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

Gaetano Geck[1], Bas Ketsman*[2], Frank Neven[3], and Thomas Schwentick[4]

1 TU Dortmund University, Dortmund, Germany
2 Hasselt University, Hasselt, Belgium; and
Transnational University of Limburg, Belgium/The Netherlands
3 Hasselt University, Hasselt, Belgium; and
Transnational University of Limburg, Belgium/The Netherlands
4 TU Dortmund University, Dortmund, Germany

Abstract

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 paper extends the study of the complexity of parallel-correctness and its constituents, parallel-soundness and parallel-completeness, to unions of conjunctive queries with and without negation. As a by-product it is shown that the containment problem for conjunctive queries with negation is coNEXPTIME-complete.

1998 ACM Subject Classification H.2.3 Query Languages, H.2.4 Distributed databases

Keywords and phrases Conjunctive queries, distributed evaluation

Digital Object Identifier 10.4230/LIPIcs.ICDT.2016.9

1 Introduction

Motivated by recent in-memory systems like Spark [7] and Shark [21], Koutris and Suciu introduced the massively parallel communication model (MPC) [15] 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 naïve setting, a query $\mathcal{Q}$ is evaluated by reshuffling the data over many servers, according to some distribution policy, and then computing $\mathcal{Q}$ at each server in a parallel but communication-free manner. A notable family of distribution policies is formed within the Hypercube algorithm [3, 9, 11]. A property of Hypercube distributions is that for any instance I, the central execution of $\mathcal{Q}(I)$ always equals the union of the evaluations of $\mathcal{Q}$ 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. [4] 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, equals the union of the evaluations of the query at each server under the given distribution policy. One of the main results of [4] is that deciding

* PhD Fellow of the Research Foundation – Flanders (FWO).

parallel-correctness for conjunctive queries (CQs) is Π_2^P-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 conjunctive query 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.

As union and negation are fundamental operators, we extend in this paper the study of parallel-correctness to unions of conjunctive queries (UCQ), conjunctive queries with negation ($\mathrm{CQ}^\neg$) and unions of conjunctive queries with negation ($\mathrm{UCQ}^\neg$). 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. Furthermore, since all monotone queries are parallel-sound, on this class parallel-correctness is equivalent to parallel-completeness.

We start by investigating parallel-correctness for UCQ. Interestingly, for a UCQ to be parallel-correct under a certain distribution policy it is not required that every disjunct is parallel-correct. We extend the characterisation for parallel-correctness in terms of minimal valuations for CQs to UCQs and thereby obtain membership in Π_2^P. The matching lower bound follows, of course, from the lower bound for CQs [4].

Next, we study parallel-correctness for (unions of) conjunctive queries with negation. Sadly, when negation comes into play, 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 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 $\mathrm{CQ}^\neg$ to parallel-correctness of $\mathrm{CQ}^\neg$ (and its two variants) and show that query containment for $\mathrm{CQ}^\neg$ is coNEXPTIME-complete. This is considerably different from what we thought was folklore knowledge of the community. Indeed, the Π_2^p-completeness result for query containment for $\mathrm{CQ}^\neg$ mentioned in [19] only seems to hold for fixed database schemas (or a fixed arity bound, for that matter). We note that Mugnier et al. [17] provide a Π_2^p upper bound proof for $\mathrm{CQ}^\neg$ 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 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 coNEXPTIME to Π_2^p if the database schema is fixed or the arity of relations is bounded, and to 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 inequalities.

Outline. This paper 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 4. We consider containment of conjunctive queries with negation in Section 5 and parallel-correctness together with its variants in Section 6. We discuss the restriction to full conjunctive queries in Section 7. We conclude in Section 8. Missing proof details can be found in the full version of this paper [14].

2 Related work

As mentioned in the introduction, Koutris and Suciu introduced the massively parallel communication model (MPC) [15]. 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, 15] and skyline queries [2] has been considered. Beame, Koutris and Suciu [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, Silberschatz, and Tsur [13] and is described in the context of map-reduce by Afrati and Ullman [3].

Ameloot et al. [4] 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 Π_2^p-complete for conjunctive queries. In addition, towards optimisation in MPC, they considered parallel-correctness *transfer*. Here, parallel-correctness transfers from $\mathcal{Q}$ to $\mathcal{Q}'$ when $\mathcal{Q}'$ is parallel-correct under every distribution policy for which $\mathcal{Q}$ is parallel-correct. The associated decision problem for conjunctive queries is shown to be Π_3^p-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. [5] (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, 5] 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 [16] 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 $\text{CQ}^\neg$ to be Π_2^p-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 Π_2^p-lowerbound is given by Farré et al. [12]. Based on [16], 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 $\text{UCQ}^\neg$ containment.

3 Definitions

3.1 Queries and instances

We assume an infinite set $\mathbf{dom}$ of data values that can be represented by strings over some fixed alphabet. By $\mathbf{dom}_n$ 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 $ar(R)$. We also write $R^{(k)}$ as a shorthand to denote that R is a relation of arity k. We call $R(\mathbf{t})$ a *fact* when R is a relation name and $\mathbf{t}$ a tuple over $\mathbf{dom}$ of appropriate arity. We say that a fact $R(\mathbf{t})$ is *over* a database schema $\mathcal{D}$ if $R \in \mathcal{D}$. For a subset $U \subseteq \mathbf{dom}$ we write $facts(\mathcal{D}, U)$ for the set of possible facts over schema $\mathcal{D}$ and U and by $facts(\mathcal{D})$ we denote $facts(\mathcal{D}, \mathbf{dom})$. A *(database) instance* I over $\mathcal{D}$ is a finite set of facts over $\mathcal{D}$. By $adom(I)$ we denote the set of data values occurring in I. A *query $\mathcal{Q}$ 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 π of **dom** and every instance I, $\mathcal{Q}(\pi(I)) = \pi(\mathcal{Q}(I))$. We say that $\mathcal{Q}$ is *contained* in $\mathcal{Q}'$, denoted $\mathcal{Q} \subseteq \mathcal{Q}'$ iff for all instances I, $\mathcal{Q}(I) \subseteq \mathcal{Q}'(I)$.

3.2 Unions of conjunctive queries with negation

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

$$T(\mathbf{x}) \leftarrow R_1(\mathbf{y_1}), \ldots, R_m(\mathbf{y_m}), \neg S_1(\mathbf{z_1}), \ldots, \neg S_n(\mathbf{z_n}), \beta_1, \ldots, \beta_p$$

where all $R_i(\mathbf{y}_i)$ and $S_i(\mathbf{z}_j)$ are atoms over $\mathcal{D}$, every β_i is an inequality of the form $s \neq s'$ where s, s' are distinct variables occurring in some $\mathbf{y}_i$ or $\mathbf{z}_j$, and $T(\mathbf{x})$ is an atom for which $T \notin \mathcal{D}$. Additionally, for safety, we require that every variable in $\mathbf{x}$ occurs in some $\mathbf{y}_i$ 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* $T(\mathbf{x})$ as $head_{\mathcal{Q}}$, to the set $\{R_1(\mathbf{y_1}), \ldots, R_m(\mathbf{y_m}), S_1(\mathbf{z_1}), \ldots, S_n(\mathbf{z_n})\}$ as $body_{\mathcal{Q}}$, and to the set $\{\beta_1, \ldots, \beta_p\}$ as $ineq_{\mathcal{Q}}$. Specifically, we refer to $\{R_1(\mathbf{y_1}), \ldots, R_m(\mathbf{y_m})\}$ as the positive atoms in $\mathcal{Q}$, denoted $pos_{\mathcal{Q}}$, and to $\{S_1(\mathbf{z_1}), \ldots, S_n(\mathbf{z_n})\}$ as the *negated* atoms of $\mathcal{Q}$, denoted $neg_{\mathcal{Q}}$. We denote by $vars(\mathcal{Q})$ the set of all variables occurring in $\mathcal{Q}$. We refer to the class of conjunctive queries with negation and inequalities by $\mathbf{CQ}^{\neg,\neq}$, its restriction to queries without inequalities, without negated atoms, and without both by $\mathbf{CQ}^{\neg}$, $\mathbf{CQ}^{\neq}$, and $\mathbf{CQ}$, respectively. As a shorthand we refer to queries from $\mathbf{CQ}^{\neg,\neq}$ as $\mathrm{CQ}^{\neg,\neq}$s and similarly for the other classes.

