# *Reasoning about Distributed Relational Data and Query Evaluation*

**Dissertation**

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Gaetano Geck

Dortmund
2019

Tag der mündlichen Prüfung: 19.11.2019
Dekan: Prof. Dr.-Ing. Gernot A. Fink

Gutachter:
Prof. Dr. Thomas Schwentick
Prof. Dr. Luc Segoufin

# Abstract

Large data sets are often stored distributedly to increase the reliability of systems and the efficiency of query evaluation in them. While some query operators—like selections and projections—are intrinsically conform with parallel evaluation, others—like joins—demand specific distribution patterns.

For relational databases, a common approach to evaluate queries in parallel relies on the use of rather simple distribution patterns for binary joins and the computation of the query result according to some query plan, operator by operator. Often, this requires the redistribution of large intermediate results (possibly larger than the input and/or output) and thus may lead to unnecessary long processing times. Thus, especially in the last decade, more elaborate distribution patterns that depend on the whole query have been studied and shown to allow more efficient query evaluation in several cases by reducing the amount of communication between servers.

Ameloot et al. have described a setting where query evaluation is studied for a broad range of distribution patterns. Their work focuses on problems to decide whether a query can be evaluated correctly under a given distribution pattern. More particularly, they have considered two problems: *parallel correctness*, where the pattern is specified explicitly, and *parallel-correctness transfer*, where the pattern is known to be appropriate for another query.

This thesis comprises the author's contributions to the complexity-theoretical investigation of these problems for conjunctive queries (and extensions thereof). These contributions complement the main characterisations and some additional complexity results by Ameloot et al. Furthermore, this thesis contains some new characterisations for 'polarised' queries.

Via the characterisations, parallel correctness and parallel-correctness transfer can be translated into questions on the co-occurrences of certain facts, induced by the query, on some server. Such questions and others can be modelled by *distribution dependencies*, a variant of the well-known tuple- and equality-generating dependencies. Modelling via these constraints allows a more general description of distribution patterns in distributed relational data.

The third contribution of this thesis is the study of the implication problem for distribution dependencies, providing lower and upper bounds for some fragments.

# Acknowledgements

This thesis is the result of a 'journey' spanning several years. Over the time, many people have crossed my path, several have accompanied me for a while and some even all the way. I owe all of them a debt of gratitude but will explicitly mention only those that had a more direct impact on my life as a student and, later, as a research assistant.

Studying computer science has been my dream from the age of 14 on, or so. Soon, I realised that being a good student is good prerequisite for being a good computer scientist and, more importantly, that being a good student is much, much simpler if you have *good teachers*. Looking back, I have been very lucky to find some, here at the Technical Univserity of Dortmund—and already earlier, in school. I am particularly indebted to two of them.

Andreas Hamerla has been my computer science teacher from class 11 to 13. I still remember how I have been hoping—back then, even praying—to get a good teacher before, not knowing that I would get the best. Very sadly, he died far too early, only a few years later. I am deeply grateful for all the answers he gave me (and the questions he provided me with). His kindness and patience have been a guidance for my own teaching.

Thomas Schwentick has been the lecturer of several courses during my bachelor and master studies and, afterwards, my advisor. He is surely the one who taught me the most; not only about theoretical computer science but also about teaching, organising, communicating and more. I am deeply grateful to him for giving me the opportunity to work in his group. His accuracy and his strive for perfection have been (and will remain) a guidance for all my work.

A few other people also deserve special thanks. First and foremost, I thank Frank Neven and Bas Ketsman for the fruitful collaboration with Thomas and me over many years. I can hardly think of a team as friendly and committed as this. Frank and Thomas, in particular, are a dream team of professors for a PhD student in theoretical computer science to work with—at least that's how they appeared to me. I am happy that I could profit so many times from their insights, ideas and questions. Thanks also to all the other kind persons from Hasselt I have met, Tom Ameloot and Brecht Vandevoort in particular.

Most of the time, however, I did not spent working abroad but in Dortmund. Thus, I have always felt very lucky about the good atmosphere in our working group and

our chair in general. For that, I thank all former and current members. I learned from all of them. Special thanks go to Christopher Spinrath, for many helpful remarks on parts of this thesis, for sharing my desire for automatisation and for being a great 'office mate' in general; to Nils Vortmeier, for his good company, particularly in the beginning—in our first years as research assistants; and to Thomas Zeume, for letting me participate in ILTIS, a neat and useful distraction.

I thank Luc Segoufin for spending his time to review this thesis and, generally, all designated members for their willingness to participate in the committee.

Finally, I thank my friends and family. With a certain lack of modesty, I guess that they would have supported me on other paths as well. Nevertheless, their indulgence and attention have helped me a lot. My warmest thanks go to all of them. Among my friends, I thank Andrej, Christian, Daniel, Kevin, Michael and Nina (in alphabetical order) in particular—for the time I could spend with them. In my family, my mother—who bought my first computer despite the narrow budget—and my grandpa—who drove me when I could barely walk—deserve special thanks, also for many reasons more.

This thesis is dedicated to my siblings—Gina, Anaximander and Carlotta.

# Contents

# 1. Introduction

Modern societies rely on data. Vast amounts of data are collected, stored and processed with a velocity that increases from day to day. Data is recorded by a myriad of sensors and it is produced by billions of users—sometimes intentionally, sometimes unwittingly; the boundaries are blurred.

Whether we are aware of it or not, the collection and storage of data ultimately aims at their analysis.[1] From the perspective of database theory, the *analysis of data* can be viewed as the *evaluation of queries* on a database. Storing and processing very large volumes of data is a non-trivial task and *distributed* databases have been used and studied as a means to cope with it—already in the latter decades of the previous century.

Naturally, this abstract description comprises a multitude of possible choices. A few central questions related to these choices are the following. Is the data highly structured or unstructured? Where is it stored? What kind of queries can be formulated? How are they evaluated?

Next, we describe the specific setting that is studied in this thesis (Section 1.1), introduce the main algorithmic problems that we consider (Section 1.2) and, lastly, give a quick overview of the contributions of the author (Section 1.3).

## 1.1. Setting

While scientists and engineers have provided several answers to each of the questions above, in this work, we consider basically *one* setting: relational data that is distributed over multiple servers organised in a shared-nothing architecture. The data is queried via logical formalisms like conjunctive queries—possibly with union, disequalities and (atomic) negation—and the result is computed in a single round. Of course, this choice has an influence on the questions that are posed and the answers that are given, and thus asks for some justification.

Practically, relational databases are a commercial success story. Theoretically, they are well-studied, offering a rich body of literature. We assume that the reader

[1] The analysis is usually guided by varying intents: it can help to influence consumers and voters, to optimise logistics and transport or to expand scientific knowledge (amongst others). These prospects raise lots of questions—ethical, political and juridical in particular—, which we neglect here despite their importance.

has a basic acquaintance with the relational database model and refer to standard textbooks for a broader view [AHV95, SKS05].

Large data sets are commonly distributed over multiple servers such that results can be computed on parts of them in parallel. In modern data centres, dozens, hundreds or even thousands of servers are connected via high-speed networks [SB18]. For our theoretical investigation, we abstract from the details of the network architecture and build on the model of *massively parallel communication* (MPC). The MPC model, initially studied by Beame, Koutris and Suciu [KS11, BKS13], follows earlier approaches to model parallel computations like Valiant's *bulk-synchronous parallel* model [Val90] but abstracts from low-level details. This abstraction moves the focus to the communication between the servers, which is nowadays an important limiting factor.

Computation in the MPC model is round-based. In each round, a communication phase is followed by a computation phase. In the communication phase, servers can freely exchange data via broadcasting. Afterwards, in the computation phase, each server processes the data on its own—without any further communication. Another round can start when all servers have finished the computation phase, which imposes a 'global barrier' on the whole computation process. In particular, the slowest servers in each round determine the overall computation time.

Despite its simplicity, the MPC model captures essential aspects of modern systems like Google's *MapReduce* [DG08], Apache's *Spark* [ZCF+10] or *Shark* [XRZ+13] that support the processing of key-value pairs or SQL queries.

To speed up the computation process, it is desirable to minimise, first, the number of rounds and, second, within each round, the amount of communication and the load—the amount of data—per server. Usually, there is a trade-off between both goals (an improvement in one dimension can result in a setback in the other) but they provide a guideline. Ensuring a good balance of the load ratios of the servers is particularly challenging if the data is 'skewed'—that is, when certain data values ('heavy hitters') occur unproportionally often—because the distribution is usually guided by one value per item, the *key*, and thus independent of other items.

