the exception of the output gate. The output gate is represented by variable $w_1$. For each AND-gate represented by variable $v_1$ with incoming edges from gates represented by variables $u_1$ and $u_2$, circuit contains an atom And$(u_1, u_2, v_1)$. Likewise for OR- and NEG-gates.

Subquery $Q_2^2$ is defined as:

$$
T() \leftarrow \text{DomainValues}(w_0, w_1, w_2), \text{circuit}(y, u), \text{Label}(z, u).
$$
Intuitively, $Q^2_2$ returns true when two nodes, witnessed to be adjacent by the circuit, have the same color.

It remains to show that the graph represented by circuit $C$ is 3-colorable if and only if $Q_1 \not\subseteq Q_2$.

(If) Suppose $Q_1 \not\subseteq Q_2$. Thus, for some instance $I$, it holds $T() \in Q_1(I)$ while $T() \notin Q_2(I)$. The former implies that $I$ contains a fact $\text{DomainValues}(a_0, a_1, a_2)$, where $a_0$, $a_1$, and $a_2$ are distinct values from $\text{dom}$. Without loss of generality, we assume $a_0 = 0$, $a_1 = 1$, and $a_2 = 2$. The instance $I$ further contains the facts Bool(0) and Bool(1), and all "logical facts" induced by $Q_1$.

For every vector $\mathbf{n} \in \{0, 1\}^\ell$, there must be some $c \in \{0, 1, 2\}$ such that $\text{Label}(\mathbf{n}; c) \in I$, since otherwise $T() \in Q^2_2$. Let $label : \{0, 1\}^\ell \to \{0, 1, 2\}$ be chosen such that, for every $\mathbf{n} \in \{0, 1\}^\ell$, it holds $\text{label}(\mathbf{n}) = c$, for some $c \in \{0, 1, 2\}$, for which $\text{Label}(\mathbf{n}; c) \in I$.

We claim that $label$ is a valid coloring of the graph represented by $C$. Towards a contradiction, let us assume that there are two nodes $\mathbf{n}$ and $\mathbf{n}'$ that are connected by an edge and for which $\text{label}(\mathbf{n}) = \text{label}(\mathbf{n'}) = c$, for some $c$. Then $Q^2_2$ could be satisfied over $I$ by choosing a valuation that corresponds to a computation of $C$ that witnesses that there is an edge between $\mathbf{n}$ and $\mathbf{n}'$ and mapping $u$ to $c$, the desired contradiction.

(Only if) For some $\ell$, let $C$ be a circuit with input length $2\ell$ that describes a 3-colorable graph $G$. Let $label : \{0, 1\}^\ell \to \{0, 1, 2\}$ be a valid coloring for $G$. Let $I$ be the database with the following facts:

\[
\{\text{Label}(\mathbf{n}; label(\mathbf{n})) \mid \mathbf{n} \in \{0, 1\}^\ell\} \cup \{\text{DomainValues}(0, 1, 2), \text{Bool}(0), \text{Bool}(1)\} \cup \\
\{\text{Neg}(1, 0), \text{Neg}(0, 1), \text{And}(0, 0, 0), \text{And}(0, 1, 0), \text{And}(1, 0, 0), \text{And}(1, 1, 1)\} \cup \\
\{\text{Or}(0, 0, 0), \text{Or}(0, 1, 1), \text{Or}(1, 0, 1), \text{Or}(1, 1, 1)\}.
\]

Obviously, $T() \in Q_1(I)$ and $T() \notin Q^1_2(I)$. However, since $I$ only contains the “correct” logical facts, to satisfy $Q^2_2$, it would be necessary to find two nodes with the same label whose adjacency is witnessed by the canonical valuation corresponding to the semantics of $C$, which does not exist. Thus, $Q_1 \not\subseteq Q_2$. $\square$

Next, we provide the above-mentioned reduction to containment for $\text{BCQ}^\sim$.

**Proposition 4.2.** Containment($\text{BCQ}^\sim, \text{UBCQ}^\sim$) $\leq_P$ Containment($\text{BCQ}^\sim, \text{BCQ}^\sim$)

**Proof.** Let $Q_1$ be in $\text{BCQ}^\sim$ and $Q_2 = \bigcup^m_{i=1} Q^i_2$ be in $\text{UBCQ}^\sim$ over some database schema $\mathcal{D}$. Recall our assumption, that each disjunct is defined over a disjoint set of variables. Next, we construct $\text{CQ}$’s $Q'_1$ and $Q'_2$ such that $Q'_1 \subseteq Q'_2$ if and only if, $Q_1 \subseteq Q_2$.

We explain the intuition behind the reduction by means of an example. To this end, let $Q_1$ be $H() \leftarrow A(x, y)$ and let $Q_2$ be the $Q^1_2 \cup Q^2_2$, where $Q^1_2$ is $H() \leftarrow A(u_1, v_1), B(u_1, v_1)$ and $Q^2_2$ is $H() \leftarrow A(u_2, v_2), \neg B(u_2, v_2)$, both formulated over the schema $\mathcal{D} = \{A^{(2)}, B^{(2)}\}$. The query $Q^2_2$ takes the following form:

$$H() \leftarrow \text{Active}(x_0, x_1; \ell_1, \ell_2), a(\ell_1, Q^1_2), a(\ell_2, Q^2_2),$$

where $a(w, Q)$ denotes the modification of the body of $Q$ by replacing every atom $R(x)$ by $R'(w, x)$. Both queries are defined over the schema $\mathcal{D}' = \{A^{(3)}, B'^{(3)}, \text{Active}^{(4)}\}$. Notice that $Q'_2$ contains a concatenation of the disjuncts of $Q_2$. In addition, relations $A$ and $B$ are extended with a new first column with the purpose of labelling tuples. This labelling allows to encode two (or even more) instances over $\mathcal{D}$ by one instance over $\mathcal{D}'$. Specifically, $\text{body}_{Q'_1}$ (not shown) is constructed in such a way that when there is a satisfying valuation for $Q'_1$ there are two different data values, say
0 and 1. So, an instance $I$ over $D$ can be encoded as $I^0 = \{A'(0,a,b) \mid A(a,b) \in I\} \cup \{B'(0,a,b) \mid B(a,b) \in I\}$ or as $I^1 = \{A'(1,a,b) \mid A(a,b) \in I\} \cup \{B'(1,a,b) \mid B(a,b) \in I\}$. In addition, when there is a satisfying valuation for $Q_1'$, there is an instance $I_0$ on which every disjunct of $Q_2$ is true, and there is an instance $I_1$ on which $Q_1$ is true. So, both $Q_2'_{1,1}$ and $Q_2'_{2,2}$ evaluate to true on $I_0$ when $\ell_1$ and $\ell_2$ are interpreted by label 0. However, for $Q_1$ to be contained in $Q_2$, we need that at least one of the disjuncts $Q_2'_{1,1}$ or $Q_2'_{2,2}$ evaluates to true over $I_1$, that is, when its labelling variable is interpreted as 1. Atom $\text{Active}(x_0, x_1, \ell_1, \ell_2)$ will ensure that $x_0$ and $x_1$ correspond with the values 0 and 1, and that at least one of the labelling variables $\ell_1$ or $\ell_2$ is equal to 1. In other words, $\text{Active}$ chooses which disjunct to activate over $I_1$. So, at least one disjunct of $Q_2$ evaluates to true on the instance $I_1$ on which $Q_1$ is satisfied.

We now explain the reduction in more detail. We assume that $\text{CQ}^- Q_1^i$ is satisfiable, for every $i \in \{1, \ldots, m\}$.\footnote{Notice that a $\text{CQ}^- Q$ is satisfiable if and only if $pos_Q \cap neg_Q = \emptyset$, which can easily be verified in polynomial time.} Set $D' \overset{\text{def}}{=} \{R^{(k+1)} \mid R^{(k)} \in D\} \cup \{\text{Active}^{(m+2)}\}$. As explained above, the relation $\text{Active}$ serves as an index for the disjuncts in $Q_2$. Whereas the first two positions encode the bits zero (0) and one (1), an atom of the form $\text{Active}(0, 1; 0, \ldots, 0, 1, \ldots, 0)$ with a 1 occurring on position $i + 2$ is meant to indicate disjunct $i$. Recall that $\alpha(w, Q)$ denotes the modification of the body of $Q$ by replacing every atom $R(x)$ by $R'(w, x)$, while retaining existent negation symbols. We further define\footnote{This is the same definition as in Section 5.2 and is stated here for convenience only.} mapping $\alpha_a^{-1}$, for $a \in \text{dom}$, to map sets of facts over $D'$ to sets of facts over $D$, by selecting all facts with first parameter $a$, deleting this parameter, and replacing each relation name $R'$ by $R$.

Now, define $Q_1'$ as:

$$H() \leftarrow \text{Active}(w_0, w_1; w_1, w_0, \ldots, w_0), \text{Active}(w_0, w_1; w_0, w_1, \ldots, w_0), \ldots,$$

$$\text{Active}(w_0, w_1; w_0, w_0, w_0, \ldots, w_0), \text{Active}(w_0, w_1; w_0, w_0, \ldots, w_0),$$

$$\neg \text{Active}(w_1, w_0; w_1, w_0, \ldots, w_0),$$

$$\alpha(w_1, Q_1), \alpha(w_0, Q_1^1), \ldots, \alpha(w_0, Q_2^m);$$

and $Q_2'$ as:

$$H() \leftarrow \text{Active}(w_0, w_1; z_1, \ldots, z_m), \alpha(z_1, Q_2^1), \ldots, \alpha(z_m, Q_2^m),$$

where $w_0, w_1, z_1, \ldots, z_m$ are fresh and distinct variables.