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

A valuation V *satisfies* $\mathcal{Q}$ on instance I if all facts required by V are in I while no fact prohibited by V is in I, that is, if $V(pos_{\mathcal{Q}}) \subseteq I$ and $V(neg_{\mathcal{Q}}) \cap I = \emptyset$. In that case, V *derives* the fact $V(head_{\mathcal{Q}})$. The *result of $\mathcal{Q}$ on instance I*, denoted $\mathcal{Q}(I)$, is defined as the set of facts that can be derived by satisfying valuations for $\mathcal{Q}$ on I.

A *union of conjunctive queries with negation and inequalities* is a finite union of $\mathrm{CQ}^{\neg,\neq}$s. That is, $\mathcal{Q}$ is of the form $\bigcup_{i=1}^{n} \mathcal{Q}_i$ where all subqueries $\mathcal{Q}_1, \ldots, \mathcal{Q}_n$ have the same relation name in their head atoms. We assume disjoint variable sets among different disjuncts in $\mathcal{Q}$. That is, $vars(\mathcal{Q}_i) \cap vars(\mathcal{Q}_j) = \emptyset$ for $i \neq j$ and, in particular, $vars(head_{\mathcal{Q}_i}) \neq vars(head_{\mathcal{Q}_j})$. By $varmax(\mathcal{Q})$ we denote the maximum number of variables that occurs in any disjunct of $\mathcal{Q}$. By $\mathbf{UCQ}^{\neg,\neq}$ we denote the class of unions of conjunctive queries with negation and inequalities and its fragments are denoted correspondingly.

A $\mathrm{CQ}^{\neg,\neq}$ is called *full* if all of its variables occur in its head. A $\mathrm{UCQ}^{\neg,\neq}$ is *full* if all its subqueries are full.

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

3.3 Networks, data distribution, and policies

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

We note that for some facts from $\mathit{facts}(\mathcal{D}, U)$ 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 $\mathit{facts}(\mathcal{D}, U)$ 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.

Given some policy $\boldsymbol{P}$ that is defined over a network $\mathcal{N}$, the *result* $[\mathcal{Q}, \boldsymbol{P}](I)$ *of the distributed evaluation of a query* $\mathcal{Q}$ *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,

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

In the decision problem for parallel correctness (to be formalised later), the input consists of a query $\mathcal{Q}$ and a distribution policy $\boldsymbol{P}$. 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 $\boldsymbol{f}$, a node κ and a policy $\boldsymbol{P}$, it is not too hard to find out whether κ is responsible for $\boldsymbol{f}$ under $\boldsymbol{P}$.

For UCQ$^{\neq}$s, 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. The reason is that testing parallel-correctness in this setting requires counter examples of size exponential in the size of the query which can not be succinctly represented by policies in $\mathcal{P}_{\text{fin}}$. We therefore introduce the class $\mathcal{P}_{\text{rule}}$ allowing for a more economic rule based description of policies. In particular, in $\mathcal{P}_{\text{rule}}$, 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 $\boldsymbol{P} = (U, \mathit{rfacts}_{\boldsymbol{P}})$ from $\mathcal{P}_{\text{fin}}$ are specified by an explicit enumeration of U and of all pairs $(\kappa, \boldsymbol{f})$ where $\kappa \in \boldsymbol{P}(\boldsymbol{f})$. A policy $\boldsymbol{P} = (U, \mathit{rfacts}_{\boldsymbol{P}})$ from $\mathcal{P}_{\text{rule}}$ 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 κ. The semantics of such a rule is as follows: for every substitution $\mu : \mathbf{var} \cup \mathbf{dom} \to \mathbf{dom}$ that maps variables to values from U and leaves constants from U unchanged, the node κ is responsible for the fact $\mu(A)$. A rule is a *fact rule* if its atom does not contain any variables, that is, $A = R(a_1, \dots, a_n)$, where $a_1, \dots, a_n \in U$. In particular, $\mathcal{P}_{\text{fin}} \subseteq \mathcal{P}_{\text{rule}}$.

▶ **Example 1.** Let distribution policy $\boldsymbol{P}$ over schema $\{\texttt{Rel}^{(3)}\}$ and network $\{\kappa_1, \kappa_2\}$ be given by $U = \{1, \dots, 10\}$ and the rules $\big(\texttt{Rel}(1, x, x), \kappa_1\big)$, $\big(\texttt{Rel}(2, x, y), \kappa_2\big)$. On global instance $I = \{\texttt{Rel}(1, 7, 7), \texttt{Rel}(1, 7, 8), \texttt{Rel}(2, 9, 8), \texttt{Rel}(2, 9, 9)\}$, policy $\boldsymbol{P}$ induces local instances $\textit{loc-inst}_{\boldsymbol{P},I}(\kappa_1) = \{\texttt{Rel}(1, 7, 7)\}$ and $\textit{loc-inst}_{\boldsymbol{P},I}(\kappa_2) = \{\texttt{Rel}(2, 9, 8), \texttt{Rel}(2, 9, 9)\}$. ☐

The most general classes of policies allow to specify policies by means of a 'test algorithm' with time bound ℓ^k, where ℓ is the length of the input and k some constant. Such an algorithm decides, for an input consisting of a node κ and fact $\boldsymbol{f}$, whether κ is responsible for $\boldsymbol{f}$.[1] A policy $\boldsymbol{P} = (U, \textit{rfacts}_{\boldsymbol{P}})$ from $\mathcal{P}^k_{\text{npoly}}$ is specified by a pair $(n, \mathcal{A}_{\boldsymbol{P}})$, where n is a natural number in unary representation and $\mathcal{A}_{\boldsymbol{P}}$ is a non-deterministic algorithm.[2] The universe U of $\boldsymbol{P}$ 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 which are represented by strings of length at most n, that is, $\mathcal{N} = \mathbf{dom}_n$. A node κ is responsible for a fact $\boldsymbol{f}$ if $\mathcal{A}_{\boldsymbol{P}}$, on input $(\kappa, \boldsymbol{f})$, has an accepting run of at most $|(\kappa, \boldsymbol{f})|^k$ steps. Clearly, each policy of $\mathcal{P}_{\text{fin}}$ can be described in $\mathcal{P}^2_{\text{npoly}}$. Let $\mathfrak{P}_{\text{npoly}}$ denote the set[3] $\{\mathcal{P}^k_{\text{npoly}} \mid k \geq 2\}$ of distribution policies and by $\mathfrak{P}$ the set $\{\mathcal{P}_{\text{fin}}, \mathcal{P}_{\text{rule}}\} \cup \mathfrak{P}_{\text{npoly}}$.

3.4 Parallel-correctness, soundness, and completeness

In this paper, we mainly consider the one-round evaluation algorithm for a query $\mathcal{Q}$ that first distributes (reshuffles) the data over the computing nodes according to $\boldsymbol{P}$, then evaluates Q in a parallel step at every computing node, and finally outputs all facts that are obtained in this way.[4] As formalised next, the one-round evaluation algorithm is correct (sound, complete) if the query $\mathcal{Q}$ is parallel-correct (parallel-sound, parallel-complete) under $\boldsymbol{P}$.