Several results for query evaluation in the MPC model have been obtained recently for computations that can be accomplished in *one* round. We mention only a few initial results, as an indicator that even this very restricted setting poses interesting challenges. The *Hypercube* algorithm was presented by Afrati and Ullman [AU10] as a method to compute multi-way—instead of just binary—joins and later proven to be load-optimal for single-round computations over skew-free data by Beame, Koutris and Suciu [BKS14]. Furthermore, the latter authors designed a worst-case optimal one-round evaluation algorithm (for skewed or skew-free data) [KBS16]. These studies centre around conjunctive queries, or fragments thereof, which we also choose as a starting point.

Generally, computations with multiple rounds are often fine from a practical

perspective and presumably also offer additional interesting theoretical questions and answers to be discovered, or more efficient algorithms to be devised. We emphasise that the restriction to single-round computations in this thesis is not meant to favour them but rather results from organisational considerations: one has to *start* somewhere.

## 1.2. Problems

This thesis focuses on variants of three static analysis problems for distributed relational data. The first two problems, *parallel correctness* and *parallel-correctness transfer*, deal with the question whether a given query can be evaluated correctly in a single computation phase of the MPC model—with varying degrees of knowledge about the previous communication phase. The third problem, the *implication* problem for distribution tgds, addresses the question of algorithmical reasoning about the existence of relational facts on nodes in a more general way, independent of queries.

The majority of the results in this work are complexity-theoretic results, usually providing matching lower and upper bounds for the mentioned problems under various restrictions of their inputs (query classes, distribution formalisms, ...). Such bounds can serve as an indication of the optimality of algorithms and can highlight certain aspects that contribute to the complexity. This, in turn, facilitates the identification of special cases where the problems can be solved more efficiently in theory—and possibly also in practice.

### 1.2.1. Parallel correctness

Suppose a user wants a query $Q$ to be evaluated by a distributed database management system. We differentiate between two approaches: either the query itself provides a specification $\mathbb{P}$, called *policy*, of the way the data is distributed, or a suitable specification is derived automatically. In the former case, the user is responsible for the correctness of the evaluation process, in the latter case, it should be guaranteed by the derivation mechanism.

Following the previously mentioned approaches for query evaluation in the MPC model, a policy $\mathbb{P}$ is defined over a network of servers $k_1, \ldots, k_n$ and specifies, for each fact, a subset of nodes responsible for this fact. For an instance $\mathcal{G}$, which is a set of facts, the policy thus describes instances $\mathbb{L}(1), \ldots, \mathbb{L}(n)$ of the nodes. The naïve distributed evaluation of the query in one round is simply the union over the query results on each node.

The *instance-specific* version of parallel correctness asks, for a given query $Q$ and a policy $\mathbb{P}$, whether the naïve distributed query evaluation yields the same result as

the standard evaluation on the complete instance. Formally, this holds if equation

$$Q(\mathcal{G}) = Q(\mathbb{L}(1)) \cup \ldots \cup Q(\mathbb{L}(n))$$

is satisfied. The static version of parallel correctness generalises this question to *every instance* $\mathcal{G}$ (over an induced schema and domain).

As the set equality indicates, parallel correctness comprises two aspects: all facts derived on the complete instance are derived on at least one node and only such facts are derived there. These properties are called *parallel completeness* and *parallel soundness*. For monotonic queries, parallel correctness is identical to parallel completeness (since they are trivially parallel-sound), for other queries parallel soundness has to be studied on its own.

Coming back to our motivation, deciding parallel correctness algorithmically has at least two potential benefits. First, it can free users from reasoning about the correctness of self-designed policies themselves. Second, it can guide the automatic selection or derivation of policies. In particular, if query $Q$ is parallel-correct under the currently applied policy $\mathbb{P}$, this allows to avoid the costly redistribution of data *without inspection of the (very large) data set*. Therefore, parallel correctness can be seen as a basic property in the context of optimised distributed query evaluation.

### 1.2.2. Parallel-correctness transfer

As discussed above, the question for parallel correctness can be posed in a scenario where a policy $\mathbb{P}$ is given and a new query $Q'$ is to be evaluated. Since policy $\mathbb{P}$ is usually derived from or designed for a query $Q$, this easily leads to a more general question: given that query $Q$ has been evaluated correctly, is query $Q'$ guaranteed to be evaluated correctly under $\mathbb{P}$ too?

This question is reflected in the notion of parallel-correctness transfer. We say that parallel-correctness *transfers* from query $Q$ to query $Q'$ if the latter query is parallel-correct under every policy where the former query is parallel-correct under. This property is equivalent to the following implication.

$$\forall\, \mathbb{P}\colon Q \text{ is parallel-correct under } \mathbb{P} \implies Q' \text{ is parallel-correct under } \mathbb{P}$$

Note that both policy *and* instance are universally quantified—while at least the policy is specified explicitly as input for the variants of the parallel correctness problem.

Compared with parallel correctness, transfer of parallel-correctness allows the optimisation of distributed query evaluation on a higher level. If multiple queries $Q_1, \ldots, Q_r$ are to be evaluated and parallel-correctness transfers from, say, query $Q_1$ to query $Q_r$, then it suffices to find a policy for queries $Q_1, \ldots, Q_{r-1}$ and to reuse it

for the evaluation of $Q_r$. This is similar to other typical approaches in 'multi-query optimisation' [LÖ18] and is particularly useful in analytical tasks where the same queries are evaluated repeatedly, over changing data. In this sense, the potential of parallel-correctness transfer resembles that of query containment.

### 1.2.3. Distribution dependencies and their implication problem

Deciding parallel correctness and transfer of parallel correctness essentially requires to reason about occurrences and co-occurrences of sets of facts—whose size and structure is mainly determined by the query (or queries) under consideration. Clearly, this is not a peculiarity of these properties but rather a fundamental aspect in distributed data management, where the placement of data determines the reliability of the system and is, moreover, critical for its scalability and the performance of query processing in particular. Thus, a question of utmost importance is the following: *how should data be replicated and partitioned over the servers?*

Despite the importance of this question and decades of research on distributed databases, typical placement strategies remained comparatively simple for a long time. Horizontal or vertical fragmentation (or hybrid variants thereof) in combination with range or hash partitioning are prominent examples [ÖV11]. Unfortunately, query evaluation under such placement strategies usually requires a communication phase for each binary join, which is why these strategies often have a negative impact on the overall computation time. Therefore, more sophisticated strategies have increasingly attracted attention in the recent past—and, sometimes, were even proven to be optimal. Examples include co-partitioning and (single or multiple) hypercubes [BKS17, KS17, SCH+18, SVS+13, ZBS15].

Nevertheless, the study of frameworks that allow reasoning about data dependencies in general has a long tradition in computer science. Tuple- and equality-generating dependencies, for instance, are an elegant and versatile formalism to model constraints on *relational* data. They have been introduced to unify several types of constraints, the practically important key constraints, more general functional dependencies and inclusion dependencies in particular [Nic78, BV81, Fag82]. Central to reasoning about constraints is the *implication* problem: given a set $\Sigma$ of constraints and a constraint $\tau$,

$$\text{does every data set that satisfies } \Sigma \text{ also satisfy } \tau?$$

For tuple- and equality-generating dependencies (*tgds* and *egds*), the *chase* is an invaluable algorithmic tool to decide the implication problem. The undecidability of the implication problem for unrestricted tgds and egds has motivated researchers to study restricted variants. Meanwhile, several fragments have been studied in an attempt to discover the boundaries of decidability and complexity. Often, these

fragments are defined by syntactic restrictions on the dependencies like weak acyclicity, weak guardedness, stickiness, wardednesss and many more [BGPS19, CGK13, CGP10, FKMP05].

However, seemingly few attempts have been made so far to use these frameworks explicitly for distributed data. Notable exceptions are, for instance, the *data exchange* setting [ABLM14] and *Webdamlog* [AAS13, ABGA11, MSAM15].

We try to bring both strands of research—placement strategies for distributed data and dependency frameworks—closer together. To this end, we adapt tuple- and equality-generating dependencies to model knowledge about distributed relational facts. The possibility to reason about distributed data may optimise query evaluation (by avoiding reshuffling, by improving the precision of estimations on query execution plans etc.).

## 1.3. Contributions

This thesis is the result of a scientific collaboration with my advisor and some colleagues from the 'Databases and Theoretical Computer Science' (*DBTI*) research group at the Hasselt University, Belgium, which has begun in 2014.