In the remainder, we argue that $Q_1' \subseteq Q_2'$ if and only if $Q_1 \subseteq Q_2$:

\textbf{(If)} Suppose $Q_1 \subseteq Q_2$. Let $I'$ be an arbitrary instance over $D'$ for which $H() \in Q_1'(I)$. Let $V_1'$ be a satisfying valuation for $Q_1'$ over $I'$. Since $V_1'$ has to satisfy both literals $\text{Active}(w_0, w_1; w_1, w_0, \ldots, w_0)$ and $\neg \text{Active}(w_1, w_0; w_1, w_0, \ldots, w_0)$, it follows that $V_1'(w_1) \neq V_1'(w_0)$. For simplicity, we assume that $V_1'(w_1) = 1$ and $V_1'(w_0) = 0$.

Thus, $I'$ contains the facts $\text{Active}(0, 1; 1, 0, \ldots, 0), \ldots, \text{Active}(0, 1; 0, \ldots, 0, 1)$, but not fact $\text{Active}(1, 0; 1, 0, \ldots, 0)$. Furthermore, for every $i$, set $V_1'(\alpha(w_0, Q_i^j))$ consists of facts from $I'$ labelled with 0, and set $V_1'(\alpha(w_1, Q_i))$ consists of facts from $I'$ labelled with 1.

Now, let $I$ be the set $\alpha_a^{-1}(I') = \{R(t) \mid R'(1, t) \in I'\}$ consisting of all facts in $I'$ labelled with 1. By construction, and since $I'$ contains all facts from $V_1'(\alpha(w_1, Q_i))$, $I$ contains all facts from $V_1'(Q_i)$. Note that $V_1$ is a valuation for $Q_1$, as well. In fact, $V_1'$ is a satisfying valuation for $Q_1$ over $I$. Consequently, thanks to $Q_1 \subseteq Q_2$, there must be a satisfying valuation $V_2$ for some disjunct $Q_2'_{1,1}$ of $Q_2$ over $I.$
We define a satisfying valuation \( V'_2 \) for \( Q'_2 \) over \( I' \) witnessing \( Q'_1 \subseteq Q'_2 \) as follows:

\[
V'_2(x) = \begin{cases} 
0 & \text{if } x = w_0, \\
1 & \text{if } x = w_1, \\
1 & \text{if } x = z_i, \\
0 & \text{if } x = z_j \text{ and } j \neq i, \\
V_2(x) & \text{if } x \text{ occurs in } Q'_2, \\
V'_1(x) & \text{if } x \text{ occurs in } Q'_2 \text{ and } j \neq i.
\end{cases}
\]

In particular, that \( V'_2(z_i) = 1 \) holds, connects the atoms \( \alpha(z_i, Q'_2) \) in query \( Q'_2 \) to the 1-labeled facts induced by atoms \( \alpha(w_1, Q_1) \) in query \( Q'_1 \).

Therefore, \( \alpha(z_i, Q'_2) \) becomes true, since \( V_2 \) satisfies \( Q'_2 \). Furthermore, it is easy to verify that \( V'_2 \) satisfies \( Q'_2 \); the first atom and all conjuncts \( \alpha(z_j, Q'_2) \) with \( j \neq i \) become true, since the respective (the 0-labeled) facts were guaranteed by \( Q'_1 \).

(Only if) The proof is by contraposition, that is, we show that \( Q_1 \not\subseteq Q_2 \) implies \( Q'_1 \not\subseteq Q'_2 \). Therefore, let \( I \) be an instance, and \( V_1 \) a valuation such that \( V_1 \) satisfies \( Q_1 \) over \( I \), but no \( Q'_1 \) has a satisfying valuation over \( I \). Since every query \( Q'_1 \) is satisfiable, there is, for every \( i \), a satisfying valuation \( V^i \). In fact, these valuations can be chosen with pairwise disjoint range. Now, we define \( I^i \) as the following set of facts:

\[
I^i = \alpha(1, I) \cup \alpha(0, V^i(pos_{Q^i})) \cup \cdots \cup \alpha(0, V^n(pos_{Q^n})), \\
\cup \{\text{Active}(0, 1; 1, 0, \ldots, 0), \ldots, \text{Active}(0, 1; 0, \ldots, 0, 1)\}.
\]

It is easy to check\(^7\) that from \( V_1 \) and the \( V^i \) a satisfying valuation for \( Q'_1 \) over \( I^i \) can be constructed. However, any satisfying valuation of \( Q'_2 \) over \( I^i \) would require to use facts from \( \alpha(1, I) \) for at least one \( \alpha(z_i, Q'_2) \) and would thus induce a valuation of \( Q'_2 \) over \( I \), the desired contradiction. \( \Box \)

Combining Propositions 4.1 and 4.2, we get the following corollary:

**Corollary 4.3.** Containment\((CQ^+, CQ^-)\) is \( \text{coNEXPTIME}-\text{hard}\).

The corresponding upper bounds hold also in the presence of disequalities and are shown by small model (i.e., counter-example) properties. To this end, we make use of a restricted monotonicity property of \( UCQ^- \)'s, which was already observed in Proposition 2.4 of Reference [2]. For an instance \( I \) and a set \( D \) of data values, we denote by \( I_D \) the restriction of \( I \) to facts that only use values from \( D \). In a nutshell, the following lemma states that the queries in \( UCQ^- \) are closed under extensions by facts with fresh constants:

**Lemma 4.4 ([2]).** Let \( Q \) be a query from \( UCQ^- \). Then \( Q(I_D) \subseteq Q(I) \) holds for every instance \( I \) and every set \( D \) of data values.

**Proof.** Let \( f \in Q(I_D) \) via a valuation \( V \) for a disjunct \( Q_i \) of \( Q \). Thus, \( V(pos_{Q_i}) \subseteq I_D \subseteq I \).

By definition, every variable \( x \) of \( Q_i \) occurs in a positive atom and therefore \( V(x) \in D \). Thus, \( V(neg_{Q_i}) \cap I = V(neg_{Q_i}) \cap I_D = \emptyset \) and \( f \in Q(I) \) as claimed. \( \Box \)

Now, we can establish the following small model property for testing containment:

**Lemma 4.5.** Let \( Q_1, Q_2 \in UCQ^- \). If there is an instance \( I \), where \( Q_1(I) \not\subseteq Q_2(I) \), then there is also an instance \( J \subseteq I \), where \( Q_1(J) \not\subseteq Q_2(J) \), and \(|\text{dom}(J)| \leq \text{varmax}(Q_1)\).

---

\(^7\)This can be proven along the lines of Claim 5.5.
Proof. Let I be as in the lemma and let f be a fact with f ∈ Q₁(I) and f /∈ Q₂(I). Let V be a valuation that derives f via some disjunct Q₁ⁱ of Q₁.

Let D \( \overset{\text{def}}{=} \text{adom}(V(\text{pos} Q₁)) \) and J \( \overset{\text{def}}{=} I|_D \) the set of all facts in I using only values from adom(V(\text{pos} Q₁)). By definition, |adom(J)| ≤ varmax(Q₁). Clearly, V is still a satisfying valuation for Q₁ⁱ over J. However, by Lemma 4.4, f /∈ Q₂(J) = Q₂(I|_D).

The upper bounds follow easily from Lemma 4.5.

Proposition 4.6. The following upper bounds hold:

1. Containment(UCQⁿ⁻, UCQⁿ⁻) is in coNEXPTIME.
2. For every k, containment of UCQⁿ⁻-queries over schemas with arity bound k is in \( \Pi^p_2 \).

Proof. In both cases, we consider the complement of Containment(UCQⁿ⁻, UCQⁿ⁻). Let \( m \overset{\text{def}}{=} \text{varmax}(Q₁) \).

1. A NEXPTIME algorithm, on input Q₁, Q₂, can simply guess an instance J with a domain of at most m elements and a fact f, and verifies that f ∈ Q₁(J) but f /∈ Q₂(J). For the latter tests, it can simply iterate, in exponential time, over all valuations over J for Q₁ and Q₂.
2. For a fixed arity bound, the minimal counter-example J is of size at most m². It can thus be guessed in polynomial time. That f ∈ Q₁(J) can be verified non-deterministically. That f /∈ Q₂(J) can be verified by a universal computation in polynomial time.

A claim of a \( \Pi^p_2 \) upper bound for containment of CQs with negation can be found in Reference [19]. It was not made clear there, that this claim assumes bounded arity of the schema. That the containment problem is \( \Pi^p_2 \)-complete for schemas of bounded arity has been explicitly shown in Reference [16]. Clearly, Proposition 4.6.2 follows directly and 4.6.1 is only a variation of it. From Proposition 4.6 and Corollary 4.3, the main result of this section immediately follows:

Theorem 2. The following problems are coNEXPTIME-complete:

1. Containment(BCQ⁻, BCQ⁻),
2. Containment(UCQ⁻, UCQ⁻).

5 PARALLEL-CORRECTNESS: UNIONS OF CONJUNCTIVE QUERIES WITH NEGATION

As mentioned in Section 3.6, for conjunctive queries without negation, parallel-soundness always holds and thus parallel-correctness and parallel-completeness coincide, thanks to monotonicity. For queries with negation, the situation is different. Distributed evaluation can be complete but not sound, or vice versa. For this reason, we have to distinguish all three problems separately: correctness, soundness, and completeness. However, the complexity is the same in all three cases.