▶ **Definition 2.** Let $\mathcal{Q}$ be a query, I an instance, and $\boldsymbol{P}$ a distribution policy.

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

▶ **Definition 3.** A query $\mathcal{Q}$ is *parallel-correct (respectively, parallel-sound and parallel-complete) under distribution policy $\boldsymbol{P} = (U, \textit{rfacts}_{\boldsymbol{P}})$*, if $\mathcal{Q}$ is parallel-correct (respectively, parallel-sound and parallel-complete) on all instances $I \subseteq \textit{facts}(\mathcal{D}, U)$.

In [4], parallel-correctness is characterised in terms of minimal valuations as defined next:

▶ **Definition 4.** Let $\mathcal{Q}$ be a CQ. A valuation V for $\mathcal{Q}$ is *minimal* for $\mathcal{Q}$ if there exists no valuation V' for $\mathcal{Q}$ such that $V(\textit{head}_{\mathcal{Q}}) = V'(\textit{head}_{\mathcal{Q}})$ and $V'(\textit{body}_{\mathcal{Q}}) \subsetneq V(\textit{body}_{\mathcal{Q}})$.

[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 $\mathcal{P}^1_{\text{npoly}}$.
[4] We note that, since $\boldsymbol{P}$ 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.

The following lemma is key in obtaining the Π_2^p upper bound on the complexity of testing parallel-correctness for conjunctive queries:

▶ **Lemma 5** (Characterisation of parallel-correctness for CQs [4]). *A CQ $\mathcal{Q}$ is parallel-correct under distribution policy $\boldsymbol{P} = (U, \mathit{rfacts}_{\boldsymbol{P}})$ if and only if the following holds:*

For every minimal valuation V for $\mathcal{Q}$ over U, there is a node $\kappa \in \mathcal{N}$ such that $V(\mathit{body}_{\mathcal{Q}}) \subseteq \mathit{rfacts}_{\boldsymbol{P}}(\kappa)$. (C1)

▶ **Remark 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 $\mathcal{C}$ and $\mathcal{C}'$ of $\mathbf{UCQ}^{\neg,\neq}$ and classes $\mathcal{P}$ of distribution policies from $\{\mathcal{P}_{\text{fin}}, \mathcal{P}_{\text{rule}}\} \cup \mathfrak{P}_{\text{npoly}}$.

CONTAINMENT$(\mathcal{C}, \mathcal{C}')$:
Input: $\mathcal{Q} \in \mathcal{C}$ and $\mathcal{Q}' \in \mathcal{C}'$
Question: Is $\mathcal{Q} \subseteq \mathcal{Q}'$?

PARALLEL-SOUND$(\mathcal{C}, \mathcal{P})$:
Input: $\mathcal{Q} \in \mathcal{C}$, $\boldsymbol{P} \in \mathcal{P}$
Question: Is $\mathcal{Q}$ parallel-sound under $\boldsymbol{P}$?

PARALLEL-COMPLETE$(\mathcal{C}, \mathcal{P})$:
Input: $\mathcal{Q} \in \mathcal{C}$, $\boldsymbol{P} \in \mathcal{P}$
Question: Is $\mathcal{Q}$ parallel-complete under $\boldsymbol{P}$?

PARALLEL-CORRECT$(\mathcal{C}, \mathcal{P})$:
Input: $\mathcal{Q} \in \mathcal{C}$, $\boldsymbol{P} \in \mathcal{P}$
Question: Is $\mathcal{Q}$ parallel-correct under $\boldsymbol{P}$?

4 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 show below that parallel-completeness remains in Π_2^p. Hardness already follows from Π_2^p-hardness of PARALLEL-CORRECT$(\mathbf{CQ}, \mathcal{P}_{\text{fin}})$ [4].

As a UCQ is parallel-complete under a policy $\boldsymbol{P}$ when all its disjuncts are, it might be tempting to assume that this condition is also necessary. However, as the following example illustrates, this is not the case.

▶ **Example 7.** Let $\mathcal{Q} = \mathcal{Q}_1 \cup \mathcal{Q}_2$, where $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are the following CQs:

$$\begin{aligned} \mathcal{Q}_1 : \quad & H(x,x) && \leftarrow && R(x,x), \\ \mathcal{Q}_2 : \quad & H(y,z) && \leftarrow && R(y,z), S(y,z). \end{aligned}$$

Further, let $\boldsymbol{P}$ be the policy over network $\{\kappa_1, \kappa_2\}$ that maps facts $R(a,a)$ to node κ_1, for all $a \in \mathbf{dom}$, and all other R-facts and all S-facts to node κ_2.

We argue that $\mathcal{Q}$ is parallel-complete under $\boldsymbol{P}$ on all instances. Indeed, assume $H(a,b) \in \mathcal{Q}(I)$ for some instance I and $a, b \in \mathbf{dom}$. If $a \neq b$, only the valuation $\{y \mapsto a, z \mapsto b\}$ can derive $H(a,b)$. This means that $\{R(a,b), S(a,b)\} \subseteq I$. Furthermore, $\{R(a,b), S(a,b)\} \subseteq \mathit{rfacts}_{\boldsymbol{P}}(\kappa_2)$. Hence, $H(a,b) \in \mathcal{Q}(\mathit{loc\text{-}inst}_{\boldsymbol{P},I}(\kappa_2))$. If $a = b$, then $R(a,a) \in I$. So, $R(a,a) \in \mathit{rfacts}_{\boldsymbol{P}}(\kappa_1)$ and $H(a,a) \in \mathcal{Q}(\mathit{loc\text{-}inst}_{\boldsymbol{P},I}(\kappa_1))$. On the other hand, $\mathcal{Q}_2$ is not parallel-complete under $\boldsymbol{P}$ on instance $I = \{R(0,0), S(0,0)\}$. Indeed, $H(0,0) \in \mathcal{Q}_2(I)$ but $\mathcal{Q}_2\big(\mathit{loc\text{-}inst}_{\boldsymbol{P},I}(\kappa_1)\big) = \mathcal{Q}_2\big(\{R(0,0)\}\big) = \emptyset$ and $\mathcal{Q}_2\big(\mathit{loc\text{-}inst}_{\boldsymbol{P},I}(\kappa_2)\big) = \mathcal{Q}_2\big(\{S(0,0)\}\big) = \emptyset$. ☐

We recall from Section 3.2 that disjuncts in unions of conjunctive queries use disjoint variable sets and a valuation for $\mathcal{Q}$ is a valuation for exactly one disjunct. As formalised next, the notion of minimality for valuations given in Definition 4 naturally extends to $\mathbf{UCQ}^{\neq}$.

▶ **Definition 8.** Let $\mathcal{Q} = \bigcup_{i=1}^{n} \mathcal{Q}_i$ be a UCQ$^{\neq}$. A valuation V_i for $\mathcal{Q}_i$, with $i \in \{1, \dots, n\}$, is *minimal* for $\mathcal{Q}$, if for no $j \in \{1, \dots, n\}$ there is a valuation V_j for $\mathcal{Q}_j$, such that $V_j(head_{\mathcal{Q}_j}) = V_i(head_{\mathcal{Q}_i})$ and $V_j(body_{\mathcal{Q}_j}) \subsetneq V_i(body_{\mathcal{Q}_i})$.