Most of the results presented here have been published in major conference and articles before, or are accepted for publication. Prof. Dr Thomas Schwentick, my advisor, and Prof. Dr Frank Neven, from DBTI, have contributed to all these publications. Furthermore, Dr Bas Ketsman and—in the beginning—Dr Tom Ameloot, both in DBTI then, have contributed to the work on parallel correctness and parallel-correctness transfer.

Since these publications have emerged from complementary research activities, a coherent presentation of the author's own results makes it necessary to refer to some of their contributions in particular. To make a clear distinction between contributions of the author and those of others, the latter are marked with a star (Definition*, Proposition*, Theorem* etc.).

The following is an overview over the author's contributions in this thesis. More details are given in the 'bibliographical remarks' sections at the end of Chapters 3, 4 and 5.

The majority of the contributions in Chapter 3 are complexity-theoretical results, which provide lower or upper bounds—often both, and matching—to the central problems around parallel correctness for various query classes. In particular, we show that parallel correctness is $\Pi_2^p$-complete for conjunctive queries (without negation) and that it is coNEXPTIME-complete for conjunctive queries with negation. Motivated by the high complexity for queries with negation in general, the fragments of 'full'

and 'polarised' CQs with negation are considered in detail and shown to be coNP-hard and $\Pi_2^p$-complete, respectively.

For polarised queries, previous key notions are extended in a non-trivial way in order to yield characterisations of parallel completeness and—for the first time—for parallel soundness. The picture is rounded up by a $\Pi_2^p$-completeness result. These results have *not* been published before.

For parallel-correctness transfer, studied in Chapter 4, most contributions are complexity-theoretical again. In particular, transfer is shown to be $\Pi_3^p$-complete for conjunctive queries (without negation) but $\Pi_2^p$-complete for 'strongly-minimal' queries with disequalities.

Building on the characterisations in the previous chapter, transfer of parallel *soundness* is then studied systematically for the first time. For polarised queries (with negation), it is characterised and shown to be $\Pi_3^p$-complete too. These results have *not* been published before.

In Chapter 5, we introduce *distribution tgds* and *egds* as a variant of classical tgds and egds, show how they can be used to model typical distribution patterns and study the associated implication problem. More particularly, we identify decidable fragments with different complexities, ranging from NP over PSPACE to EXPTIME. For each of the fragments, which are defined by simple syntactic criteria, we provide matching lower and upper bounds. The latter rely, of course, on an adaptation of the *chase*.

# 2. Preliminaries

In this chapter, we fix notations for basic concepts and provide some general definitions. More specific definitions are given where needed, in the following chapters.

Let $\mathbb{N}$ denote the set of all positive integers and let $\mathbb{N}_0 \stackrel{\text{def}}{=} \mathbb{N} \uplus \{0\}$ denote the set of natural numbers. Symbol $\uplus$ denotes the disjoint union of sets and is often used to represent a partition $\{S_1, \ldots, S_r\}$ in the form $S_1 \uplus \cdots \uplus S_r$. The powerset of a set $A$ is denoted $2^A$ and the cardinality of $A$ is denoted $|A|$.

A mapping $f : X \to Y$ over a finite domain $X = \{x_1, \ldots, x_n\}$ is sometimes specified in the form $\{x_1 \mapsto y_1, \ldots, x_n \mapsto y_n\}$, meaning that $f(x_1) = y_1, \ldots, f(x_n) = y_n$ holds. For mappings $f_1 : X_1 \to Y_1$ and $f_2 : X_2 \to Y_2$ over disjoint domains $X_1$ and $X_2$, the combined mapping $f : X_1 \cup X_2 \to Y_1 \cup Y_2$ that behaves like $f_1$ on $X_1$ and like $f_2$ on $X_2$ is denoted $f_1 \cup f_2$. Furthermore, $f[x_1 \mapsto y_1, \ldots, x_n \mapsto y_n]$ denotes the mapping $f' : X \cup \{x_1, \ldots, x_n\} \to Y \cup \{y_1, \ldots, y_n\}$ derived from $f : X \to Y$ such that $f'(x_i) = y_i$ for every $i \in \{1, \ldots, n\}$ and $f'(x) = f(x)$ for every other $x \in X$. In particular, if $\{x_1, \ldots, x_n\}$ is disjoint from $X$, then $f'$ is an extension of $f$.

For a binary relation $\mathbb{R} \subseteq A \times B$ and sets $A'$ and $B'$, two natural types of restrictions are defined: the *left-restriction* $A' \lhd \mathbb{R} \stackrel{\text{def}}{=} \{(a, b) \in \mathbb{R} \mid a \in A'\}$ and the *right-restriction* $\mathbb{R} \rhd B' \stackrel{\text{def}}{=} \{(a, b) \in \mathbb{R} \mid b \in B'\}$. Furthermore, such a binary relation $\mathbb{R}$ canonically induces two mappings $\mathbb{R} : A \to 2^B$, where $\mathbb{R}(a) = \{b \mid (a, b) \in \mathbb{R}\}$ for every $a \in A$ and, analogously $\mathbb{R}^{-1} : B \to 2^A$, where $\mathbb{R}^{-1}(b) = \{a \mid (a, b) \in \mathbb{R}\}$ for every $b \in B$. The former mapping is used without difference in denotation—it will be clear from the context whether we refer to the relation or the induced mapping.

In complexity analyses, $\|o\|$ denotes the encoding size of object $o$ in a standard representation over some suitable, fixed alphabet. To keep formulations simple, we usually say that a problem $P'$ has 'higher' complexity than problem $P$ if $P$ is in some complexity class $\mathcal{C}$ and $P'$ is hard for a complexity class $\mathcal{C}'$ where $\mathcal{C} \subseteq \mathcal{C}'$ holds and $\mathcal{C} \subsetneq \mathcal{C}'$ is commonly *assumed*—even if *not proven*. In this sense, a problem that is $\Pi_2^p$-hard has higher complexity than a problem in $\mathsf{NP}$, for instance.

## 2.1. Relational databases and queries, classically

The following concepts are defined under the assumption of disjoint, countably infinite sets $\mathsf{dom}$ of *data values*—in queries also called *constants*—and $\mathsf{var}$ of *variables*. The

set $\mathsf{var}$ of variables, in particular, is the disjoint union of two countably infinite sets $\mathsf{xvar}$ and $\mathsf{nvar}$ of *data variables* and *node variables*.[1]

To answer algorithmical questions, it is sometimes necessary to restrict to a finite subset of data values where, usually, the exact values do not matter. For simplicity, we assume that $\mathsf{dom}$ contains $\mathbb{N}$ and define the initial fragment $\mathsf{dom}_s \stackrel{\text{def}}{=} \{1, \ldots, s\}$ of size $s$ for every $s \in \mathbb{N}$.

### 2.1.1. Databases

A *schema* is a non-empty, finite set $\mathcal{S}$ of relation symbols, where each relation symbol $R \in \mathcal{S}$ has a fixed arity $\mathsf{ar}(R)$.

An *$R$-atom* for a relation symbol $R$ of arity $m = \mathsf{ar}(R)$ is of the form $R(\tau_1, \ldots, \tau_m)$ where each term $\tau_1, \ldots, \tau_m$ is either a data value or a variable. The set $\{\tau_1, \ldots, \tau_m\}$ of *terms* of such an atom $A = R(\tau_1, \ldots, \tau_m)$ is denoted $\mathsf{term}(A)$ and the corresponding subsets of data variables and data values are defined as $\mathsf{xvar}(A) \stackrel{\text{def}}{=} \mathsf{term}(A) \cap \mathsf{xvar}$ and $\mathsf{dom}(A) \stackrel{\text{def}}{=} \mathsf{term}(A) \cap \mathsf{dom}$, respectively. The latter set is also called the *active domain* of atom $A$. Analogously, the set of node variables is defined as $\mathsf{nvar}(A) \stackrel{\text{def}}{=} \mathsf{term}(A) \cap \mathsf{nvar}$. Atoms over nullary relations are usually denoted without parentheses, that is, we write $R$ instead of $R()$ if $\mathsf{ar}(R) = 0$.