Our results show a second, more crucial difference. Whereas parallel completeness for CQs without negation could be characterised in terms of valuations, that is, objects of polynomial size, our algorithms for CQs with negation involve counter-examples of exponential size (if the arity of schemas is not bounded) and the coNEXPTIME lower bound results indicate that this is unavoidable. We illustrate the observation that counter-examples might need an exponential number of tuples by the following example:

Example 5.1. Let \( Q \) be the following conjunctive query with negation:

\[
H() \leftarrow \text{Bool}(w₀, w₀), \text{Bool}(w₁, w₁), \text{Bool}(x₁, x₁), \ldots, \text{Bool}(xₙ, xₙ), \\
\neg \text{Bool}(w₀, w₁), \neg \text{Rel}(x₁, \ldots, xₙ).
\]
Let \( P \) be the policy defined over universe \( U = \{0, 1\} \) and two-node network \( \{\kappa_1, \kappa_2\} \), which distributes all facts except \( \text{Rel}(0, \ldots, 0) \) to node \( \kappa_1 \) and only fact \( \text{Rel}(0, \ldots, 0) \) to node \( \kappa_2 \).

Query \( Q \) is not parallel-sound under policy \( P \), but the smallest counter-example \( I \) is of exponential size, as we argue next. Indeed, let \( I = \{\text{Bool}(0, 0), \text{Bool}(1, 1)\} \cup \{\text{Rel}(a_1, \ldots, a_n) \mid (a_1, \ldots, a_n) \in \{0, 1\}^n\} \). Furthermore, let valuation \( V \) map variables \( w_1 \) and \( w_0 \) to 1 and 0, respectively, and map \( x_i \) to 0, for every \( i \in \{1, \ldots, n\} \). Then, valuation \( V \) satisfies \( Q \) on instance \( \text{loc-inst}_P(I) = I \setminus \{\text{Rel}(0, \ldots, 0)\} \), because neither \( \text{Bool}(0, 1) \) nor \( \text{Rel}(0, \ldots, 0) \) is contained in the local instance. Furthermore, there is no satisfying valuation \( W \) for \( Q \) on the global instance \( I \), because \( W \) would have to map each \( x_i \) to either 0 or 1, implying that \( W(\text{Rel}(x_1, \ldots, x_n)) \in I \).

However, there is no smaller instance: Let \( I^* \) be some instance over universe \( U \) that has a locally satisfying valuation \( V \). The combination of atoms \( \text{Bool}(w_0, w_1), \text{Bool}(w_1, w_1), \) and \( \neg\text{Bool}(w_0, w_1) \) in query \( Q \) then implies existence of both facts \( \text{Bool}(0, 0) \) and \( \text{Bool}(1, 1) \), because variables \( w_0 \) and \( w_1 \) cannot be mapped onto the same data value.

Assume that fact \( \text{Rel}(a_1, \ldots, a_n) \), for some \( (a_1, \ldots, a_n) \in \{0, 1\}^n \) is missing from \( I^* \). Then the valuation \( W \) that maps \( w_0 \mapsto 0, w_1 \mapsto 1 \) and \( x_i \mapsto a_i \), for every \( i \in \{1, \ldots, n\} \), satisfies \( Q \) also globally, on instance \( I^* \), and can therefore be no example against parallel-soundness, which contradicts our choice of \( I^* \). Thus, \( \text{Rel}(a_1, \ldots, a_n) \in I^* \), for every \( (a_1, \ldots, a_n) \in \{0, 1\}^n \). We therefore have \( I \subseteq I^* \) and, in particular, instance \( I^* \) contains at least as many facts as instance \( I \).

The results of this section are summarised in the following theorem:

**Theorem 3.** For every class \( \mathcal{P} \in \{\mathcal{P}_{\text{rule}}, \mathcal{P}_{\text{poly}}\} \) of distribution policies, the following problems are \( \text{coNEXPTIME} \)-complete:

1. \( \text{Parallel-Sound}(\text{UCQ}^\neg, \mathcal{P}) \),
2. \( \text{Parallel-Complete}(\text{UCQ}^\neg, \mathcal{P}) \),
3. \( \text{Parallel-Correct}(\text{UCQ}^\neg, \mathcal{P}) \).

Theorem 3 follows from Propositions 5.2 and 5.4 below. It is easy to show that, when restricted to schemas with some fixed (but sufficiently large, for hardness) arity bound, all these problems are \( \Pi_2^P \)-complete.

### 5.1 Upper Bounds

In this section, we show the upper bounds of Theorem 3, summarised in the following proposition.

**Proposition 5.2.** For every class \( \mathcal{P} \in \mathcal{P} \), the following problems are in \( \text{coNEXPTIME} \):

1. \( \text{Parallel-Sound}(\text{UCQ}^\neg, \mathcal{P}) \),
2. \( \text{Parallel-Complete}(\text{UCQ}^\neg, \mathcal{P}) \),
3. \( \text{Parallel-Correct}(\text{UCQ}^\neg, \mathcal{P}) \).

If the arity of schemas is bounded by some fixed number, then these problems are in \( \Pi_2^P \).

**Proof.** As already indicated above, the proof relies on a bound on the size of a smallest counter-example. More specifically, we first show the following claim:

**Claim 5.3.** Let \( Q \in \text{UCQ}^\neg \) and let \( P \) be an arbitrary distribution policy. Then the following statements hold:

1. If \( Q \) is not parallel-complete under \( P \), then there is an instance \( J \) over a domain with at most \( \text{varmax}(Q) \) elements such that \( Q \) is not parallel-complete on \( J \) under \( P \).
2. If \( Q \) is not parallel-sound under \( P \), then there is an instance \( J \) over a domain with at most \( \text{varmax}(Q) \) elements such that \( Q \) is not parallel-sound on \( J \) under \( P \).
Towards statement (1), let us assume that $Q$ is not parallel-complete on some instance $I$ under $P$. Let $V$ be a valuation of a disjunct $Q_i$ of $Q$ that derives a fact $f$ globally that is not derived on any node of the network. Let $D \overset{\text{def}}{=} \text{dom}(V(\text{pos}_{Q_i}))$ and $J \overset{\text{def}}{=} I|_D$. Clearly, $|D| \leq \text{varmax}(Q)$ and $V$ still derives $f$ globally on instance $J$ via $Q_i$. However, for every node $\kappa$, it holds $Q(\text{loc-instp}_I(\kappa)) = Q(\text{loc-instp}_I(\kappa)|_D) \subseteq Q(\text{loc-instp}_I(\kappa))$, thanks to Lemma 4.4. Therefore $f$ is not derived on $\kappa$, and thus $J$ witnesses the lack of parallel-completeness of $Q$ under $P$.

The proof of statement (2) is completely analogous. Given a counter-example $I$ and a valuation $V$ that derives a fact $f$ on some node $\kappa$ via $Q_i$, for which $f$ is not derived globally, we define $D \overset{\text{def}}{=} I_{\text{dom}(V(\text{pos}_{Q_i}))}$ and show that $J \overset{\text{def}}{=} I|_D$ is the desired counter-example.

It only remains to describe the algorithm that tests the complement of parallel completeness non-deterministically. On input $Q$ and $P$ (specified by $(n^t, T) \in \mathcal{P}_n^{k_{\text{poly}}}$), the algorithm simply guesses an instance $J$ over a domain with at most $\text{varmax}(Q)$ values from $\text{dom}_n$, and verifies that $J$ is a counter-example showing that $Q$ is not parallel-complete under $P$. From Claim 5.3, it follows that this algorithm is correct, since a counter-example must exist if $Q$ is not parallel-complete under $P$, and the actual data values do not matter. It remains to show the complexity bounds and, in particular, to describe how the verification part can be done.

In the general case, without a bound on the arity of the schema, the verification is done as follows: The algorithm guesses a valuation $V$, which produces some fact $f$ globally and which is not derived at any node. To test that $f$ is not derived at any node, the algorithm cycles through all nodes and all valuations $V$ over $\text{dom}_n$. The number of combinations is bounded\(^8\) by $2^n \times (2^n)^{\text{varmax}(Q)} = 2^{n(\text{varmax}(Q)+1)}$. Each test can be performed by a simulation of all runs of $T$, which amounts to at most $2^n k$ simulations of at most $n^k$ steps each. Altogether, the algorithm needs time at most $2^{|(Q,P)|^{k+2}}$.

If there is a fixed bound $\ell$ on the arity of the underlying schema, then the maximum size of the minimal counter-example becomes polynomial and the test that $f$ is derived globally can be done non-deterministically in polynomial time and the test that it is not derived locally can be done universally in polynomial time, thus altogether yielding a $\Pi^p_2$-computation.

The case of parallel-soundness is completely analogous (using the second statement of Claim 5.3 and the case of parallel-correctness follows, since it suffices to test parallel completeness and soundness).

\section{5.2 Lower Bounds}

The lower bounds stated in Theorem 3 follow from a polynomial time reduction from problem \textsc{Containment}(BCQ\textsuperscript{+}, BCQ\textsuperscript{−}), for which we showed coNEXPTIME-hardness in Section 4.

\textbf{Proposition 5.4.} The following problems are coNEXPTIME-hard:

\begin{enumerate}
  \item Parallel-Sound(CQ\textsuperscript{−}, \mathcal{P}_\text{rule}),
  \item Parallel-Complete(CQ\textsuperscript{−}, \mathcal{P}_\text{rule}),
  \item Parallel-Correct(CQ\textsuperscript{−}, \mathcal{P}_\text{rule}).
\end{enumerate}

\textbf{Proof.} Interestingly, all three results are shown by the same reduction from decision problem \textsc{Containment}(BCQ\textsuperscript{+}, BCQ\textsuperscript{−}).

The basic idea for this reduction is very simple: it combines both queries $Q_1, Q_2 \in \text{BCQ}^+$ of the given containment instance into a single query $Q \in \text{BCQ}^-$ and infers an appropriate distribution policy $P$. To emulate separate derivation for both queries in the combined query, an activation mechanism is used that resembles the proof of Proposition 4.2. In this fashion, the two queries can

\(^8\)We assume a binary alphabet, here.
be evaluated over different subsets of the considered instance by annotating both the facts in the instance as well as the atoms of the query.