▶ **Example 9.** Consider a simple UCQ$^{\neq}$ $\mathcal{Q} = \mathcal{Q}_1 \cup \mathcal{Q}_2$ where $\mathcal{Q}_1, \mathcal{Q}_2 \in \mathbf{CQ}^{\neq}$ are as follows:

$$\begin{array}{llll} \mathcal{Q}_1: & H(u,v) & \leftarrow & R(u,v), R(v,u), R(u,u), \\ \mathcal{Q}_2: & H(x,y) & \leftarrow & R(x,y), R(y,z), y \neq z. \end{array}$$

Valuation $V_2 \overset{\text{def}}{=} \{x \mapsto 0, y \mapsto 0, z \mapsto 1\}$ is not minimal for $\mathcal{Q}$ because valuation $V_1 \overset{\text{def}}{=} \{u \mapsto 0, v \mapsto 0\}$ derives the same fact $H(0,0)$ requiring only $\{R(0,0)\} \subsetneq \{R(0,0), R(0,1)\}$. Similarly, valuation $W_1 \overset{\text{def}}{=} \{u \mapsto 0, v \mapsto 1\}$, requiring $\{R(0,1), R(1,0), R(0,0)\}$, is not minimal for $\mathcal{Q}$ because valuation $W_2 \overset{\text{def}}{=} \{x \mapsto 0, y \mapsto 1, z \mapsto 0\}$ only requires $\{R(0,1), R(1,0)\}$. ☐

The notion of minimality leads to basically the same simple characterisation of parallel-completeness:

▶ **Lemma 10.** *A UCQ$^{\neq}$ $\mathcal{Q}$ is parallel-correct under distribution policy $\boldsymbol{P} = (U, rfacts_{\boldsymbol{P}})$ if and only if the following holds:*

For every minimal valuation V for $\mathcal{Q}$ over U, there is a node $\kappa \in \mathcal{N}$ such that $V(body_{\mathcal{Q}}) \subseteq rfacts_{\boldsymbol{P}}(\kappa)$. (*C1′*)

Proof. *(If)* Assume ($C1'$) holds. Because of monotonicity, we only need to show that $\mathcal{Q}(I) \subseteq \bigcup_{\kappa \in \mathcal{N}} \mathcal{Q}(loc\text{-}inst_{\boldsymbol{P},I}(\kappa))$ for every instance I. To this end, let $\boldsymbol{f}$ be an arbitrary fact that is derived by some valuation V for $\mathcal{Q}$ on I. Then, there is also a minimal valuation V' that is satisfying on I and which derives $\boldsymbol{f}$. Because of ($C1'$), there is a node $\kappa \in \mathcal{N}$ where all facts required by V' meet (cf. Remark 6). Hence, $\boldsymbol{f} \in \bigcup_{\kappa \in \mathcal{N}} \mathcal{Q}(loc\text{-}inst_{\boldsymbol{P},I}(\kappa))$, i.e. query $\mathcal{Q}$ is parallel-correct under policy $\boldsymbol{P}$.

(Only if) For a proof by contraposition, suppose that there is a minimal valuation V' for $\mathcal{Q}$ for which the required facts do *not* meet under $\boldsymbol{P}$. Consider the input instance $I = V'(body_{\mathcal{Q}})$. By definition of minimality, there is no valuation that agrees on the head variables and is satisfying for $\mathcal{Q}$ on a strict subset of $V'(body_{\mathcal{Q}})$. Therefore, $V'(head_{\mathcal{Q}})$ is in $\mathcal{Q}(I)$ but it is not derived on any node and thus query $\mathcal{Q}$ is not parallel-complete under policy $\boldsymbol{P}$. ◀

The characterisation in Lemma 10, in turn, can be used to prove a Π_2^p upper bound.

▶ **Lemma 11.** Parallel-Correct($\boldsymbol{UCQ}^{\neq}, \mathcal{P}$) *is in* Π_2^p, *for every* $\mathcal{P} \in \mathfrak{P}$.

Proof. It suffices to show that the complement of Parallel-Complete($\mathbf{UCQ}^{\neq}, \mathcal{P}^k_{\text{npoly}}$) is in Σ_2^p for arbitrary $k \geq 2$. Let $\boldsymbol{P} = (n, T)$ be a policy from $\mathcal{P}^k_{\text{npoly}}$. We have to consider only instances whose data values can be represented by strings of length n over networks whose nodes can be represented by strings of length n.

By Lemma 10, a query $\mathcal{Q}$ is not parallel-correct under distribution policy $\boldsymbol{P}$ if and only if there exists a minimal valuation V that satisfies $\mathcal{Q}$ on some instance I with $adom(I) \subseteq \mathbf{dom}_n$ such that no node in $\mathbf{dom}_n$ is responsible for all facts from $V(body_{\mathcal{Q}})$.

First, the algorithm non-deterministically guesses a valuation V, which can be represented by a string in length polynomial in $\mathcal{Q}$ and n. Subsequently, it checks for all valuations V', all nodes κ, and all strings x of polynomial length whether V' contradicts minimality of V (in which case the algorithm rejects the input) and, by use of algorithm T, whether node κ is not responsible for at least one fact from $V(body_{\mathcal{Q}})$ (if so, the algorithm continues, otherwise it rejects). All tests can be done in polynomial time. ◀

From [4] we know the following result.

▶ **Theorem 12** ([4]). Parallel-Correct($\boldsymbol{CQ}, \mathcal{P}_{fin}$) *is* Π_2^p*-complete.*

Together with Lemma 11 we get the following result.

▶ **Theorem 13.** Parallel-Correct($\boldsymbol{UCQ}^{\neq}, \mathcal{P}$) *is* Π_2^p*-complete, for every* $\mathcal{P} \in \mathfrak{P}$.

5 Containment of CQ$^\neg$ and UCQ$^\neg$

In this section, we establish the complexity of containment for $\mathbf{CQ}^\neg$ and $\mathbf{UCQ}^\neg$. We need these results to establish lower bounds on parallel-correctness and its constituents in the next section. Whereas containment for $\mathbf{CQ}$ has been intensively studied in the literature, the analogous problems for $\mathbf{CQ}^\neg$ and $\mathbf{UCQ}^\neg$ have hardly been addressed and seem to belong to folklore. In fact, we only found a reference of a complexity result for containment of $\mathbf{CQ}^\neg$ in [19], where a Π_2^p-algorithm for the problem is given, based on observations in [16], and the existence of a matching lower bound is mentioned. However, as we show below, although the problem is indeed in Π_2^p 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($\mathbf{BCQ}^\neg, \mathbf{UBCQ}^\neg$) is coNEXPTIME-hard by a reduction from the succinct 3-colorability problem and afterwards that Containment($\mathbf{BCQ}^\neg, \mathbf{UBCQ}^\neg$) can be reduced to Containment($\mathbf{BCQ}^\neg, \mathbf{BCQ}^\neg$). Here, $\mathbf{BCQ}^\neg$ and $\mathbf{UBCQ}^\neg$ denote the class of Boolean CQ$^\neg$s and unions of Boolean CQ$^\neg$s, respectively. Together this establishes that Containment($\mathbf{BCQ}^\neg, \mathbf{BCQ}^\neg$) and therefore also Containment($\mathbf{CQ}^\neg, \mathbf{CQ}^\neg$) are coNEXPTIME-hard.

▶ **Proposition 14.** Containment($\boldsymbol{BCQ}^\neg$, $\boldsymbol{UBCQ}^\neg$) *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 2ℓ 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 2ℓ Boolean inputs. We construct queries $\mathcal{Q}_1$ and $\mathcal{Q}_2$ such that $\mathcal{Q}_1 \not\subseteq \mathcal{Q}_2$ if and only if the graph described by C is 3-colorable.