An *$R$-fact* is an $R$-atom whose terms are only data values (not variables) and an $\mathcal{S}$-fact is an $R$-fact for some relation $R \in \mathcal{S}$. An *instance* over a schema $\mathcal{S}$ is a finite set of $\mathcal{S}$-facts and the class of all $\mathcal{S}$-instances whose active domain is a subset of $U$ is denoted $\mathsf{inst}(\mathcal{S}, U)$ or simply $\mathsf{inst}(\mathcal{S})$ if the domain is unrestricted, $U = \mathsf{dom}$. In this context, the set $U$ is often called the *universe*. Similarly, $\mathsf{facts}(\mathcal{S}, U)$ and $\mathsf{facts}(\mathcal{S})$ denote the sets of all facts that may occur in $\mathcal{S}$-instances over universe $U$ or $\mathsf{dom}$, respectively.

A *substitution* for an atom $A = R(\tau_1, \ldots, \tau_m)$ is a partial mapping $s : \mathsf{term} \rightharpoonup \mathsf{term}$ that is defined for all terms in $A$. Its image is $s(A) = R(s(\tau_1), \ldots, s(\tau_m))$. For the sake of brevity, the application of a substitution is sometimes denoted $A[\tau_{i_1}/\tau'_{i_1}, \ldots, \tau_{i_p}/\tau'_{i_p}]$, where only the changes $\tau_{i_j} \neq \tau'_{i_j} = s(\tau_{i_j})$ are specified. Note that a substitution can replace data values (by variables or other data values).

There are two important, successive, refinements of substitutions. First, substitutions that preserve data values (that is, behave like the identity on $\mathsf{dom}$) are called *homomorphisms*. Second, a homomorphism $V$ that maps terms only to data values is called a *valuation*, which is said to *induce fact* $V(A)$ *on atom* $A$. For simplicity and without loss of generality, we assume that homomorphisms and thus also valuations are defined for all data values in $\mathsf{dom}$.

With slight abuse of notation, the canonical generalisations to tuples, to sets and to

[1] Node variables are used in Chapter 5 only. In the previous chapters, the term *variable* always refers to a *data variable*.

similar concepts (like atoms) of the mappings defined here and in the following are used without difference in denotation.

In a few proofs it turns out to be useful to amend facts (or, more generally, atoms) by some information, often by a single data value (or variable). Following this idea, an arbitrary schema $\mathcal{S}$ can be transformed into its *extended schema* $\mathcal{S}'$ that contains the same relation symbols as $\mathcal{S}$ but each with an arity increased by 1. Correspondingly, for a term $\tau$, the *$\tau$-extension* of a set $\mathcal{A} = \{R_1(\boldsymbol{x}_1), \ldots, R_m(\boldsymbol{x}_m)\}$ of atoms is the set $\mathsf{ext}_\tau(\mathcal{A}) \stackrel{\text{def}}{=} \{R_1(\tau, \boldsymbol{x}_1), \ldots, R_m(\tau, \boldsymbol{x}_m)\}$ of atoms over the extended schema, where term $\tau$ is assigned to the additional first position. Each set $\mathcal{A}'$ of atoms over an extended schema is naturally translated back to a set $\mathsf{ext}^{-1}(\mathcal{A}')$ of atoms over the (plain) schema by discarding the term in the first position of each atom. For a set $\mathcal{A}'$ of atoms over an extended schema—that possibly results as a union of multiple extensions—and a single term $\tau$, the *filtered* set $\mathsf{filt}_\tau(\mathcal{A}')$ contains only those atoms with $\tau$ in their first component. Finally, the *$\tau$-reduced* set results by filtering the atoms first and then translating them back, that is, $\mathsf{red}_\tau(\mathcal{A}') \stackrel{\text{def}}{=} (\mathsf{ext}^{-1} \circ \mathsf{filt}_\tau)(\mathcal{A}')$ for every set $\mathcal{A}'$ of atoms over an extended schema. As an example, consider instances $\mathcal{I} = \{R(7), S(8,9)\}$ and a possible union of extensions $\mathcal{I}' = \{R(1,7), S(1,8,9), S(2,7,7)\}$. The former yields an extended instance $\mathsf{ext}_1(\mathcal{I}) = \{R(1,7), S(1,8,9)\}$ and the latter a reduced instance $\mathsf{red}_1(\mathcal{I}') = \{R(7), S(8,9)\}$, respectively. Clearly, for pairwise different terms $\tau_1, \ldots, \tau_m$, equality $\mathsf{red}_{\tau_i}(\mathsf{ext}_{\tau_1}(\mathcal{A}_1) \cup \cdots \cup \mathsf{ext}_{\tau_m}(\mathcal{A}_m)) = \mathcal{A}_i$ holds for every $i \in \{1, \ldots, m\}$ and arbitrary sets $\mathcal{A}_1, \ldots, \mathcal{A}_m$ of atoms.

### 2.1.2. Queries

Queries are mappings from databases over some source schema $\mathcal{S}$ to databases over some target schema $\mathcal{T}$. More precisely, a query $Q$ defines, for every database $D$ over $\mathcal{S}$, a database $Q(D)$, the result of query $Q$ on $D$.

Several formalisms to define queries on relational databases are known. Many of them are variants of logical formulas in first-order predicate logic. In the following, the subtle distinction between a mapping and its representation as a formula or a set of rules is silently ignored in most cases. Both are simply called *query*.

This work has a strong focus on the following query classes that are strict subclasses of first-order predicate logic—syntactically, after an obvious and well-known translation of the rules into formulas, as well as with respect to their expressiveness. The most prominent subclasses among them are *conjunctive queries (CQ)* and *unions of conjunctive queries (UCQ)*, either with or without a restricted form of negation.

### Conjunctive queries (with and without negation)

A *conjunctive query $Q$ with negation ($CQ^\neg$) over schema $\mathcal{S}$* is of the form

$$H \leftarrow A_1, \ldots, A_m, \neg B_1, \ldots, \neg B_n,$$

where the single relation atom $\mathsf{head}(Q) \stackrel{\text{def}}{=} H$ is called the *head of $Q$* and a non-empty, finite set $\mathsf{pos}(Q) \stackrel{\text{def}}{=} \{A_1, \ldots, A_m\}$ of relation atoms over $\mathcal{S}$, is called the *positive body of $Q$*, and a finite set $\mathsf{neg}(Q) \stackrel{\text{def}}{=} \{B_1, \ldots, B_n\}$, is called the *negative body of $Q$*. The query's *body* is the set $\mathsf{body}(Q) \stackrel{\text{def}}{=} \{A_1, \ldots, A_m, \neg B_1, \ldots, \neg B_n\}$ of atoms. The variables in $\mathsf{head}(Q)$ are called *head variables*, all other *existential* or *projection variables*. Each query is required to satisfy the following safety conditions:

- *safe head:* every head variable occurs in at least one positive atom;
- *safe negation:* every variable in a negated atom occurs in a positive atom.

An important subtype of queries is the following: a query $Q$ is a *conjunctive query* (without negation), CQ, if its body does not contain any negated atom—in which case $\mathsf{body}(Q) = \mathsf{pos}(Q)$ holds.

It is common to define restrictions of query classes by the use of variables. A query is *full* (projection-free) if it does not have any existential variables and it is *boolean* if it does not have any head variables.

A *valuation for a conjunctive query $Q$ with negation* is a valuation for $\mathsf{pos}(Q)$, that is, it is a mapping $V : \mathsf{term} \rightharpoonup \mathsf{dom}$ that is defined for every variable of the query, $\mathsf{var}(Q) \stackrel{\text{def}}{=} \mathsf{var}(\mathsf{pos}(Q))$. Valuation $V$ *requires* facts $V(\mathsf{pos}(Q))$ and *prohibits* facts $V(\mathsf{neg}(Q))$ for $\mathrm{CQ}^\neg$ $Q$. It is said to *satisfy $Q$ on instance $\mathcal{I}$* if the instance contains all required facts and no prohibited fact, that is, if $V(\mathsf{pos}(Q)) \subseteq \mathcal{I}$ and $V(\mathsf{neg}(Q)) \cap \mathcal{I} = \emptyset$ holds. In this case, valuation $V$ *derives* fact $V(\mathsf{head}(Q))$. The *query result* on an instance $\mathcal{I}$ is the instance $Q(\mathcal{I})$ that contains exactly those facts that are derived by some valuation that satisfies $Q$ on $\mathcal{I}$.