We next describe the reduction in detail: Thus, let \( Q_1, Q_2 \in \text{BCQ}^- \) be queries over some schema \( \mathcal{D} \) and let \( m \equiv \max \{ \text{varmax}(Q_1), \text{varmax}(Q_2) \} \). Without loss of generality, we assume the variable sets of \( Q_1 \) and \( Q_2 \) to be disjoint. We will also assume in the following that both \( Q_1 \) and \( Q_2 \) are satisfiable. This is the case (for \( Q_1 \)) if and only if \( \text{pos}_{Q_1} \cap \text{neg}_{Q_1} = \emptyset \) and can therefore be easily tested in polynomial time. If one of the tests fails, then some appropriate constant instance of \( \text{Parallel-Complete}(\text{CQ}^-, \mathcal{P}_\text{rule}) \) or one of the other problem variants, respectively, can be computed.

We define a (Boolean) query \( Q \in \text{BCQ}^- \) and a policy \( P \in \mathcal{P}_\text{rule} \) over domain \( \{1, \ldots, m\} \) that can be computed from \( Q_1 \) and \( Q_2 \) in polynomial time. The schema for \( Q \) is \( \mathcal{D}' \equiv \{ R^{n(k+1)} | R(k) \in \mathcal{D} \} \). That is, each relation name \( R \) of \( \mathcal{D} \) occurs as \( R' \) in \( \mathcal{D}' \) with an arity incremented by one. Additionally, \( Q \) uses relation names Type, Start\(_1\), Start\(_2\), and Stop, which we assume not to occur in schema \( \mathcal{D} \). Besides the variables of \( Q_1 \) and \( Q_2 \), query \( Q \) uses variables \( \ell_1, \ell_2, t \).

We use the function \( \alpha \), defined in the proof of Proposition 4.2, which adds its first parameter as first component to every tuple in its second parameter and translates relation names \( R \) into \( R' \). In Proposition 4.2, the first parameter was always a variable and the second a set of atoms, but we use \( \alpha \) also for a data value as first and a set of facts as second parameter in the obvious way. We write \( \alpha^{-1}_a \) for the function mapping sets of facts over \( \mathcal{D}' \) to sets of facts over \( \mathcal{D} \), by selecting, from a set of facts, all facts with first parameter \( a \), deleting this parameter, and replacing each name \( R \) by \( R' \). Finally, \( \pi_a(I) \equiv \alpha(a, \alpha^{-1}_a(I)) \) is the restriction of \( I \) to all facts with \( a \) in their first component.

The combined query \( Q \) has \( \text{head}_Q \equiv H() \) and body

\[
\text{body}_Q \equiv \alpha(\ell_1, \text{body}_{Q_1}) \cup \alpha(\ell_2, \text{body}_{Q_2}) \cup \{ \text{Type}(t), \text{Start}_1(\ell_1), \text{Start}_2(\ell_1), \text{Start}_2(\ell_2) \} \cup \{ \neg \text{Stop}(\ell_1), \neg \text{Stop}(\ell_2) \}. 
\]

Policy \( P \) is defined over universe \( U \equiv \{1, \ldots, m\} \), schema \( \mathcal{D}' \cup \{ \text{Type}, \text{Start}_1, \text{Start}_2, \text{Stop} \} \), and network \( \mathcal{N} \equiv \{ \kappa_1, \ldots, \kappa_m, \sigma_1, \ldots, \sigma_m, \rho \} \). Facts are distributed as follows:

- Every node \( \kappa_i \) is responsible for the facts Type(1), Start\(_1\)(i), Start\(_2\)(i), and all facts from \( \text{facts}(\mathcal{D}', U) \).
- Every node \( \sigma_i \) is responsible for the facts Type(2), Start\(_1\)(i), and all Start\(_2\)-facts, and all facts from \( \text{facts}(\mathcal{D}', U) \).
- Finally, node \( \rho \) is responsible for facts Type(3), \ldots, Type(m), and all facts over other relation names.

It is easy to see that \( P \) can be expressed by a polynomial number of rules and that \( Q \) and \( P \) can be computed in polynomial time. It remains to show that the described function is a reduction from Containment(BCQ\(^-\), BCQ\(^-\)) to Parallel-Complete(CQ\(^-\), \mathcal{P}_\text{rule}), and the corresponding variants of Parallel-Sound(CQ\(^-\), \mathcal{P}_\text{rule}) and Parallel-Correct(CQ\(^-\), \mathcal{P}_\text{rule}).

To this end, we show first that containment \( Q_1 \subseteq Q_2 \) implies parallel-completeness and parallel-soundness of query \( Q \) under policy \( P \), and that lack of containment, \( Q_1 \nsubseteq Q_2 \), implies that query \( Q \) is neither parallel-complete nor parallel-sound under policy \( P \).

In both directions, we will make use of the following easy observations:

**Claim 5.5.** Let \( I \) be an arbitrary instance and \( i \in \{1, 2\} \).
(1) Let $V$ be a valuation for $Q$, let $a \overset{\text{def}}{=} V(\ell_i)$, and $V_i$ be the restriction of valuation $V$ to variables in $Q_i$. If $V$ satisfies $Q$ on $I$, then $V_i$ satisfies $Q_i$ on $a^{-1}(I)$.

(2) Let $V_1, V_2$ be valuations for queries $Q_1, Q_2$. Let $a, b$ be data values such that $V_1$ and $V_2$ satisfy $Q_1$ and $Q_2$, respectively, on $a^{-1}(I)$, and such that Type($b$), Start$_1(a), \text{Start}_2(a) \in I$, and Stop($a$) $\notin I$. Then the valuation $W^b_a$, that agrees with $V_1$ and $V_2$ on all variables in $Q_1$ and $Q_2$, respectively, and maps $\ell_1, \ell_2 \mapsto a$, and $t \mapsto b$, satisfies $Q$ on $I$. $\square$

Let us now assume $Q_1 \subseteq Q_2$. To show that $Q$ is parallel-complete under $P$, let $V$ be a valuation that globally satisfies $Q$ on some arbitrary instance $I$ over $U$. Let $a \overset{\text{def}}{=} V(\ell_i)$ and $b \overset{\text{def}}{=} V(t)$. Satisfaction of $Q$ on $I$ by $V$ then particularly implies Type($b$), Start$_1(a), \text{Start}_2(a) \in I$, and Stop($a$) $\notin I$. Let $V_1$ be the satisfying valuation for $Q_1$ on instance $a^{-1}(I)$, as given by Claim 5.5.1. Since $Q_1 \subseteq Q_2$, there exists a valuation $V_2$ that satisfies $Q_2$ on instance $a^{-1}(I)$. By Claim 5.5.2, valuation $W \overset{\text{def}}{=} W^b_a$ satisfies $Q$ on $\pi_a(I) \cup \{\text{Type} (b), \text{Start}_1(a), \text{Start}_2(a)\}$. If $W(t) = 1$, then node $\kappa_a$ is responsible for these facts; if $W(t) = 2$, then node $\kappa_a$ is responsible for these facts; and otherwise, node $\rho$ is responsible for these facts. Hence, query $Q$ is parallel-complete under policy $P$.

The proof that $Q$ is parallel-sound under $P$ is similar. To this end, let $I$ be a global instance and $V$ be a valuation that satisfies $Q$ on the local instance $I_k$ of some node $\kappa \in N$. Let $a \overset{\text{def}}{=} V(\ell_i)$ and $b \overset{\text{def}}{=} V(t)$. By definition of $P$, we can infer Type($b$), Start$_1(a), \text{Start}_2(a) \in I$, and Stop($a$) $\notin I$ from satisfaction of $Q$ by $V$. Let $V_1$ be the satisfying valuation for $Q_1$ on instance $a^{-1}(I_k)$, as given by Claim 5.5.1. Since $Q_1 \subseteq Q_2$, there exists a valuation $V_2$ that satisfies $Q_2$ on instance $a^{-1}(I)$. By definition of $P$, we have $a^{-1}(I_k) = a^{-1}(I)$, and therefore $V_2$ satisfies $Q_2$ on $a^{-1}(I)$. Thus, by Claim 5.5.2, valuation $W \overset{\text{def}}{=} W^b_a$ satisfies $Q$ on $I$. Hence, query $Q$ is parallel-sound under policy $P$.

Now let $Q_1 \not\subseteq Q_2$ and let $I_1$ be an instance such that $Q_1(I_1) \neq \emptyset$, and $Q_2(I_1) = \emptyset$. Thanks to Lemma 4.5, we can assume that $I_1$ is over a domain of size at most $\text{varmax}(Q_1)$. Thanks to genericity, we can assume that the domain is a subset of $U$. Let $V_1$ be a valuation that satisfies query $Q_1$ on $I_1$. Furthermore, let $V_2$ be some consistent valuation for $Q_2$ and $I_2 \overset{\text{def}}{=} V_2(\text{pos}_{Q_2})$, which exists thanks to our assumption that $Q_2$ is satisfiable.

To show that $Q$ is not parallel-complete under $P$, we define

$$I \overset{\text{def}}{=} \alpha_1(I_1) \cup \alpha_2(I_2) \cup \{\text{Type} (1), \text{Start}_1(1), \text{Start}_2(1), \text{Stop}_2(2)\}.$$  

Then, valuation $V$, which maps all variables in $Q_1$ and $Q_2$ as $V_1$ and $V_2$, respectively, and $\ell_1 \mapsto 1$, $\ell_2 \mapsto 2$, and $t \mapsto 1$ satisfies query $Q$ on instance $I$. However, there is no locally satisfying valuation for $Q$. For a contradiction, assume existence of such a valuation $W$. Since Type($1$) and Start$_1(1)$ are the only Type- and Start$_1$-facts contained in the global instance, we have $W(t) = W(\ell_1) = 1$. By definition of $P$, this valuation can only be satisfying on node $\kappa_1$. Since only Start$_1(1) \in \text{loc-inst}_{P,i}(\kappa_1)$, Claim 5.5.1 implies existence of a satisfying valuation $W_2$ for $Q_2$ on $a^{-1}(I) = I_1$, which contradicts the choice of instance $I_1$. Hence, query $Q$ is not parallel-complete under policy $P$.