Both queries are over schema $\mathcal{D}$, which consists of relation names `DomainValues`$^{(3)}$, `Bool`$^{(1)}$, `And`$^{(3)}$, `Or`$^{(3)}$, `Neg`$^{(2)}$, and `Label`$^{(\ell+1)}$. Intuitively, satisfaction of $\mathcal{Q}_1$ 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 $\mathcal{Q}_1$ 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, $\mathcal{Q}_1$ 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 $\texttt{Label}$ represent nodes together with their respective color (one can think of the representation of a node by ℓ-ary addresses over a ternary alphabet).
We define query $\mathcal{Q}_1$ as follows:

$$\begin{aligned} T() \leftarrow & \texttt{DomainValues}(w_0, w_1, w_2), \neg\texttt{DomainValues}(w_1, w_0, w_2), \\ & \neg\texttt{DomainValues}(w_2, w_1, w_0), \neg\texttt{DomainValues}(w_0, w_2, w_1), \\ & \texttt{Bool}(w_0), \texttt{Bool}(w_1), \texttt{Neg}(w_1, w_0), \texttt{Neg}(w_0, w_1), \\ & \texttt{And}(w_0, w_0, w_0), \texttt{And}(w_0, w_1, w_0), \texttt{And}(w_1, w_0, w_0), \texttt{And}(w_1, w_1, w_1), \\ & \texttt{Or}(w_0, w_0, w_0), \texttt{Or}(w_0, w_1, w_1), \texttt{Or}(w_1, w_0, w_1), \texttt{Or}(w_1, w_1, w_1). \end{aligned}$$

It is easy to see that $\mathcal{Q}_1$ enforces the conditions mentioned above.

In the following, we denote sequences $x_1, \dots, x_\ell$ of ℓ variables by $\boldsymbol{x}$.
We define $\mathcal{Q}_2$ as the union of the queries $\mathcal{Q}_2^1$ and $\mathcal{Q}_2^2$, where subquery $\mathcal{Q}_2^1$ is defined as:

$$\begin{aligned} T() \leftarrow & \texttt{Bool}(x_1), \texttt{Bool}(x_2), \dots, \texttt{Bool}(x_\ell), \texttt{DomainValues}(y_r, y_g, y_b), \\ & \neg\texttt{Label}(\boldsymbol{x}, y_r), \neg\texttt{Label}(\boldsymbol{x}, y_g), \neg\texttt{Label}(\boldsymbol{x}, y_b). \end{aligned}$$

Intuitively, $\mathcal{Q}_2^1$ can be satisfied in a database if for some node, represented by $\boldsymbol{x}$, there is no color.

Subquery $\mathcal{Q}_2^2$ 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 $\boldsymbol{y}$ and $\boldsymbol{z}$, respectively, there is an edge between u and v.

To this end, CIRCUIT uses the variables $y_1, \dots, y_\ell$, $z_1, \dots, z_\ell$, representing the input and, at the same time, the 2ℓ 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 $\texttt{And}(u_1, u_2, v_1)$. Likewise for OR- and NEG-gates.

Subquery $\mathcal{Q}_2^2$ is defined as:

$$T() \leftarrow \texttt{DomainValues}(w_0, w_1, w_2), \textsc{circuit}, \texttt{Label}(\boldsymbol{y}, u), \texttt{Label}(\boldsymbol{z}, u).$$

Intuitively, $\mathcal{Q}_2^2$ returns true when two nodes, witnessed to be adjacent by the circuit, have the same color.

Correctness of the reductions can be shown rather straightforwardly, as is done in the full version of this paper [14]. ◀

Next, we provide the above mentioned reduction.

▶ **Proposition 15.** CONTAINMENT($\boldsymbol{BCQ}^\neg$, $\boldsymbol{UBCQ}^\neg$) $\leq_p$ CONTAINMENT($\boldsymbol{BCQ}^\neg$, $\boldsymbol{BCQ}^\neg$) .

Proof. Let $\mathcal{Q}_1$ be in $\mathbf{BCQ}^\neg$ and $\mathcal{Q}_2 = \bigcup_{i=1}^m \mathcal{Q}_2^i$ be in $\mathbf{UBCQ}^\neg$ over some database schema $\mathcal{D}$. Recall our assumption, that each disjunct is defined over a disjoint set of variables. Next, we construct CQs $\mathcal{Q}_1'$ and $\mathcal{Q}_2'$ such that $\mathcal{Q}_1' \subseteq \mathcal{Q}_2'$ if and only if, $\mathcal{Q}_1 \subseteq \mathcal{Q}_2$.

We explain the intuition behind the reduction by means of an example. To this end, let $\mathcal{Q}_1$ be $H() \leftarrow A(x, y)$ and let $\mathcal{Q}_2$ be the $\mathcal{Q}_2^1 \cup \mathcal{Q}_2^2$, where $\mathcal{Q}_2^1$ is $H() \leftarrow A(u_1, v_1), B(u_1, v_1)$

and $\mathcal{Q}_2^2$ 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 $\mathcal{Q}_2'$ takes the following form:

$$H() \leftarrow \mathtt{Active}(x_0, x_1; \ell_1, \ell_2), \underbrace{\alpha(\ell_1, \mathcal{Q}_2^1)}_{\mathcal{Q}_{2,1}'}, \underbrace{\alpha(\ell_2, \mathcal{Q}_2^2)}_{\mathcal{Q}_{2,2}'},$$

where $\alpha(w, \mathcal{Q})$ denotes the modification of the body of $\mathcal{Q}$ by replacing every atom $\mathtt{R}(\boldsymbol{x})$ by $\mathtt{R}'(w, \boldsymbol{x})$. Both queries are defined over the schema $\mathcal{D}' = \{A'^{(3)}, B'^{(3)}, \mathtt{Active}^{(4)}\}$. Notice that $\mathcal{Q}_2'$ contains a concatenation of the disjuncts of $\mathcal{Q}_2$. 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, $body_{\mathcal{Q}_1'}$ (not shown) is constructed in such a way that when there is a satisfying valuation for $\mathcal{Q}_1'$ there are two different data values, say 0 and 1. So, an instance I over $\mathcal{D}$ 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 $\mathcal{Q}_1'$, there is an instance I_2 on which every disjunct of $\mathcal{Q}_2$ is true, and there is an instance I_1 on which $\mathcal{Q}_1$ is true. So, both $\mathcal{Q}_{2,1}'$ and $\mathcal{Q}_{2,2}'$ evaluate to true on I_2^0 when ℓ_1 and ℓ_2 are interpreted by label 0. However, for $\mathcal{Q}_1$ to be contained in $\mathcal{Q}_2$, we need that at least one of the disjuncts $Q_{2,1}'$ or $Q_{2,2}'$ evaluates to true over I_1^1, that is, when its labelling variable is interpreted as 1. Atom $\mathtt{Active}(x_0, x_1; \ell_1, \ell_2)$ will ensure that x_0 and x_1 correspond with the values 0 and 1, and that at least one of the labelling variables ℓ_1 or ℓ_2 is equal to 1. In other words, $\mathtt{Active}$ chooses which disjunct to activate over I_1. So, at least one disjunct of $\mathcal{Q}_2$ evaluates to true on the instance I_1 on which $\mathcal{Q}_1$ is satisfied.

The reduction is explained in more detail in the full version of this paper [14]. ◀

Combining Propositions 14 and 15 we get the following corollary:

▶ **Corollary 16.** CONTAINMENT($\boldsymbol{CQ}^{\neg}$, $\boldsymbol{CQ}^{\neg}$) *is* coNEXPTIME-*hard.*

The corresponding upper bounds hold also in the presence of inequalities and are shown by small model (i.e., counter-example) properties. To this end, we make use of a restricted monotonicity property of UCQ$^{\neg,\neq}$s which was already observed in Proposition 2.4 of [1]. 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.