A query $Q$ is *consistent* if there exists at least one instance $\mathcal{I}$ such that the query result $Q(\mathcal{I})$ is *not* empty. It is well-known that a conjunctive query $Q$ with negation is consistent if and only if there is no direct contradiction in its positive and negated atoms, $\mathsf{pos}(Q) \cap \mathsf{neg}(Q) \neq \emptyset$. In the following, every conjunctive query with negation is silently assumed to be consistent.[2]

### Unions of conjunctive queries (with and without negation)

A *union of conjunctive queries with negation ($UCQ^\neg$) over schema $\mathcal{S}$* consists of finitely many conjunctive queries $Q_1, \ldots, Q_k$ with negation over $\mathcal{S}$, where $k \geq 1$, whose

[2] This does not affect the computational complexity of the studied problems since the syntactical criterion can be checked efficiently.

heads all refer to the same relation symbol. Such a UCQ$^\neg$ is denoted $Q = Q_1 \cup \cdots \cup Q_k$ and queries $Q_1, \ldots, Q_k$ are referred to as the *subqueries* or *disjuncts* of $Q$.

Several properties are generalised to a UCQ$^\neg$ directly from its subqueries. A UCQ$^\neg$ is a *union of conjunctive queries* (without negation) if all its subqueries are conjunctive queries (without negation). Similarly, it is *full* or *boolean* if all its subqueries are *full* or *boolean*, respectively.

A *valuation* for UCQ$^\neg$ $Q = Q_1 \cup \cdots \cup Q_k$ is a valuation for one of its subqueries $Q_i$. Valuation $V$ satisfies $Q$ if it satisfies some $Q_i$, in which case it derives the corresponding fact $V(\mathsf{head}(Q_i))$. The *query result* on instance $\mathcal{I}$ is the union of the results of its subqueries, $Q(\mathcal{I}) \stackrel{\text{def}}{=} Q_1(\mathcal{I}) \cup \cdots \cup Q_k(\mathcal{I})$.

**Remark 2.1.1 (Valuation-subquery correspondance).** Formally, the same valuation can derive multiple facts for different subqueries. On instance $\{R(a,b), S(a)\}$, for example, a single valuation $V = \{x \mapsto a, y \mapsto b\}$ derives three facts $H(a,b)$, $H(b,a)$ and $H(a,a)$ for the query $Q = Q_1 \cup Q_2 \cup Q_3$ with subqueries

$$Q_1 = H(x,y) \leftarrow R(x,y),$$
$$Q_2 = H(y,x) \leftarrow R(x,y) \text{ and}$$
$$Q_3 = H(x,x) \leftarrow S(x).$$

However, this is of no importance in most of the arguments to follow and we hence usually neglect this possible ambiguity, assuming that all subqueries use disjoint sets of variables and that a valuation is defined only for the variables of one subquery. In the case of the query above, we assume an (equivalent) representation like

$$Q_1 = H(u,w) \leftarrow R(u,w),$$
$$Q_2 = H(y,x) \leftarrow R(x,y) \text{ and}$$
$$Q_3 = H(z,z) \leftarrow S(z).$$

Furthermore, a valuation that is defined for $\{u, w\}$ is neither defined for all variables in $\{x, y\}$ nor for all variables in $\{z\}$ (and, analogously, for other sets of variables). Under these assumptions, a valuation $V$ determines a *unique* subquery $Q_i$ of the UCQ $Q$. For simplicity, we then often write $V(\mathsf{head}(Q))$ instead of $V(\mathsf{head}(Q_i))$, eliminating the need for an explicit quantification; not only for $\mathsf{head}(Q)$ but also for $\mathsf{pos}(Q)$, $\mathsf{neg}(Q)$ and so on.

### Queries with disequalities

All previously defined query classes are naturally extended by addition of disequality atoms. A *disequality atom* $\tau \neq \tau'$ relates two terms $\tau, \tau'$ and is *satisfied* by a valuation $V$ if this valuation is defined for both terms and maps them to different

data values. Following our assumption on the consistency of queries, disequalities of the form $a \neq b$ for different constants $a, b$ do *not* occur.

For instance, a *conjunctive query $Q$ with negation and disequalities*, $\mathrm{CQ}^{\neg,\neq}$, is of the form

$$H \leftarrow A_1, \ldots, A_m, \neg B_1, \ldots, \neg B_n, \Delta_1, \ldots, \Delta_p,$$

where $\mathsf{diseq}(Q) = \{\Delta_1, \ldots, \Delta_p\}$ is a finite set of disequality atoms, where each disequality atom is of the form $\tau \neq \tau'$ for terms $\tau, \tau' \in \mathsf{var}(\mathsf{pos}(Q)) \cup \mathsf{dom}$. A valuation $V$ *satisfies* query $Q$ on instance $\mathcal{I}$ if it satisfies each disequality atom and the underlying disequality-free query, $H \leftarrow A_1, \ldots, A_m, \neg B_1, \ldots, \neg B_n$.

The class of all conjunctive queries is denoted $\mathsf{CQ}$. Similarly, $\mathsf{UCQ}$ and $\mathsf{CQ}^{\neq}$ denote the classes of unions of conjunctive queries and conjunctive queries with disequalities, respectively. Further extensions result by combination of unions, disequalities or negations that are easily identified by the class name like $\mathsf{CQ}^{\neg}$, $\mathsf{UCQ}^{\neg,\neq}$ and so on. These rather general (syntactic) properties are sometimes accompanied by further restrictions, which are then specified in brackets. For instance, $\mathsf{CQ}^{\neg}[\mathsf{full}]$ denotes the subclass of queries from $\mathsf{CQ}^{\neg}$ that are full. Analogously, the tags $\mathsf{sm}$ and $\mathsf{pol}$ refer to *strongly minimal* and *polarised* queries, respectively, which we introduce later.

**Remark 2.1.2.** Although queries, as defined above, allow constants in relation and disequality atoms, many complexity results in this work do *not* depend on the use of constants. Therefore, class names $\mathsf{CQ}$, $\mathsf{UCQ}$, ... are reserved for the respective subsets of those queries that are *free of constants* while $\mathsf{CQ}_{\mathrm{dom}}$, $\mathsf{UCQ}_{\mathrm{dom}}$, ... denote those classes where constants may occur.

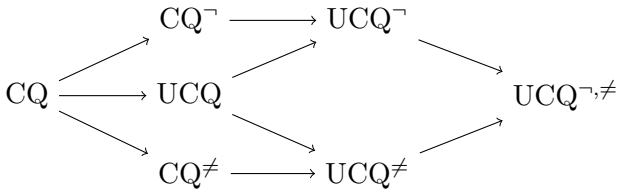


Figure 2.1.: The most important query classes considered in this work and their relationship with respect to syntactical restrictions (arrows towards extensions).

### Properties of queries

Different query representations can describe the same query mapping: query representations $Q$ and $Q'$ are *equivalent*, denoted $Q \equiv Q'$, if they have the same result, $Q(\mathcal{I}) = Q'(\mathcal{I})$, on every instance $\mathcal{I}$. A weaker relationship is query containment:

query $Q$ is *contained* in query $Q'$, denoted $Q \sqsubseteq Q'$, if the result of the latter contains the result of the former, $Q(\mathcal{I}) \subseteq Q'(\mathcal{I})$, for every instance $\mathcal{I}$. Naturally, two queries are equivalent if and only if they mutually contain each other.

If a query $Q_1$ is *not* contained in another query $Q_2$, then, by definition, there is a finite instance $\mathcal{I}$ witnessing that. More precisely, the size of a minimal such instance is bounded by the size of the queries in the following way.

**Fact 2.1.3 (Counterexamples for containment).** Let $Q_1$ and $Q_2$ be queries in $\mathsf{UCQ}^{\neg,\neq}$. If $Q_1 \not\sqsubseteq Q_2$, then there is an instance $\mathcal{I}$ of size at most $rv^\alpha$ such that $Q_1(\mathcal{I}) \not\subseteq Q_2(\mathcal{I})$, where $v$ is the number of variables in $Q_1$ and $r$ and $\alpha$ are the number and the maximal arity of relation symbols in both queries $Q_1$ and $Q_2$, respectively. Indeed, if query $Q_2$ is in $\mathsf{UCQ}^{\neq}$, the size bound depends on $Q_1$ alone.

Note in particular that the size, $rv^\alpha$, is polynomial in the size of the queries if $\alpha$ is viewed as constant. Moreover, for queries $Q$ and $Q'$ from $\mathsf{CQ}$, containment can be characterised by the existence of homomorphisms. A *homomorphism from $Q$ to $Q'$* is a homomorphism $h$ where $h(\mathsf{head}(Q)) = \mathsf{head}(Q')$ and $h(\mathsf{pos}(Q)) \subseteq \mathsf{pos}(Q')$. Homomorphisms from a query to itself are called *endomorphisms*.