To show that $Q$ is also not parallel-sound under $P$, let instance

$$I \overset{\text{def}}{=} \alpha_1(I_1) \cup \alpha_2(I_2) \cup \{\text{Type} (2), \text{Start}_1(1), \text{Start}_2(1), \text{Start}_2(2), \text{Stop}_2(2)\}.$$  

Then, valuation $V$, which maps all variables in $Q_1$ and $Q_2$ as $V_1$ and $V_2$, respectively, and $\ell_1 \mapsto 1$, $\ell_2 \mapsto 2$, and $t \mapsto 2$ satisfies query $Q$ on the local instance $\text{loc-inst}_{P,i}(\kappa_1)$ of node $\kappa_1$, because this node is not responsible for fact Stop($2$). However, there is no globally satisfying valuation for $Q$. Towards a contradiction, assume existence of such a valuation $W$. Since Start$_2(1), \text{Start}_2(2)$ are the only Start$_2$-facts in the global instance, we have $W(\ell_2) = 1$ or $W(\ell_2) = 2$. The latter cannot hold, because the valuation then prohibits the present fact Stop($2$). The former implies, by Claim 5.5.1, a satisfying valuation $W_2$ for $Q_2$ on $a^{-1}(I) = I_1$, which contradicts the choice of instance $I_1$. Hence, query $Q$ is not parallel-sound under policy $P$. This
completes the proof that problem \textsc{Containment}(\texttt{BCQ}^-, \texttt{BCQ}^-) is reducible in polynomial time to \textsc{Parallel-Complete}(\texttt{CQ}^-, \mathcal{P}_{\text{rule}}) and its variants for parallel-soundness and parallel-correctness and thus shows coNEXPTIME-hardness via Corollary 4.3.

6 FULL CONJUNCTIVE QUERIES

In this section, we focus attention on full conjunctive queries in an attempt to lower the complexity of testing parallel-correctness. Requiring queries to be full is a very natural restriction that is known to have practical benefits. For example, the Hypercube algorithm, which describes an optimal way to compute CQs in a setting very similar to ours, completely ignores projections when shuffling data and only applies them when computing the query locally. The latter is possible, because correctness for the full-variant of a query is in a sense more strict than correctness for the query itself.

Formally, a (union of) conjunctive queries is called full if all variables of the body also occur in the head. We denote by \texttt{FCQ}^- and \texttt{UFCQ}^- the class of full \texttt{CQ}^- and full \texttt{UCQ}^- queries, respectively, and likewise for other fragments.

The presentation is similar to that of Sections 4 and 5. First, we establish the complexity of query containment. Then, we show that containment reduces to parallel-correctness (and variants). Finally, we obtain matching upper bounds.

The following theorem shows that, unlike for general conjunctive queries, the complexity of deciding containment for \texttt{FCQ}^- and \texttt{UFCQ}^- do not coincide.

**Theorem 4.** The following complexity results hold:

1. \textsc{Containment}(\texttt{FCQ}^-, \texttt{FCQ}^-) is in \texttt{P},
2. \textsc{Containment}(\texttt{FCQ}^-, \texttt{UFCQ}^-) is \texttt{coNP-complete},
3. \textsc{Containment}(\texttt{UFCQ}^-, \texttt{UFCQ}^-) is \texttt{coNP-complete}.

All these results also hold for queries with disequalities.

The proof is given in Section 6.1.

As one can reduce from \textsc{Containment}(\texttt{FCQ}^-, \texttt{UFCQ}^-) to parallel-soundness, completeness, and correctness, we obtain the following hardness results:

**Proposition 6.1.** For every \( \mathcal{P} \in \{\mathcal{P}_{\text{rule}}\} \cup \mathfrak{C}_{\text{poly}} \), the following problems are \texttt{coNP-hard}:

1. \textsc{Parallel-Sound}(\texttt{UFCQ}^-, \mathcal{P}),
2. \textsc{Parallel-Complete}(\texttt{UFCQ}^-, \mathcal{P}),
3. \textsc{Parallel-Correct}(\texttt{UFCQ}^-, \mathcal{P}).

The following theorem determines the complexity for the upper bounds:

**Theorem 5.** The following problems are \texttt{coNP-complete}:

1. \textsc{Parallel-Sound}(\texttt{UFCQ}^-, \mathcal{P}_{\text{rule}}),
2. \textsc{Parallel-Complete}(\texttt{UFCQ}^-, \mathcal{P}_{\text{rule}}),
3. \textsc{Parallel-Correct}(\texttt{UFCQ}^-, \mathcal{P}_{\text{rule}}).

All these results also hold for queries with disequalities.

**Proof.** Let \( Q \) be a UFCQ^- with \( m \) disjuncts, and \( P = (U, r\text{facts}_P) \in \mathcal{P}_{\text{rule}}. \)

1. From the definition of parallel-soundness, we immediately obtain that \( Q \) is not parallel-sound on \( P \) if and only if there is an instance \( I \subseteq \text{facts}(D, U) \), a fact \( f \), and a node \( \kappa \), where \( f \in Q(\text{loc-inst}_{P, I}(\kappa)) \) and \( f \notin Q(I) \). The former implies that there is a valuation \( V \) for some of

ACM Transactions on Computational Logic, Vol. 20, No. 3, Article 18. Publication date: June 2019.
the disjuncts of $Q$, say $Q_i$, which derives $f$ on $\text{loc-inst}_{P,1}(\kappa)$. The latter means that for all disjuncts $Q_i$, and valuations $V_j$ for $Q_j$, where $V_j(\text{head}_{Q_j}) = f$, $V_j$ fails to satisfy for $Q_j$ on $I$.

As $Q$ is full, for each disjunct $Q_j$ there is at most one eligible valuation $V_j$, which is uniquely defined by the head of $Q_j$, and the desired output fact $f$. We thus have to check only $m$ valuations $V_1, \ldots, V_m$. From this observation, it also follows that we can restrict the above condition to instances $I$ of size at most $\max_{j \in \{1, \ldots, m\}}(|\text{pos}_{Q_j}|) + m$. Indeed, given $I$, $\kappa$, and valuation $V$ (being the valuation showing $f \in Q(\text{loc-inst}_{P,1}(\kappa))$, obviously $V(\text{pos}_{Q_j}) \subseteq I$, and, for every $j \in \{1, \ldots, m\}$, either $V_j(\text{pos}_{Q_j}) \not\subseteq I$ or there are facts in $V_j(\text{neg}_{Q_j}) \cap I$. In the latter case, let $g_j$ be one of these facts. Now, we can construct an instance $I'$ of the desired size by simply taking the facts in $V(\text{pos}_{Q_i})$ and the chosen facts $g_j$. It is easy to see that $f \in Q(\text{loc-inst}_{P,1}(\kappa))$, while $f \not\in Q(I')$.

As $V(\text{pos}_{Q_i}) \subseteq r\text{facts}_P(\kappa)$ and $V(\text{neg}_{Q_i}) \cap r\text{facts}_P(\kappa) \cap I = \emptyset$ can be verified in polynomial time in $P$, it follows that verifying violation of parallel-soundness can be done in polynomial time in the size of the input by additionally providing an appropriate node $\kappa$, instance $I$, where $|I| \leq \max_{j \in \{1, \ldots, m\}}(|\text{pos}_{Q_j}|) + m$, and integer $i$ denoting disjunct $Q_i$.

2. From the definition of parallel-completeness, it follows that $Q$ is not parallel-complete on $P$ if and only if there is an instance $I \subseteq \text{facts}(D, U)$ and fact $f$, such that $f \in Q(I)$, and for all nodes $\kappa$, $f \not\in Q(\text{loc-inst}_{P,1}(\kappa))$. The former particularly implies, for some valuation $V$ and integer $i \in \{1, \ldots, m\}$, that $V(\text{head}_{Q_i}) = f$ and $V(\text{pos}_{Q_i}) \subseteq I$ and $V(\text{neg}_{Q_i}) \cap I = \emptyset$. The latter implies that for all integers $j \in \{1, \ldots, m\}$, the valuation $V_j$ defined by the head of $Q_j$ and fact $f$, does not satisfy on any of the nodes $\kappa$, that is, $V_j(\text{pos}_{Q_j}) \not\subseteq \text{loc-inst}_{P,1}(\kappa)$ or $V_j(\text{neg}_{Q_j}) \cap \text{loc-inst}_{P,1}(\kappa) \neq \emptyset$.

Again the size of $I$ can be bounded; in particular, it suffices to consider only instances of size at most $\max_{j \in \{1, \ldots, m\}}(|\text{pos}_{Q_j}|) + |N| \cdot m$, where $N$ denotes the network where $P$ is defined over. Indeed, for every instance $I$ as in the condition, we can construct an instance $I'$ containing all the facts in $V(\text{pos}_{Q_i})$ and for every node $\kappa$, and every $j \in \{1, \ldots, m\}$ either add nothing (when $V_j(\text{pos}_{Q_j}) \not\subseteq I$), or add some fact $g_j \in V_j(\text{neg}_{Q_j}) \cap I \cap r\text{facts}_P(\kappa)$.

By definition, the considered distribution policies have only polynomially many nodes in $n$, and verifying whether a fact is available on some node can be done in polynomial as well. Hence, the result follows.