▶ **Lemma 17** ([1]). *For $\mathcal{Q} \in \boldsymbol{UCQ}^{\neg,\neq}$, I an instance with a compatible schema, and D a set of data values, it holds that $\mathcal{Q}(I_{|D}) \subseteq \mathcal{Q}(I)$.*

Proof. Let $\boldsymbol{f} \in \mathcal{Q}(I_{|D})$ via a valuation V for a disjunct $\mathcal{Q}_i$ of $\mathcal{Q}$. Thus, $V(pos_{\mathcal{Q}_i}) \subseteq I_{|D} \subseteq I$. By definition, every variable x of $\mathcal{Q}_i$ occurs in a positive atom and therefore $V(x) \in D$. Thus, $V(neg_{\mathcal{Q}_i}) \cap I = V(neg_{\mathcal{Q}_i}) \cap I_{|D} = \emptyset$ and $\boldsymbol{f} \in \mathcal{Q}(I)$ as claimed. ◀

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

▶ **Lemma 18.** *Let $\mathcal{Q}_1, \mathcal{Q}_2 \in \boldsymbol{UCQ}^{\neg,\neq}$. If there is an instance I, where $\mathcal{Q}_1(I) \not\subseteq \mathcal{Q}_2(I)$, then there is also an instance $J \subseteq I$, where $\mathcal{Q}_1(J) \not\subseteq \mathcal{Q}_2(J)$, and $|adom(J)| \leq varmax(\mathcal{Q}_1)$.*

Proof. Let I be as in the lemma and let $\boldsymbol{f}$ be a fact with $\boldsymbol{f} \in \mathcal{Q}_1(I)$ and $\boldsymbol{f} \notin \mathcal{Q}_2(I)$. Let V be a valuation that derives $\boldsymbol{f}$ via some disjunct $\mathcal{Q}_1^i$ of $\mathcal{Q}_1$.

Let $D \overset{\text{def}}{=} adom(V(pos_{\mathcal{Q}_1^i}))$ and $J \overset{\text{def}}{=} I_{|D}$ the set of all facts in I using only values from $adom(V(pos_{\mathcal{Q}_i}))$. By definition, $|adom(J)| \leq varmax(\mathcal{Q}_1)$. Clearly, V is still a satisfying valuation for $\mathcal{Q}_1^i$ over J. However, by Lemma 17, $\boldsymbol{f} \notin \mathcal{Q}_2(J) = \mathcal{Q}_2(I_{|D})$. ◀

The upper bounds follow easily from Lemma 18.

▶ **Proposition 19.** *The following upper bounds hold:*
1. Containment($\boldsymbol{UCQ}^{\neg,\neq}$, $\boldsymbol{UCQ}^{\neg,\neq}$) *is in* coNEXPTIME.
2. *For every k, containment of $\boldsymbol{UCQ}^{\neg,\neq}$-queries over schemas with arity bound k is in Π_2^p.*

Proof. In both cases, we consider the complement of Containment($\mathbf{UCQ}^{\neg,\neq}$, $\mathbf{UCQ}^{\neg,\neq}$). Let $m \stackrel{\text{def}}{=} \mathit{varmax}(\mathcal{Q}_1)$.
1. A NEXPTIME algorithm, on input $\mathcal{Q}_1, \mathcal{Q}_2$, can simply guess an instance J with a domain of at most m elements and a fact $\boldsymbol{f}$, and verifies that $\boldsymbol{f} \in \mathcal{Q}_1(J)$ but $\boldsymbol{f} \notin \mathcal{Q}_2(J)$. For the latter tests, it can simply cycle, in exponential time, to all valuations over J for $\mathcal{Q}_1$ and $\mathcal{Q}_2$.
2. For a fixed arity bound, the minimal counter-example J is of size at most m^k. It can thus be guessed in polynomial time. That $\boldsymbol{f} \in \mathcal{Q}_1(J)$ can be verified non-deterministically. That $\boldsymbol{f} \notin \mathcal{Q}_2(J)$ can be verified by a universal computation in polynomial time. ◀

A claim of a Π_2^p upper bound for containment of CQs with negation can be found in [19]. It was not made clear there, that this claim assumes bounded arity of the schema. That the containment problem is Π_2^p-complete for schemas of bounded arity has been explicitly shown in [17]. Clearly, Proposition 19.2 follows directly and 19.1 is only a variation of it. From Proposition 19 and Corollary 16 the main result of this section immediately follows.

▶ **Theorem 20.** Containment($\boldsymbol{BCQ}^{\neg}$, $\boldsymbol{BCQ}^{\neg}$) *and* Containment($\boldsymbol{UCQ}^{\neg,\neq}$, $\boldsymbol{UCQ}^{\neg,\neq}$) *are* coNEXPTIME*-complete.*

Of course, the theorem also holds for all classes $\mathcal{C}$ of queries with $\mathbf{BCQ}^{\neg} \subseteq \mathcal{C} \subseteq \mathbf{UCQ}^{\neg,\neq}$.

6 Parallel-correctness: unions of conjunctive queries with negation

As mentioned in Section 4, 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 21.** Let $\mathcal{Q}$ be the following conjunctive query with negation:

$$\begin{aligned} H() \quad \leftarrow \quad & \texttt{Bool}(w_0, w_0), \texttt{Bool}(w_1, w_1), \texttt{Bool}(x_1, x_1), \ldots, \texttt{Bool}(x_n, x_n), \\ & \neg\texttt{Bool}(w_0, w_1), \neg\texttt{Rel}(x_1, \ldots, x_n). \end{aligned}$$

Let $\boldsymbol{P}$ be the policy defined over universe $U = \{0, 1\}$ and two-node network $\{\kappa_1, \kappa_2\}$, which distributes all facts except $\texttt{Rel}(0, \ldots, 0)$ to node κ_1 and only fact $\texttt{Rel}(0, \ldots, 0)$ to node κ_2.

Query $\mathcal{Q}$ is not parallel-sound under policy $\boldsymbol{P}$, but the smallest counter-example I is of exponential size as we argue next. Indeed, let $I \stackrel{\text{def}}{=} \{\texttt{Bool}(0, 0), \texttt{Bool}(1, 1)\} \cup \{\texttt{Rel}(a_1, \ldots, a_n) \mid (a_1, \ldots, a_n) \in \{0, 1\}^n\}$. 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 $\mathcal{Q}$ on instance $\textit{loc-inst}_{\boldsymbol{P},I}(\kappa_1) = I \setminus \{\texttt{Rel}(0, \ldots, 0)\}$ because neither $\texttt{Bool}(0, 1)$ nor $\texttt{Rel}(0, \ldots, 0)$ is contained in the local instance. Furthermore, there is no satisfying valuation W for $\mathcal{Q}$ on the global instance I because W would have to map each x_i to either 0 or 1 implying that $W\big(\texttt{Rel}(x_1, \ldots, x_n)\big) \in I$.

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 $\texttt{Bool}(w_0, w_0), \texttt{Bool}(w_1, w_1)$, and $\neg\texttt{Bool}(w_0, w_1)$ in query $\mathcal{Q}$ then implies existence of both facts $\texttt{Bool}(0, 0)$ and $\texttt{Bool}(1, 1)$ because variables w_0 and w_1 cannot be mapped onto the same data value.

Assume that fact $\texttt{Rel}(a_1, \ldots, a_n)$, 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 $\mathcal{Q}$ also globally, on instance I^*, and can therefore be no example against parallel-soundness, which contradicts our choice of I^*. Thus, $\texttt{Rel}(a_1, \ldots, a_n) \in I^*$, 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 22.** *For every class $\mathcal{P} \in \{\mathcal{P}_{rule}\} \cup \mathfrak{P}_{npoly}$ of distribution policies, the following problems are* coNEXPTIME*-complete.*
- Parallel-Sound$(\boldsymbol{UCQ}^{\neg}, \mathcal{P})$
- Parallel-Complete$(\boldsymbol{UCQ}^{\neg}, \mathcal{P})$
- Parallel-Correct$(\boldsymbol{UCQ}^{\neg}, \mathcal{P})$

Theorem 22 follow from Propositions 23 and 25 below. It also holds for $\mathbf{UCQ}^{\neg,\neq}$. It is easy to show that, when restricted to schemas with some fixed (but sufficiently large, for hardness) arity bound, all these problems are Π_2^p-complete.