For some queries, the result set can never diminish if facts are added to the instance: a query $Q$ is *monotonic* if $Q(\mathcal{I}) \subseteq Q(\mathcal{I}')$ for all instances $\mathcal{I}, \mathcal{I}'$ where $\mathcal{I} \subseteq \mathcal{I}'$. The following fact is well-known.

**Fact 2.1.4 (Monotonicity of $\mathsf{UCQ}^{\neq}$).** Let $\mathcal{I}, \mathcal{I}'$ be instances where $\mathcal{I} \subseteq \mathcal{I}'$ and let $Q$ be some $\mathrm{UCQ}^{\neq}$. If a valuation $V$ satisfies $Q$ on $\mathcal{I}$, then $V$ satisfies $Q$ also on $\mathcal{I}'$. In particular, this implies that query $Q$ is monotonic.

If at all, the query result usually depends on the concrete data values in the instance only in a restricted way. For a set $D \subseteq \mathsf{dom}$ of data values, a query is *$D$-generic* if $\pi(Q(\mathcal{I})) = Q(\pi(\mathcal{I}))$ holds for every permutation $\pi : \mathsf{dom} \to \mathsf{dom}$ that behaves like the identity on $D$. In the special case where $D = \emptyset$, the query is simply called *generic*. Every query $Q \in \mathsf{UCQ}^{\neg,\neq}_{\mathrm{dom}}$ is $\mathsf{dom}(Q)$-generic. In particular, each constant-free query is generic.

## 2.2. Relational databases and queries, distributed

The simplicity of the MPC model, where there is no hierarchy on the servers and no restriction on the communication between them (as in other network topologies), justifies the following definition. A *network* is a non-empty finite set, whose elements are called *nodes*. The *size* of a network $N$ is its cardinality $|N|$.

### 2.2.1. Databases

Basically, distributed databases are collections of instances that are stored on the nodes of a network. Since we are particularly interested in the comparison of parallel and centralised (classical) evaluation, we follow the 'local-as-view' approach and additionally keep track of *all* facts via a 'global' instance.

**Definition 2.2.1 (Distribution and distributed database).** Given some network $N$ and a schema $\mathcal{S}$, a *distribution* of an instance $\mathcal{I}$ of $\mathcal{S}$-facts over $N$ is a relation $\mathbb{L} \subseteq N \times \mathcal{I}$ that associates every node $k \in N$ with the *local instance* $\mathbb{L}(k)$ *of node* $k$. A *distributed database over* $N$ is a generalisation of this concept; it is a pair $\mathbb{D} = (\mathcal{G}, \mathbb{L})$ consisting of an $\mathcal{S}$-instance $\mathcal{G}$, the *global instance of* $\mathbb{D}$, and a *distribution* $\mathbb{L}$ of $\mathcal{G}$ over $N$. ◀

The formalisation above explicitly allows replication of facts across nodes. Furthermore, it does *not* require that all facts of the global instance $\mathcal{G}$ are actually present at some node of the network.

The sets of those nodes and facts that *are* in the domain and range of relation $\mathbb{L}$ are denoted $\mathsf{net}(\mathbb{L})$ and $\mathsf{facts}(\mathbb{L})$, respectively. A distributed database $\mathbb{D} = (\mathcal{G}, \mathbb{L})$ is *complete* if $\mathcal{G} \subseteq \mathsf{facts}(\mathbb{L})$, in which case the two sets are equal since $\mathsf{facts}(\mathbb{L}) \subseteq \mathcal{G}$ is guaranteed by Definition 2.2.1. In this case, the global database $\mathcal{G}$ is implicitly defined by $\mathbb{L}$ and often omitted. For incomplete distributed databases, the facts that are in $\mathcal{G} - \mathsf{facts}(\mathbb{L})$ are said to be *skipped* by $\mathbb{D}$.

### 2.2.2. Query evaluation

The result of a distributed, communication-free evaluation of a query is the union of all partial results obtained at the individual nodes of the network.

**Definition 2.2.2 (Distributed evaluation).** Let $\mathbb{D} = (\mathcal{G}, \mathbb{L})$ be a distributed database over some schema $\mathcal{S}$ and some network $N$. Furthermore, let $Q$ be a query from schema $\mathcal{S}$ to schema $\mathcal{T}$.

The *distributed evaluation* of $Q$ on $\mathbb{D}$ yields a new distribution $Q(\mathbb{L}) \subseteq N \times \mathsf{inst}(\mathcal{T})$ by pointwise evaluation of the query on the local instances, that is,

$$Q(\mathbb{L}) \stackrel{\text{def}}{=} \bigcup_{k \in N} \{k\} \times Q(\mathbb{L}(k)).$$

We are particularly interested in those facts that are located (or derived) on *some* node of a distribution $\mathbb{L}$, forgetting about where it is derived. In these cases, the set $\mathsf{facts}(\mathbb{L})$ is denoted $\bigsqcup \mathbb{L}$, emphasising that it results as the union of the local instances. In particular, the *distributed query result* $\bigsqcup Q(\mathbb{L})$ *of* $Q$ *on* $\mathbb{D}$ is the union $\mathsf{facts}(Q(\mathbb{L}))$ of all partial results. ◀

To facilitate the description, the mapping $\mathbb{L}^{-1}$ induced by a distribution $\mathbb{L}$ is generalised from single facts to sets of facts in the following way without notational difference: for every set $\mathcal{F}$ of facts, $\mathbb{L}^{-1}(\mathcal{F}) \stackrel{\text{def}}{=} \bigcap_{f \in \mathcal{F}} \mathbb{L}^{-1}(f)$ is the set of those nodes whose local instances contain *all* facts from $\mathcal{F}$.

Note that the evaluation strategy in Definition 2.2.2 is rather naïve. It does not take into account any knowledge about the relationship between the global instance and the local instances. In particular, when a fact $f$ is missing in a local instance, it is unclear whether it is also missing in the global instance or only skipped.

### 2.2.3. Distribution policies

One of our goals is to explore the scenarios where the (naïve) distributed evaluation yields the same result as the centralised evaluation. Typically, distributions are not determined arbitrarily but rather following some pattern. In our setting, this regularity is captured by the notion of a distribution *policy*. We start with the definition and an example of policies and continue with a discussion of different representations for them.

#### Definition

Policies determine which nodes are responsible for which facts.

**Definition* 2.2.3.** A *(distribution) policy over a network $N$ and a schema $\mathcal{S}$* is a finite distribution $\mathbb{P} \subseteq N \times \mathsf{facts}(\mathcal{S})$ of arbitrary facts over $\mathcal{S}$. ◀

The formalisation of distribution policies is a weak generalisation of distributions (where the set of facts is not only restricted by the schema but also by facts in the given instance). Therefore, each distribution can be seen as a distribution policy and vice versa. However, both concepts are used in a different sense: while a distribution is meant to represent an *actual* placement of facts on the nodes of a network, the policy is meant to represent the *intended* placement of facts. Given a policy $\mathbb{P}$, a node $k$ is said to be *responsible* for a fact $f$ if $f \in \mathbb{P}(k)$.

In this sense, for every fixed instance $\mathcal{G}$, a policy $\mathbb{P}$ induces a distribution $\mathbb{L} = \mathbb{P} \triangleright \mathcal{G}$, as illustrated in Figure 2.2 (on page 18). We note that the definition implies equality

$$\mathbb{L}(k) = (\mathbb{P} \triangleright \mathcal{G})(k) = \mathbb{P}(k) \cap \mathcal{G}$$

for each node $k$. That is, the local instance comprises exactly those facts for which the node is responsible and which are present in the global instance. Notably, this definition implies that the actual placement of a fact is independent of the occurrence of other facts in instance $\mathcal{G}$: for every node $k$ and every fact $f$ it holds $f \in \mathbb{L}(k)$ if and only if $f \in \mathcal{G}$ and $f \in \mathbb{P}(k)$.

**Remark 2.2.4.** Notably, the finiteness of the relation particularly implies that the set of data values occurring in the facts referred to by $\mathbb{P}$ is finite too. This set is called the *universe of* $\mathbb{P}$, denoted $\mathsf{univ}(\mathbb{P})$.