3. As parallel-correctness for $Q$ under $P$ means $Q$ being parallel-sound and parallel-complete under $P$, it follows immediately from (1) and (2) that deciding parallel-correctness for $\text{UFCQ} \rightarrow^*$ and $\mathcal{P}_{\text{rule}}$ is in $\text{coNP}$. \hfill \square

6.1 Proof for Theorem 4

Theorem 4 follows from the hardness results in Proposition 6.2, and the upper bound in Proposition 6.3, which are given below:

**Proposition 6.2.** For full queries with negation and disequalities, the following results hold:

1. $\text{CONTAINMENT}(\text{FCQ} \rightarrow^*, \text{FCQ} \rightarrow^*)$ is in $\text{P}$,
2. $\text{CONTAINMENT}(\text{FCQ, UFCQ}^*)$ is $\text{coNP}$-hard,
3. $\text{CONTAINMENT}(\text{FCQ, UFCQ}^\neg)$ is $\text{coNP}$-hard.

**Proof.** 1. Let $Q_1, Q_2 \in \text{FCQ} \rightarrow^*$. We show that $Q_1 \subseteq Q_2$ if and only if $h(\text{pos}_{Q_1}) \subseteq \text{pos}_{Q_2}$ and $h(\text{neg}_{Q_1}) \subseteq \text{neg}_{Q_2}$, for the substitution $h$ that identifies the head relations,\footnote{In particular, containment implies existence of such a substitution despite possible multiple occurrences of the same variable in $\text{head}_{Q_2}$, as the following proof shows.} $h(\text{head}_{Q_1}) = \text{head}_{Q_2}$. It then immediately follows that $\text{CONTAINMENT}(\text{FCQ} \rightarrow^*, \text{FCQ} \rightarrow^*)$ is in $\text{P}$.

ACM Transactions on Computational Logic, Vol. 20, No. 3, Article 18. Publication date: June 2019.
To show the claim, let $Q_1 \subseteq Q_2$. Let $I^{-}$ be the minimal canonical database for $Q_1$, i.e., $I^{-}$ consists of the frozen atoms in $pos_{Q_1}$. By $I^{+}$, we denote the maximal canonical database for $Q_1$, i.e., $I^{+}$ contains every frozen atom over $vars(Q_1)$ (and the relations where $Q_1$ and $Q_2$ are defined over) that is not in $neg_{Q_1}$. By construction, $head_{Q_1} \in Q_1(I^{-})$ and $head_{Q_1} \in Q_1(I^{+})$, which implies by containment that $head_{Q_1} \in Q_2(I^{-})$ and $head_{Q_1} \in Q_2(I^{+})$. By fullness of $Q_2$, both are derived by the same valuation $V$. Now, the former implies $V(pos_{Q_1}) \subseteq I^{-} = pos_{Q_1}$, and the latter implies $V(neg_{Q_1}) \cap I^{+} = \emptyset$. Thus, $V(neg_{Q_1}) \subseteq neg_{Q_2}$. Hence, $V$ describes the desired substitution.

For the other direction, suppose that substitution $h$ has the desired properties. Let $I$ be an arbitrary instance, and $f \in Q_1(I)$. Thus, there is a valuation $V_1$, where $V_1(head_{Q_1}) = f$ and $V_1(pos_{Q_1}) \subseteq I$ and $V_1(neg_{Q_1}) \cap I = \emptyset$. Let $V_2 \defeq V_1 \circ h$. Then, $V_2(head_{Q_1}) = V_1(head_{Q_1}) = f$ and $V_2(pos_{Q_1}) \subseteq V_1(pos_{Q_1}) \subseteq I$ and $V_2(neg_{Q_1}) \subseteq V_1(neg_{Q_1})$. The latter implies $V_2(neg_{Q_2}) \cap I = \emptyset$, and thus $f \in Q_2(I)$.

2. The proof is by a very simple reduction from the (non-succinct) graph 3-colorability problem, which is known to be NP-complete. It asks for a given graph $G = (V, E)$ whether there is a coloring of the nodes in $G$, using only three colors, such that adjacent nodes have distinct colors. Let $G$ be an arbitrary input for the described problem. Without loss of generality, we assume that $V = \{1, \ldots, n\}$, for some $n$.

We construct queries $Q_1 \in FCQ$ and $Q_2 \in UFCQ^*$, and show $Q_1 \subseteq Q_2$ if and only if $G$ is not 3-colorable. Intuitively, this is done by letting $Q_1$ derive tuples that represent colourings of $V$ and by letting test query $Q_2$ whether a given coloring is not a valid 3-coloring for $G$. The latter is implemented as a union of queries where each disjunct detects a particular issue.

Queries $Q_1$ and $Q_2$ are defined over database schema $D \defeq \{L^{(1)}\}$, where $L$ is actually only needed to make the queries safe.

Query $Q_1$ is defined as:

$$H(x_1, \ldots, x_n) \leftarrow L(x_1), \ldots, L(x_n).$$

We note that each tuple that $H$ produces can be seen as a coloring of $V$.

Query $Q_2$ is the union of two sub-queries, $Q^{\text{inval}}$, which detects whether a coloring assigns the same color to some pair of adjacent nodes; and $Q^{>3}$, which detects whether a coloring uses more than three colors.

To this end, $Q^{\text{inval}}$ is the union of all queries

$$H(x_1, \ldots, x_n) \leftarrow L(x_1), \ldots, L(x_n), x_i = x_j,$$

with $(i, j) \in E$. We note that the equality atom could be avoided by identifying $x_i$ and $x_j$ in the head.

The subquery $Q^{>3}$, is defined as the union of queries:

$$H(x_1, \ldots, x_n) \leftarrow L(x_1), \ldots, L(x_n), \bigwedge_{1 \leq k < \ell \leq 4} x_{i_k} \neq x_{i_\ell},$$

for all quadruples $(i_1, i_2, i_3, i_4)$ over $\{1, \ldots, n\}$.

Clearly, $Q_1$ and $Q_2$ can be constructed in polynomial time in the size of $G$.

Correctness. We show that $Q_1 \not\subseteq Q_2$ if and only if $G$ is 3-colorable.

(If) Suppose that $G$ is 3-colorable. So, there is a coloring $C$ mapping the nodes of $G$ onto 3 colors $a, b, c$, such that no two adjacent nodes are assigned the same color.

Let $I$ be the instance with exactly the facts $L(a), L(b), L(c)$. Then the tuple $(C(1), \ldots, C(n))$ is in the query result of $Q_1$ over $I$. However, since $C$ uses only three colors and colors each pair of
adjacent nodes by two different colors, \((C(1), \ldots, C(n))\) is not in the query result of \(Q_2\) over \(I\). Hence, \(Q_1 \not\subseteq Q_2\).

(Only if) Suppose \(Q_1 \not\subseteq Q_2\). Thus, there is an instance \(I\) and a tuple \(t = (c_1, \ldots, c_m)\) in the query result of \(Q_1\) over \(I\), but not in the query result of \(Q_2\) over \(I\). Due to \(Q^{\geq 3}\), at most three different values occur in \(t\), and due to \(Q^{\text{invalid}}\), the mapping \(i \mapsto c_i\) is a valid coloring of \(G\). Therefore, \(G\) is 3-colorable.

3. The proof is by a very similar reduction from the graph 3-colorability problem as the previous one, trading inequalities for negation.

Let \(G = (V, E)\) be as in that proof. The idea is to use an additional relation \(\text{Eq}\) and to enforce that it is an equivalence relation on the relevant data values.

To this end, \(Q_1\) shall ensure that \(\text{Eq}\) is reflexive on values appearing in result tuples. It is defined as:

\[
H(x_1, \ldots, x_n) \leftarrow L(x_1), \ldots, L(x_n), \text{Eq}(x_1, x_1), \ldots, \text{Eq}(x_n, x_n).
\]

The query \(Q_2\) gets the additional task to ensure symmetry and transitivity. For this purpose, it uses two subqueries, \(Q^{\text{sym}}\) and \(Q^{\text{trans}}\), that filter out violations of symmetry and transitivity, respectively. Subquery \(Q^{\text{sym}}\) is the union of queries of the form

\[
H(x_1, \ldots, x_n) \leftarrow L(x_1), \ldots, L(x_n), \text{Eq}(x_i, x_j), \neg \text{Eq}(x_j, x_i),
\]

for all \(i, j \in \{1, \ldots, n\}\), and \(Q^{\text{trans}}\) is the union of queries of the form

\[
H(x_1, \ldots, x_n) \leftarrow L(x_1), \ldots, L(x_n), \text{Eq}(x_i, x_j), \text{Eq}(x_j, x_k), \neg \text{Eq}(x_i, x_k),
\]

for all \(i, j, k \in \{1, \ldots, n\}\).

The subquery \(Q^{\text{invalid}}\) of the previous reduction is redefined as the union of all queries

\[
H(x_1, \ldots, x_n) \leftarrow L(x_1), \ldots, L(x_n), \text{Eq}(x_i, x_j),
\]

with \((i, j) \in E\), and the subquery \(Q^{\geq 3}\), as the union of queries

\[
H(x_1, \ldots, x_n) \leftarrow L(x_1), \ldots, L(x_n), \bigwedge_{1 \leq k < \ell \leq 4} \neg \text{Eq}(x_{i_k}, x_{i_\ell}),
\]

for all quadruples \((i_1, i_2, i_3, i_4)\) over \(\{1, \ldots, n\}\). Finally, \(Q_2\) is \(Q^{\text{sym}} \cup Q^{\text{trans}} \cup Q^{\text{invalid}} \cup Q^{\geq 3}\).

In the correctness proof, the (if)-part is easy to adapt: in \(I\), the relation \(\text{Eq}\) just has to be chosen as the equality relation on the active domain.