6.1 Upper bounds

In this section, we show the upper bounds of Theorem 22, summarised in the following proposition.

▶ **Proposition 23.** Parallel-Sound$(\boldsymbol{UCQ}^{\neg,\neq}, \mathcal{P})$, Parallel-Complete$(\boldsymbol{UCQ}^{\neg,\neq}, \mathcal{P})$, *and* Parallel-Correct$(\boldsymbol{UCQ}^{\neg,\neq}, \mathcal{P})$ *are in* coNEXPTIME*, for every class $\mathcal{P} \in \mathfrak{P}$ of distribution policies. If the arity of schemas is bounded by some fixed number, these problems are in Π_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 24.** Let $\mathcal{Q} \in \mathbf{UCQ}^{\neg,\neq}$ and let $\boldsymbol{P}$ be an arbitrary distribution policy. Then the following statements hold:

1. If $\mathcal{Q}$ is not parallel-complete under $\boldsymbol{P}$, then there is an instance J over a domain with at most $\textit{varmax}(\mathcal{Q})$ elements such that $\mathcal{Q}$ is not parallel-complete on J under $\boldsymbol{P}$.
2. If $\mathcal{Q}$ is not parallel-sound under $\boldsymbol{P}$, then there is an instance J over a domain with at most $\textit{varmax}(\mathcal{Q})$ elements such that $\mathcal{Q}$ is not parallel-sound on J under $\boldsymbol{P}$.

Towards (1) let us assume that $\mathcal{Q}$ is not parallel-complete on some instance I under $\boldsymbol{P}$. Let V be a valuation of a disjunct $\mathcal{Q}_i$ of $\mathcal{Q}$ that derives a fact $\boldsymbol{f}$ globally that is not derived on any node of the network. Let $D \overset{\text{def}}{=} \textit{adom}(V(\textit{pos}_{\mathcal{Q}_i}))$ and $J \overset{\text{def}}{=} I_{|D}$. Clearly, $|D| \leq \textit{varmax}(\mathcal{Q})$ and V still derives $\boldsymbol{f}$ globally on instance J via $\mathcal{Q}_i$. On the other hand, for every node κ, $\mathcal{Q}\big(\textit{loc-inst}_{\boldsymbol{P},J}(\kappa)\big) = \mathcal{Q}\big(\textit{loc-inst}_{\boldsymbol{P},I}(\kappa)_{|D}\big) \subseteq \mathcal{Q}\big(\textit{loc-inst}_{\boldsymbol{P},I}(\kappa)\big)$, thanks to

Lemma 17. Therefore $\boldsymbol{f}$ is not derived on κ, and thus J witnesses the lack of parallel-completeness of $\mathcal{Q}$ under $\boldsymbol{P}$.

The proof of (2) is completely analogous. Given a counter-example I and a valuation V that derives a fact $\boldsymbol{f}$ on some node κ via $\mathcal{Q}_i$, for which $\boldsymbol{f}$ is not derived globally, we define $D \overset{\text{def}}{=} I_{|adom(V(pos_{\mathcal{Q}_i}))}$ and show that $J \overset{\text{def}}{=} I_{|D}$ is the desired counter-example.

An algorithm that tests the complement of parallel-completeness non-deterministically is described in the full version of this paper [14]. ◀

6.2 Lower bounds

The lower bounds stated in Theorem 22 follow from a polynomial time reduction from problem CONTAINMENT($\mathbf{BCQ}^{\neg}, \mathbf{BCQ}^{\neg}$), for which we showed coNEXPTIME-hardness in Section 5.

▶ **Proposition 25.** PARALLEL-COMPLETE($\boldsymbol{CQ}^{\neg}, \mathcal{P}_{rule}$), PARALLEL-SOUND($\boldsymbol{CQ}^{\neg}, \mathcal{P}_{rule}$), *and* PARALLEL-CORRECT($\boldsymbol{CQ}^{\neg}, \mathcal{P}_{rule}$) *are* coNEXPTIME-*hard.*

Proof. Interestingly, all three results are shown by the *same* reduction from decision problem CONTAINMENT($\mathbf{BCQ}^{\neg}, \mathbf{BCQ}^{\neg}$).

The basic idea for this reduction is very simple: it combines both queries $\mathcal{Q}_1, \mathcal{Q}_2 \in \mathbf{BCQ}^{\neg}$ of the given containment instance into a single query $\mathcal{Q} \in \mathbf{BCQ}^{\neg}$ and infers an appropriate distribution policy $\boldsymbol{P}$. To emulate separate derivation for both queries in the combined query, an activation mechanism is used that resembles the proof of Proposition 15. In this fashion, the two queries can 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. Let thus $\mathcal{Q}_1, \mathcal{Q}_2 \in \mathbf{BCQ}^{\neg}$ be queries over some schema $\mathcal{D}$ and let $m \overset{\text{def}}{=} \max\{varmax(\mathcal{Q}_1), varmax(\mathcal{Q}_2)\}$. Without loss of generality, we assume the variable sets of $\mathcal{Q}_1$ and $\mathcal{Q}_2$ to be disjoint. We will also assume in the following that both $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are satisfiable. This is the case (for $\mathcal{Q}_1$) if and only if $pos_{\mathcal{Q}_1} \cap neg_{\mathcal{Q}_1} = \emptyset$ and can therefore be easily tested in polynomial time. If one of the test fails, some appropriate constant instance of PARALLEL-COMPLETE($\mathbf{CQ}^{\neg}, \mathcal{P}_{\text{rule}}$) or one of the other problem variants, respectively, can be computed.

We define a (Boolean) query $\mathcal{Q} \in \mathbf{BCQ}^{\neg}$ and a policy $\boldsymbol{P} \in \mathcal{P}_{\text{rule}}$ over domain $\{1, \dots, m\}$ that can be computed from $\mathcal{Q}_1$ and $\mathcal{Q}_2$ in polynomial time. The schema for $\mathcal{Q}$ is $\mathcal{D}' \overset{\text{def}}{=} \{R'^{(k+1)} \mid R^{(k)} \in \mathcal{D}\}$. That is, each relation name R of $\mathcal{D}$ occurs as R' in $\mathcal{D}'$ with an arity incremented by one. Additionally, $\mathcal{Q}$ uses relation names $\texttt{Type}$, $\texttt{Start}_1$, $\texttt{Start}_2$, and $\texttt{Stop}$, which we assume not to occur in schema $\mathcal{D}$. Besides the variables of $\mathcal{Q}_1$ and $\mathcal{Q}_2$, query $\mathcal{Q}$ uses variables ℓ_1, ℓ_2, t.

We use the function α, defined in the proof of Proposition 15, which adds its first parameter as first component to every tuple in its second parameter and translates relation names R into R'. In Proposition 15, the first parameter was always a variable and the second a set of atoms, but we use α also for a data value as first and a set of facts as second parameter in the obvious way. We write α_a^{-1} 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, $\pi_a(I) \overset{\text{def}}{=} \alpha\big(a, \alpha_a^{-1}(I)\big)$ is the restriction of I to all facts with a in their first component.