The set of all relation symbols actually referenced by a policy $\mathbb{P}$ is denoted $\mathsf{sch}(\mathbb{P})$. Together with the policy's universe it induces the set $\mathsf{inst}(\mathbb{P}) \stackrel{\text{def}}{=} \mathsf{inst}(\mathsf{sch}(\mathbb{P}), \mathsf{univ}(\mathbb{P}))$ of instances. We say that an instance $\mathcal{I}$ is an *instance over* $\mathbb{P}$ if $\mathcal{I} \in \mathsf{inst}(\mathbb{P})$ and, similarly, that a valuation $V$ is a *valuation over* $\mathbb{P}$ if it maps only to values in $\mathsf{univ}(\mathbb{P})$.

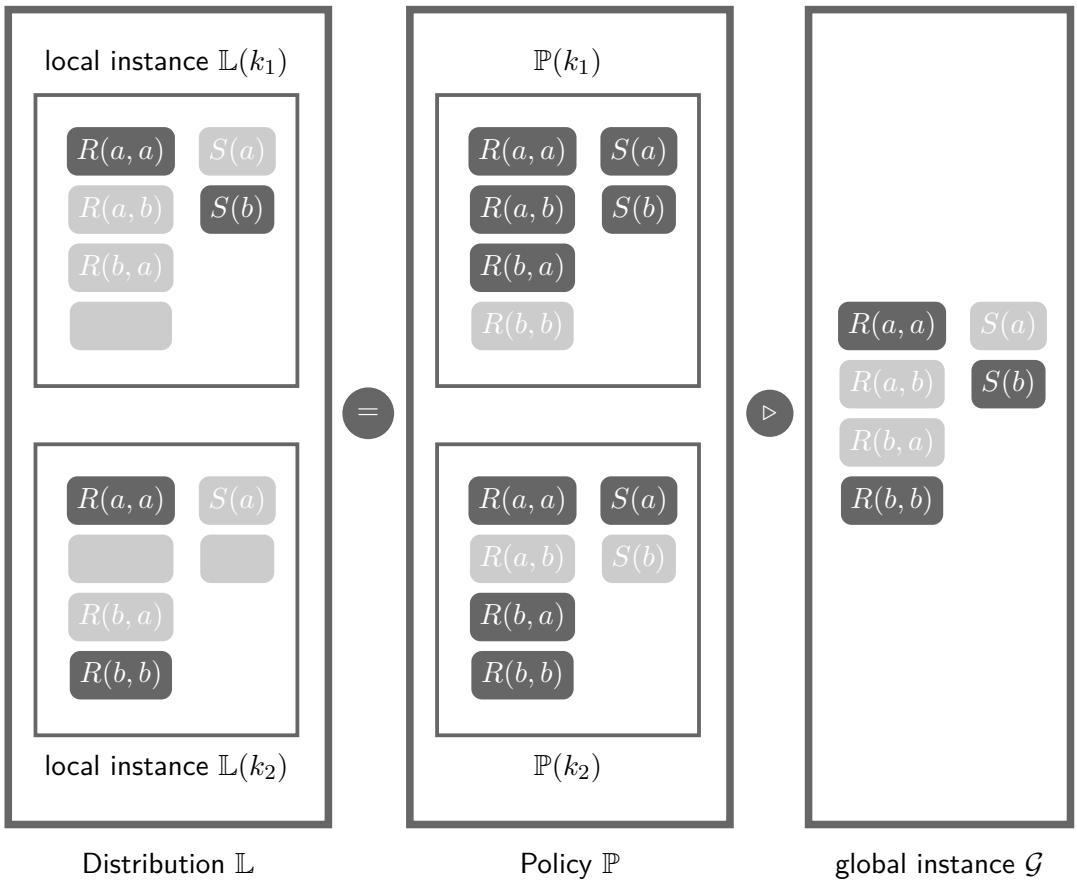


Figure 2.2.: Example distribution $\mathbb{L}$ with two local instances $\mathbb{L}(k_1) = \{R(a,a), S(b)\}$ and $\mathbb{L}(k_2) = \{R(a,a), R(b,b)\}$. This distribution results as the right restriction of policy $\mathbb{P}$ to the global instance $\mathcal{G} = \{R(a,a), R(b,b), S(b)\}$, where $\mathbb{P}$ is the union of $\{k_1\} \times \{R(a,a), R(a,b), R(b,a), S(a), S(b)\}$ and $\{k_2\} \times \{R(a,a), R(b,a), R(b,b), S(a)\}$.

### Algorithmical representation

Since some of the main algorithmical problems studied later expect a policy as input, an obvious question is how such a policy is represented. For all representations, we assume that nodes and facts are encoded as words over some fixed alphabet.

The most simple type considered in this thesis is the explicit representation of the relation as a list of pairs. The class $\mathcal{P}_{\text{list}}$ contains exactly those representations. Besides this type, we consider two other kinds of policy representations: a representation based on rules and one based on algorithms.

The second—slightly more implicit—representation is given by a finite set of rules of the form $A \to k$ for an atom $A$ and a node $k$. If policy $\mathbb{P}$ is represented by the set $\{A_1 \to k_1, \ldots, A_r \to k_r\}$ of rules, then its network is $\mathsf{net}(\mathbb{P}) = \{k_1, \ldots, k_r\}$ and its universe is the set of constants explicitly present in its defining atoms, $\mathsf{univ}(\mathbb{P}) = \mathsf{dom}(A_1) \cup \cdots \cup \mathsf{dom}(A_r)$. Under a rule-based policy, a node $k \in \mathsf{net}(\mathbb{P})$ is considered responsible for a fact $f$ over $\mathsf{univ}(\mathbb{P})$ if there is a rule $A_i \to k$ and a valuation $V$ for $A_i$ such that $V(A_i) = f$ holds.

As an example, rules $\{R(0,1) \to k, R(1,0) \to k, R(x,x) \to \ell\}$ represent a two-node policy with universe $\{0,1\}$ that distributes facts $R(0,1)$ and $R(1,0)$ to node $k$ while it distributes facts $R(0,0)$ and $R(1,1)$ to node $\ell$.

The class $\mathcal{P}_{\text{rule}}$ consists of all such rule-based representations. In particular, it contains the class $\mathcal{P}_{\text{list}}$, whose policies can be interpreted as lists of variable-free rules. Contrary to list-based policies, for rule-based policies the number of facts can grow exponentially in the representation size (depending on the arity of the relation symbols). This is relevant for a few of the hardness results to follow.

The third—much more implicit—way to define a policy $\mathbb{P}$ is by means of an algorithm that decides relation $\mathbb{P}$. Although, in principle, arbitrary algorithms could be considered, a restriction to 'efficient' algorithms seems desirable from a practical point of view. In this work, the measure of efficiency is the worst-case run time of the algorithm.

For every constant $c \in \mathbb{N} - \{1\}$, the class $\mathcal{P}^c_{\text{alg}}$ contains algorithmic representations of policies of the form $(\mathbb{A}_{\mathbb{P}}, \beta)$ , where $\mathbb{A}_{\mathbb{P}}$ is an algorithm that decides nondeterministically in time $\mathcal{O}(\|(k,f)\|^c)$ whether $(k,f) \in \mathbb{P}$ holds. For technical reasons, the set of inputs for algorithm $\mathbb{A}_{\mathbb{P}}$ is restricted by the unary-coded bound $\beta$ such that every node and every fact can be represented by at most $\beta$ symbols over the assumed (fixed) alphabet. In particular, each data value is encoded by at most $\beta$ symbols and the universe is the set of all values with such encodings. Note that the size of the network can be exponential in $\beta$. Nevertheless, due to the unary encoding of $\beta$, each node has an identifier of polynomial length, which hence can be 'guessed' by an NP-algorithm that gets representation $(\mathbb{A}_{\mathbb{P}}, \beta)$ as an input.

Restriction to *deterministic* algorithms yields the class $\mathcal{P}^c_{\text{d-alg}}$ for every $c \in \mathbb{N} - \{1\}$. Here, bound $\beta$ is not seen as a bound on the encoding size of nodes but as a bound on

the *number* of nodes, say, the first $\beta$ nodes according to a fixed lexicographical order on the alphabet. This yields a linear bound on the size of the network and allows to compute all local instances induced by a given global instance deterministically in polynomial time.

In the following, the formal distinction between a policy and its representation is often ignored and only made explicit when necessary. Ignoring this distinction, class $\mathcal{P}_{\text{rule}}$ is evidently a subclass of $\mathcal{P}^c_{\text{d-alg}}$ and thus also of $\mathcal{P}^c_{\text{alg}}$, for a suitable $c$, since every explicit representation can be translated into a naïve look-up algorithm (with an appropriate bound $\beta$).