(Only if) Suppose \(Q_1 \not\subseteq Q_2\). Thus, there is an instance \(I\) and a tuple \(t = (c_1, \ldots, c_m)\) in the query result of \(Q_1\) over \(I\), but not in the query result of \(Q_2\) over \(I\). Since \(t\) is neither a query result of \(Q^{\text{sym}}\) nor of \(Q^{\text{trans}}\), the relation \(\text{Eq}\) is symmetric and transitive, and thus an equivalence relation, with respect to the data values occurring in \(t\). Due to \(Q^{\geq 3}\), at most three different equivalence classes occur in \(t\); and due to \(Q^{\text{invalid}}\), the function that maps each vertex to its equivalence class is a valid coloring of \(G\). Therefore, \(G\) is 3-colorable.

\[
\text{Proposition 6.3. Containment}(UFCQ^{\wedge, \#}, UFCQ^{\wedge, \#}) \text{ is in coNP}.
\]

\textbf{Proof.} We observe that when \(Q \not\subseteq Q'\), there is an instance \(I\) and fact \(f\), such that \(f \in Q(I)\) and \(f \not\in Q'(I)\). In particular, by fullness of \(Q_1\) and \(Q_2\), there is such an instance of size at most

\[
l = \max_{i \in \{1, \ldots, n\}} \{|\text{pos}_{Q_i}|\} + m,
\]

where \(n\) denotes the number of disjuncts of \(Q\), and \(m\) the number of disjuncts in \(Q'\).

Indeed, to see this, let \(V\) and \(Q_i\) be the valuation and disjunct of \(Q\) where \(f\) is derived by on \(I\). Let \(J \overset{\text{def}}{=} V(\text{pos}_{Q_i})\). By fullness, it follows that for each disjunct \(Q_j\) of \(Q'\) there is at most one

ACM Transactions on Computational Logic, Vol. 20, No. 3, Article 18. Publication date: June 2019.
valuation \( V_j \) eligible to derive \( f \) for \( Q \). As, by choice of \( f \), \( V_j \) does not satisfy on \( I \), it must be that either \( V_j(\text{pos}_{Q}) \not\subseteq I \), or \( V_j(\text{neg}_{Q}) \cap I \neq \emptyset \). We ignore the former. In the latter case, we choose one fact from \( V_j(\text{neg}_{Q}) \cap I \) and add it to \( J \). One can now easily verify that \( J \) is as desired.

As the above shows that there always is a witnessing instance \( I \) of size \( \ell \), given \( I, V \), and \( Q \), as polynomial size certificate, we can easily verify that \( Q \) is indeed not contained in \( Q' \), by simply verifying that \( V \) indeed satisfies for disjunct \( Q_i \) of \( Q \) on \( I \); and for all \( m \) eligible valuations for conjuncts of \( Q' \), either not all required facts are in \( I \), or at least one of the prohibited facts is present.

\( \square \)

### 6.2 Proof of Proposition 6.1

The result follows from Proposition 6.2 and the reductions below.

**Proposition 6.4.** The following reducibility relations hold, for every \( \mathcal{P} \in \{ \mathcal{P}_{\text{rule}} \} \cup \mathcal{W}_{\text{poly}} $$:

1. Containment\((\text{FCQ}^-, \text{UFCQ}^-) \leq_p \text{Parallel-Sound}(\text{UFCQ}^-, \mathcal{P})\).
2. Containment\((\text{FCQ}^-, \text{UFCQ}^+) \leq_p \text{Parallel-Complete}(\text{UFCQ}^-, \mathcal{P})\).
3. Containment\((\text{FCQ}^-, \text{UFCQ}^+) \leq_p \text{Parallel-Correct}(\text{UFCQ}^-, \mathcal{P})\).

**Proof.** The idea underlying the following reductions is simple: Extend both \( \text{CQ}^- \)'s \( Q \) and \( Q' \) for the containment problem by a nullary atom \( \text{Global}() \) or its negation \( \neg\text{Global}() \), respectively, and combine them (by union) into a single query \( Q' \). By mapping the fact \( \text{Global}() \) onto an isolated node, the distribution policy then allows to control global and local derivability on behalf of \( Q \) and \( Q' \). Without loss of generality, we always assume that queries \( Q \) and \( Q' \) do not use the auxiliary relation \( \text{Global} \).

For a \( \text{CQ}^- \) \( Q \), let \( \text{Q}_{\text{Global}} \) and \( \text{Q}_{\neg\text{Global}} \) denote the queries obtained by adding the literal \( \text{Global}() \) or \( \neg\text{Global}() \) to \( Q \), respectively. For unions of \( \text{CQ}^- \)'s, this particularly means adding \( \text{Global}() \) or \( \neg\text{Global}() \) to every disjunct of \( Q \). The following identities can be easily proven to hold for every query \( Q \) and every instance \( I \):

\[
\text{Q}_{\text{Global}}(I \cup \{\text{Global}()\}) = Q(I),
\]

\[
\text{Q}_{\text{Global}}(I \setminus \{\text{Global}()\}) = \emptyset,
\]

\[
\text{Q}_{\neg\text{Global}}(I \cup \{\text{Global}()\}) = \emptyset,
\]

\[
\text{Q}_{\neg\text{Global}}(I \setminus \{\text{Global}()\}) = Q(I).
\]

We only argue the reductions for policy class \( \mathcal{P}_{\text{rule}} \). It is obvious that such a policy can also be represented by a policy from any class \( \mathcal{P}_{\text{npoly}} \). 

1. We start with Containment\((\text{FCQ}^-, \text{UFCQ}^-) \leq_p \text{Parallel-Sound}(\text{UFCQ}^-, \mathcal{P}_{\text{rule}})\).

Let \( Q \) and \( Q' \) be full \( \text{CQ}^- \)'s. We define a full \text{UFCQ}^- \( Q' \) and a policy \( P \) as follows. For this, let \( Q' \triangleq Q_{\neg\text{Global}} \cup Q'_{\text{Global}} \).

We construct a subset \( D \) of \( \text{dom} \), where \( |D| = |\text{vars}(Q_1)| \). Since the actual data values do not matter, we can choose those with the shortest representation length and thus also represent set \( D \) polynomially in the size of query \( Q_1 \). Now, \( P \) is defined as a distribution policy over network \( N = \{k_1, k_2\} \) that forwards every fact over \( D \) except \( \text{Global}() \) to \( k_1 \), and \( \text{Global}() \) to node \( k_2 \). As the described distribution policy can be straightforwardly expressed with a distribution policy in \( \mathcal{P}_{\text{fin}} \), the construction of both \( Q' \) and \( P \) can be done in polynomial time.

**Correctness.** It remains to show that \( Q \subseteq Q' \) if and only if \( Q' \) is parallel-sound under \( P \). The following observations are crucial for the correctness argument:
First, for each instance $I$ that contains $\text{Global}(I)$, $Q^*$ is equivalent to $Q'$:
\[
Q^*(I) = Q_{\text{Global}}(I) \cup Q'_{\text{Global}}(I) = \emptyset \cup Q'(I) = Q'(I).
\] (2)

Second, for each instance $I$ that does not contain $\text{Global}(I)$, $Q^*$ is equivalent to $Q$:
\[
Q^*(I) = Q_{\text{Global}}(I) \cup Q'_{\text{Global}}(I) = Q(I) \cup \emptyset = Q(I) = Q(I \cup \{\text{Global}(I)\}).
\] (3)

Further, as $\kappa_2$ can only contain the fact $\text{Global}(I)$, it follows that for each instance $I$, we have $Q(\text{loc-inst}_P(I(\kappa_2))) = \emptyset$.

(Only if) Assume $Q \subseteq Q'$. Let $I$ be an arbitrary subset of facts($P$). If $\text{Global}(I) \notin I$, then the local instance of node $\kappa_1$ is identical to the global instance, that is $\text{loc-inst}_P(I(\kappa_1)) = I$. Now, by definition of $P$ and thus also the result sets, $Q^*(\text{loc-inst}_P(I(\kappa_1))) = Q'(I)$. In particular, this implies $Q^*(\text{loc-inst}_P(I(\kappa_1))) \subseteq Q^*(I)$, that is, parallel-soundness of $Q^*$ under $P$ on instance $I$.

If $\text{Global}(I) \in I$, then the local instance is $\text{loc-inst}_P(I(\kappa_1)) = I \setminus \{\text{Global}(I)\}$. By Equations (2) and (3), $Q^*(\text{loc-inst}_P(I(\kappa_1))) \subseteq Q^*(I)$ if and only if $Q(I) \subseteq Q'(I)$, which holds by assumption of containment. Therefore, in both cases, query $Q^*$ is parallel-sound under policy $P$.

(If) For a proof by contraposition, assume $Q \not\subseteq Q'$. This implies existence of an instance $I$ where $Q(I) \not\subseteq Q'(I)$. Without loss of generality, we can assume $\text{dom}(I) \subseteq D$. The latter is a safe assumption, as from Lemma 4.5 it follows that an instance $J \subseteq I$ exists that preserves the desired property, and where $|\text{dom}(J)| \leq |D|$. Further, by genericity of $Q_1$ and $Q_2$, we can uniquely rename data values in $J$ to data values in $D$, resulting in an instance with the desired properties.

Additionally, we may safely assume that $\text{Global}(I) \in I$, because neither $Q$ nor $Q'$ refers to relation $\text{Global}$. This results in a local instance $\text{loc-inst}_P(I(\kappa_1)) = I \setminus \{\text{Global}(I)\}$. Again, by Equations (2) and (3), we conclude $Q^*(\text{loc-inst}_P(I(\kappa_1))) = Q(I) \not\subseteq Q'(I) = Q'(I)$, that is $Q^*$ is not parallel-sound under policy $P$ on instance $I$. Therefore, by contraposition, parallel-soundness of query $Q^*$ under policy $P$ and domain $D$ implies containment $Q \subseteq Q'$.