The combined query $\mathcal{Q}$ has $head_{\mathcal{Q}} \overset{\text{def}}{=} H()$ and body

$$\begin{aligned} body_{\mathcal{Q}} \overset{\text{def}}{=}\; & \alpha(\ell_1, body_{\mathcal{Q}_1}) \cup \alpha(\ell_2, body_{\mathcal{Q}_2}) \\ \cup\; & \underbrace{\{\texttt{Type}(t), \texttt{Start}_1(\ell_1), \texttt{Start}_2(\ell_1), \texttt{Start}_2(\ell_2)\}}_{\mathcal{A}} \cup \underbrace{\{\neg\texttt{Stop}(\ell_1), \neg\texttt{Stop}(\ell_2)\}}_{\mathcal{A}^{\neg}}. \end{aligned}$$

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

- Every node κ_i is responsible for the facts $\mathtt{Type}(1), \mathtt{Start}_1(i), \mathtt{Start}_2(i), \mathtt{Stop}(i)$, and all facts from $\mathit{facts}(\mathcal{D}', U)$.
- Every node σ_i is responsible for the facts $\mathtt{Type}(2), \mathtt{Start}_1(i), \mathtt{Stop}(i)$, all $\mathtt{Start}_2$-facts, and all facts from $\mathit{facts}(\mathcal{D}', U)$.
- Finally, node ρ is responsible for facts $\mathtt{Type}(3), \ldots, \mathtt{Type}(m)$, and *all* facts over other relation names.

It is easy to see that $\boldsymbol{P}$ can be expressed by a polynomial number of rules and that $\mathcal{Q}$ and $\boldsymbol{P}$ can be computed in polynomial time. In the full version of this paper [14], we show that the described function is indeed the desired reduction. ◀

7 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 which 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 $\mathbf{FCQ}^{\neg,\neq}$ and $\mathbf{UFCQ}^{\neg,\neq}$ the class of full $\mathrm{CQ}^{\neg,\neq}$ and full $\mathrm{UCQ}^{\neg,\neq}$ queries, respectively, and likewise for other fragments.

The presentation is similar to that of Section 5 and 6. 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 $\mathrm{FCQ}^{\neg}$ and $\mathrm{UFCQ}^{\neg}$ do not coincide.

▶ **Theorem 26.**
1. Containment($\boldsymbol{FCQ}^{\neg}, \boldsymbol{FCQ}^{\neg}$) *is in* P*;*
2. Containment($\boldsymbol{FCQ}^{\neg}, \boldsymbol{UFCQ}^{\neg}$) *is* coNP*-complete; and*
3. Containment($\boldsymbol{UFCQ}^{\neg}, \boldsymbol{UFCQ}^{\neg}$) *is* coNP*-complete.*

All these results also hold for queries with inequalities.

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

▶ **Proposition 27.** Parallel-Sound($\boldsymbol{UFCQ}^{\neg}, \mathcal{P}$), Parallel-Complete($\boldsymbol{UFCQ}^{\neg}, \mathcal{P}$), *and* Parallel-Correct($\boldsymbol{UFCQ}^{\neg}, \mathcal{P}$) *are* coNP*-hard, for every* $\mathcal{P} \in \{\mathcal{P}_{rule}\} \cup \mathfrak{P}_{npoly}$.

The following theorem determines the complexity for the upper bounds:

▶ **Theorem 28.** *The following problems are* coNP*-complete:*
1. Parallel-Sound($\boldsymbol{UFCQ}^{\neg}, \mathcal{P}_{rule}$)*;*
2. Parallel-Complete($\boldsymbol{UFCQ}^{\neg}, \mathcal{P}_{rule}$)*;*
3. Parallel-Correct($\boldsymbol{UFCQ}^{\neg}, \mathcal{P}_{rule}$)*.*

The result also holds for queries with inequalities.

8 Discussion

In this paper, we continued the study of parallel-correctness initiated by Ameloot et al. [4] 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. Consequently, we obtained that deciding parallel-correctness for unions of conjunctive queries remains in Π_2^p, while the analog problem in the presence of negation is hard for 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 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 coNP-complete.

There are 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 [4] in the presence of union and negation.

References

1 Foto N. Afrati, Stavros S. Cosmadakis, and Mihalis Yannakakis. On datalog vs. polynomial time. *J. Comput. Syst. Sci.*, 51(2):177–196, 1995. `doi:10.1006/jcss.1995.1060`.

2 Foto N. Afrati, Paraschos Koutris, Dan Suciu, and Jeffrey D. Ullman. Parallel skyline queries. In *International Conference on Database Theory (ICDT 2012)*, pages 274–284, 2012. `doi:10.1145/2274576.2274605`.

3 Foto N. Afrati and Jeffrey D. Ullman. Optimizing joins in a map-reduce environment. In *Extending Database Technology (EDBT 2010)*, pages 99–110, 2010. `doi:10.1145/1739041.1739056`.

4 Tom J. Ameloot, Gaetano Geck, Bas Ketsman, Frank Neven, and Thomas Schwentick. Parallel-correctness and transferability for conjunctive queries. In *Principles of Database Systems (PODS 2015)*, pages 47–58. ACM, 2015.

5 Tom J. Ameloot, Bas Ketsman, Frank Neven, and Daniel Zinn. Weaker forms of monotonicity for declarative networking: a more fine-grained answer to the CALM-conjecture. In *Principles of Database Systems (PODS 2014)*, pages 64–75, 2014.

6 Tom J. Ameloot, Frank Neven, and Jan Van den Bussche. Relational transducers for declarative networking. *J. ACM*, 60(2):15, 2013. `doi:10.1145/2450142.2450151`.

7 Apache spark. URL: `http://spark.apache.org`.

8 Paul Beame, Paraschos Koutris, and Dan Suciu. Communication steps for parallel query processing. In *Principles of Database Systems (PODS 2013)*, pages 273–284, 2013.

9 Paul Beame, Paraschos Koutris, and Dan Suciu. Skew in parallel query processing. In *Principles of Database Systems (PODS 2014)*, pages 212–223, 2014.

10 Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Symposium on Theory of Computing (STOC 1979)*, pages 77–90, 1977.

11 Shumo Chu, Magdalena Balazinska, and Dan Suciu. From theory to practice: Efficient join query evaluation in a parallel database system. In *ACM SIGMOD Conference*, pages 63–78, 2015.

12 Carles Farré, Werner Nutt, Ernest Teniente, and Toni Urpí. Containment of conjunctive queries over databases with null values. In *International Conference on Database Theory (ICDT 2007)*, pages 389–403, 2007. `doi:10.1007/11965893_27`.

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

14 Gaetano Geck, Bas Ketsman, Frank Neven, and Thomas Schwentick. Parallel-correctness and containment for conjunctive queries with union and negation. *CoRR*, abs/1512.06246, 2015. URL: `http://arxiv.org/abs/1512.06246`.

15 Paraschos Koutris and Dan Suciu. Parallel evaluation of conjunctive queries. In *Principles of Database Systems (PODS 2011)*, pages 223–234, 2011.

16 Alon Y. Levy and Yehoshua Sagiv. Queries independent of updates. In *International Conference on Very Large Data Bases (VLDB 1993)*, pages 171–181, 1993.

17 Marie-Laure Mugnier, Geneviève Simonet, and Michaël Thomazo. On the complexity of entailment in existential conjunctive first-order logic with atomic negation. *Inf. Comput.*, 215:8–31, 2012.

18 Christos H. Papadimitriou and Mihalis Yannakakis. A note on succinct representations of graphs. *Information and Control*, 71(3):181–185, 1986. `doi:10.1016/S0019-9958(86)80009-2`.

19 Jeffrey D. Ullman. Information integration using logical views. *Theoretical Computer Science*, 239(2):189–210, 2000.

20 Fang Wei and Georg Lausen. Containment of conjunctive queries with safe negation. In *International Conference on Database Theory (ICDT 2003)*, pages 343–357, 2003.

21 Reynold S. Xin, Josh Rosen, Matei Zaharia, Michael J. Franklin, Scott Shenker, and Ion Stoica. Shark: SQL and rich analytics at scale. In *ACM SIGMOD Conference*, 2013.

22 Daniel Zinn, Todd J. Green, and Bertram Ludäscher. Win-move is cordination-free (sometimes). In *International Conference on Database Theory (ICDT 2012)*, pages 99–113, 2012.