2. For Containment($\text{FCQ}^-, \text{FCQ}^-$) $\preceq_P \text{Parallel-Complete}(\text{UFCQ}^-, P_{\text{rule}})$ the proof is analogous. Policy $P$ is defined as before, while query $Q^* \overset{\text{def}}{=} Q_{\text{Global}} \cup Q'_{\text{Global}}$, i.e., atoms $\text{Global}(I)$ and $\neg\text{Global}(I)$, are swapped between the CQ’s compared to the soundness reduction.

Analogously to Equations (2) and (3), this leads to $Q^*(I) = Q(I)$ and $Q'(I) = Q'(I \cup \{\text{Global}(I)\})$, for every instance $I$ where $\text{Global}(I) \in I$ and $\text{Global}(I) \notin I$, respectively. Correctness then follows again from Lemma 4.5.

3. For Containment($\text{FCQ}^-, \text{FCQ}^-$) $\preceq_P \text{Parallel-Correct}(\text{UFCQ}^-, P_{\text{rule}})$ is also similar. Policy $P$ is defined as before while query $Q^* \overset{\text{def}}{=} Q_{\text{Global}} \cup Q'$. If $\text{Global}(I) \in I$, then $Q^*(I) = Q'(I)$ just as in the first reduction. However, if $\text{Global}(I) \notin I$, then $Q^*(I) = Q(I) \cup Q'(I)$.

(Only if) Assume $Q \subseteq Q'$. Thus, $Q \cup Q' \iff Q'$ imply $Q^* \equiv Q'$. Let $I$ be an arbitrary instance such that $\text{dom}(I) \subseteq D$. If $\text{Global}(I) \notin I$, then the local instance of node $\kappa_1$ is identical to the global instance (i.e., $\text{loc-inst}_P(I(\kappa_1)) = I$) by definition of $P$, and thus also $Q^*(\text{loc-inst}_P(I(\kappa_1))) = Q'(I)$.

If $\text{Global}(I) \in I$ is $\text{loc-inst}_P(I(\kappa_1)) = I \setminus \{\text{Global}(I)\}$. Thus, $Q^*(\text{loc-inst}_P(I(\kappa_1))) = Q'(\text{loc-inst}_P(I(\kappa_1))) = Q'(I) = Q'(I)$. Therefore, in both cases, query $Q^*$ is parallel-correct under policy $P$.

(If) We assume $Q \not\subseteq Q'$. This implies again by Lemma 4.5 and genericity of $Q$ and $Q'$, the existence of an instance $I$ where $Q(I) \not\subseteq Q'(I)$ and $\text{dom}(I) \subseteq D$. It follows $Q(I) \cup Q'(I) \neq Q'(I)$. We may safely assume that $\text{Global}(I) \in I$, because neither $Q$ nor $Q'$ refers to relation $\text{Global}$. Therefore, $\text{loc-inst}_P(I(\kappa_1)) = I \setminus \{\text{Global}(I)\}$. Again, by Equations (2) and (3), we conclude $Q^*(\text{loc-inst}_P(I(\kappa_1))) = Q'(I)$.
\(Q(I) \cup Q'(I) \neq Q'(I) = Q^*(I)\), that is, \(Q^*\) is not parallel-correct under \(P\) on instance \(I\), the desired contradiction. □

7 DISCUSSION

In this article, we continued the study of parallel-correctness initiated by Ameloot et al. [5] as a framework for reasoning about one-round evaluation algorithms for conjunctive queries under arbitrary distribution policies. Specifically, we considered the case with union and negation. While parallel-correctness for unions of conjunctive queries can be tested by examining properties of single valuations, just like in the union-free case, the latter no longer holds true when negation is present. Indeed, the presence of negations raises the complexity from \(\Pi_2^P\) to \(\text{coNEXPTIME}\). Since conjunctive queries with negation are no longer monotone, we considered the related problems of parallel-completeness and parallel-soundness as well and obtained the same bounds. Interestingly, when negation is present, containment of conjunctive queries can be reduced to parallel-correctness (and its variants), allowing the transfer of lower bounds. We prove that containment for conjunctive queries with negation is hard for \(\text{coNEXPTIME}\), which, to the best of our knowledge, is a novel result. In an attempt to lower complexity, we show that parallel-correctness for unions of full conjunctive queries with negation is \(\text{coNP}\)-complete.

There is quite a number of directions towards future work. While parallel-correctness for first-order logic is undecidable, it would be interesting to determine the exact frontier for decidability. As the considered problem is a static analysis problem that relates to the size of the queries and not to the size of the instances (at least in the setting of \(\mathcal{P}_{\text{rule}}\)), exponential lower bounds do not necessarily exclude practical application. It could still be interesting to identify settings that would make parallel-correctness tractable. Possibly independent of tractability considerations, such settings could incorporate bag semantics, integrity constraints, or specific classes (and representations) of distribution policies. We also plan to consider evaluation algorithms that use knowledge about the distribution policy to compute better query results, locally. Another direction for future work is to investigate transferability of parallel-correctness for conjunctive queries, as defined in Reference [5] in the presence of union and negation.

REFERENCES

[1] Apache Software Foundation. 2010. Spark. Retrieved from: http://spark.apache.org.
[2] Foto N. Afrati, Stavros S. Cosmadakis, and Mihalis Yannakakis. 1995. On datalog vs. polynomial time. J. Comput. Syst. Sci. 51, 2 (1995), 177–196. DOI: https://doi.org/10.1006/jcss.1995.1060
[3] Foto N. Afrati, Paraschos Koutris, Dan Suciu, and Jeffrey D. Ullman. 2012. Parallel skyline queries. In Proceedings of the International Conference on Database Theory (ICDT'12). 274–284.
[4] F. N. Afrati and J. D. Ullman. 2010. Optimizing joins in a map-reduce environment. In Proceedings of the International Conference on Extending Database Technology (EDBT’10), 99–110.
[5] Tom J. Ameloot, Gaetano Geck, Bas Ketsman, Frank Neven, and Thomas Schwentick. 2017. Parallel-correctness and transferability for conjunctive queries. J. ACM 64, 5 (2017), 36:1–36:38. DOI: https://doi.org/10.1145/3106412
[6] Tom J. Ameloot, Bas Ketsman, Frank Neven, and Daniel Zinn. 2014. Weaker forms of monotonicity for declarative networking: A more fine-grained answer to the CALM-conjecture. In Proceedings of the Principles of Database Systems (PODS'14). 64–75.
[7] Tom J. Ameloot, Frank Neven, and Jan Van den Bussche. 2013. Relational transducers for declarative networking. J. ACM 60, 2 (2013), 15. DOI: https://doi.org/10.1145/2450142.2450151
[8] Paul Beame, Paraschos Koutris, and Dan Suciu. 2013. Communication steps for parallel query processing. In Proceedings of the Symposium on Principles of Database Systems (PODS’13). 273–284.
[9] Paul Beame, Paraschos Koutris, and Dan Suciu. 2014. Skew in parallel query processing. In Proceedings of the Principles of Database Systems (PODS’14). 212–223.
[10] Ashok K. Chandra and Philip M. Merlin. 1977. Optimal implementation of conjunctive queries in relational data bases. In Proceedings of the Symposium on the Theory of Computing (STOC’77). 77–90.
[11] Shumo Chu, Magdalena Balazinska, and Dan Suciu. 2015. From theory to practice: Efficient join query evaluation in a parallel database system. In Proceedings of the International Conference on Management of Data (SIGMOD’15). 63–78.

[12] Carles Farré, Werner Nutt, Ernest Teniente, and Toni Urpi. 2007. Containment of conjunctive queries over databases with null values. In Proceedings of the 11th International Conference on Database Theory, ICDT’07). 389–403. http://dx.doi.org/10.1007/11965893_27

[13] Sumit Ganguly, Abraham Silberschatz, and Shalom Tsur. 1992. Parallel bottom-up processing of datalog queries. J. Log. Program. 14, 1 & 2 (1992), 101–126.

[14] Paraschos Koutris and Dan Suciu. 2011. Parallel evaluation of conjunctive queries. In Proceedings of the Symposium on Principles of Database Systems (PODS’11). 223–234.

[15] Alon Y. Levy and Yehoshua Sagiv. 1993. Queries independent of updates. In Proceedings of the International Conference on Very Large Data Bases (VLDB’93). 171–181.

[16] Marie-Laure Mugnier, Geneviève Simonet, and Michaël Thomazo. 2012. On the complexity of entailment in existential conjunctive first-order logic with atomic negation. Inform. Comput. 215 (2012), 8–31.

[17] Alan Nash and Bertram Ludäscher. 2004. Processing first-order queries under limited access patterns. In Proceedings of the 23rd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. 307–318. DOI: https://doi.org/10.1145/1055558.1055601

[18] Christos H. Papadimitriou and Mihalis Yannakakis. 1986. A note on succinct representations of graphs. Inform. Contr. 71, 3 (1986), 181–185. DOI: https://doi.org/10.1016/S0019-9958(86)80009-2

[19] Jeffrey D. Ullman. 2000. Information integration using logical views. Theoret. Comput. Sci. 239, 2 (2000), 189–210.

[20] Fang Wei and Georg Lausen. 2003. Containment of conjunctive queries with safe negation. In Proceedings of the 9th International Conference on Database Theory (ICDT 2003). 343–357. DOI: https://doi.org/10.1007/3-540-36285-1_23

[21] R. Xin, J. Rosen, M. Zaharia, M. Franklin, S. Shenker, and I. Stoica. 2013. Shark: SQL and rich analytics at scale. In Proceedings of the International Conference on Management of Data (SIGMOD’13).

[22] Daniel Zinn, Todd J. Green, and Bertram Ludäscher. 2012. Win-move is coordination-free (sometimes). In Proceedings of the International Conference on Database Theory (ICDT’12). 99–113.

Received November 2018; revised March 2019; accepted April 2019