**Remark 2.2.5 (Choice of algorithms).** The consideration of even *nondeterministic* algorithms is probably not of practical relevance. It is, however, theoretically advisable since it is formally less restrictive while often *not* increasing the worst-case complexity bounds discussed in this work. Furthermore, classes $\mathcal{P}^1_{\text{alg}}$ and $\mathcal{P}^1_{\text{d-alg}}$ are excluded because 'linear time' is a subtle notion.

For the upper bounds proven in the following chapters, it is often unimportant which of the representations is used. Similarly, the lower bounds often hold already for the most simple (list-based or rule-based) representations. For ease of reference, we thus define the following two families of policy classes.

**Definition 2.2.6 (Natural policy classes).** The family of *deterministic natural policy classes* is $\mathfrak{P}_{\text{det}} \stackrel{\text{def}}{=} \{\mathcal{P}_{\text{list}}, \mathcal{P}_{\text{rule}}\} \cup \{\mathcal{P}^c_{\text{d-alg}} \mid c \geq 2\}$, a restriction of the family of *natural policy classes*, $\mathfrak{P} \stackrel{\text{def}}{=} \mathfrak{P}_{\text{det}} \cup \{\mathcal{P}^c_{\text{alg}} \mid c \geq 2\}$. ◀

# 3. Parallel correctness

Typically, database management systems have to answer several queries in a short period of time—perhaps repeatedly, over changing data. The naïve approach for distributed systems is to redistribute the data for the evaluation of the query at hand, regardless of the current distribution (resulting from previous computations). Since communication over the network causes a significant part of the total computation time, it seems desirable to minimise the amount of redistributed data or—ideally—to avoid it completely.

This demands a concise representation of the distribution or, at least, of the distribution *pattern* and the possibility to reason about its usefulness for the query. *Parallel correctness* formalises the ideal scenario (of avoiding redistribution completely) for policies as distribution patterns: given a query $Q$ and a policy $\mathbb{P}$, is the result on instance $\mathcal{G}$ equal to the result of the distributed evaluation on the local instances $\mathbb{L}(1), \ldots, \mathbb{L}(n)$ defined by $\mathcal{G}$ and $\mathbb{P}$? Is that the case for *the one* instance $\mathcal{G}$ at hand, in which case equation

$$Q(\mathcal{G}) = Q(\mathbb{L}(1)) \cup \ldots \cup Q(\mathbb{L}(n))$$

holds, or does it even hold for *every* suitable instance? Moreover, equality can be seen as the convergence of an underapproximation and an overapproximation of the query result by the distributed computation, resulting in problems related to parallel correctness—*parallel soundness* and *parallel completeness*.

**Structure of this chapter.** After providing a precise definition of the notion of parallel correctness and some variants in Section 3.1, we study the complexity of the corresponding decision problems for several query classes. In Section 3.2, we begin with conjunctive queries or unions thereof, possibly with disequalities, and extend our study by allowing negation of relation atoms in the subsequent Section 3.3. Finally, we provide an overview of related results in Section 3.4.

For queries in $\mathsf{UCQ}^{\neq}$, which are monotonic, parallel correctness is equivalent to parallel completeness. Parallel completeness of a query, in turn, can be characterised by *minimally requiring* valuations (Theorem* 3.2.5) and we prove it to be $\Pi_2^p$-complete (Theorem 3.2.7) in general, while it is coNP-complete for *strongly minimal* queries—which only have minimal valuations (Proposition* 3.2.17).

For queries in $\mathsf{UCQ}^{\neg,\neq}$, which are not necessarily monotonic, parallel soundness is not trivial but studied separately. Although all variants remain $\Pi_2^p$-complete if the maximum arity of the relations is bounded in advance, the complexity raises considerably without this restriction: parallel correctness—as soundness and completeness—is coNEXPTIME-complete in general (Theorem 3.3.4). However, we identify fragments that exhibit a lower complexity.

For *full* queries—without projection—, the problems are in coNP (Proposition* 3.3.9) and, indeed, coNP-hard (Proposition 3.3.7). As a side-product, we get that lower bounds for the containment problem are automatically lower bounds for the parallel correctness problem for certain query classes like the relational algebra or Datalog (Corollary* 3.3.8).

For *polarised* queries—where each relation occurs only positively or only negated—, the notion of minimally requiring valuations can be extended to incorporate prohibited facts, induced by negated atoms. Interestingly, the canonical extension to *minimal* valuations allows to characterise parallel soundness (Proposition 3.3.15) but *not* to characterise parallel completeness. The latter is possible though by extending the notion of minimal valuations to $\mathbb{P}$-*minimal* valuations (Proposition 3.3.21), which differentiates between 'effectively' and 'ineffectively prohibited' facts (with respect to policy $\mathbb{P}$). The corresponding decision problems are $\Pi_2^p$-complete, irrespective of the arities of the underlying schema (Theorem 3.3.22).

## 3.1. Definition

The most restricted form of 'parallel correctness' considers the evaluation of a query on a *specific* distributed database, as defined and illustrated next. We define consecutive generalisations of this notion further below.

**Definition* 3.1.1 (Parallel correctness on distributed database).** Let $Q$ be a query and $\mathbb{D} = (\mathcal{G}, \mathbb{L})$ be a distributed database. Query $Q$ is *parallel-correct on* $\mathbb{D}$ if

1. $Q$ is *parallel-sound on* $\mathbb{D}$, that is, if $\bigsqcup Q(\mathbb{L}) \subseteq Q(\mathcal{G})$; and
2. $Q$ is *parallel-complete on* $\mathbb{D}$, that is, if $\bigsqcup Q(\mathbb{L}) \supseteq Q(\mathcal{G})$. ◀

An important special case of the previous definition arises when the distributed database results by a policy. In this case, a policy $\mathbb{P}$ and a global instance $\mathcal{G}$ are given and induce a distributed database $\mathbb{D} = (\mathcal{G}, \mathbb{L})$, where $\mathbb{L} = \mathbb{P} \triangleright \mathcal{G}$. Correspondingly, we say that query $Q$ *is parallel-correct on* $\mathcal{G}$ *under* $\mathbb{P}$ if the conditions above are satisfied (and similarly for the constituent properties, parallel soundness and parallel completeness).

**Example 3.1.2.** Let $\mathbb{D} = (\mathcal{G}, \mathbb{L})$ be the distributed database that is induced by the global instance $\mathcal{G} = \{R(a,a), R(b,b), S(b)\}$ and the two-node policy

$$\mathbb{P} \stackrel{\text{def}}{=} \{k_1\} \times \{R(a,a), R(a,b), R(b,a), S(a), S(b)\} \\ \cup \{k_2\} \times \{R(a,a), R(b,a), R(b,b), S(a)\}$$

from Figure 2.2 (on page 18). For query $Q : H(x) \leftarrow R(x,x), \neg S(x)$, this yields the distributed query result

$$\bigsqcup Q(\mathbb{L}) = \{H(a), H(b)\} \supsetneq \{H(a)\} = Q(\mathcal{G}),$$

as illustrated in Figure 3.1 (on page 24).

On the one hand, query $Q$ is parallel-complete on $\mathcal{G}$ under $\mathbb{P}$ because the distributed result set contains all facts from the result set of the (classical) centralised evaluation. On the other hand, query $Q$ is *not* parallel-sound on $\mathcal{G}$ under Policy $\mathbb{P}$ because additional facts are derived locally, namely $H(b)$. The latter is a consequence of the unawareness of node $k_2$ of the prohibited fact $S(b)$, due to the definition of policy $\mathbb{P}$. Clearly, query $Q$ is not parallel-correct on $\mathcal{G}$ under $\mathbb{P}$, since it is not parallel-sound. ■

The properties introduced in Definition 3.1.1 refer to the behaviour of the given query on a *single* global instance. These properties are naturally extended to *all* global instances, or—strictly speaking—to *all instances induced by the policy*.[1]

**Definition* 3.1.3 (Parallel correctness under policy).** Let $Q$ be a query and let $\mathbb{P}$ be a policy. Query $Q$ is *parallel-correct under* $\mathbb{P}$ if

1. $Q$ is *parallel-sound under* $\mathbb{P}$, that is,
   if $Q$ is parallel-sound on $\mathcal{G}$ under $\mathbb{P}$ for every $\mathcal{G} \in \mathsf{inst}(\mathbb{P})$; and
2. $Q$ is *parallel-complete under* $\mathbb{P}$, that is,
   if $Q$ is parallel-complete on $\mathcal{G}$ under $\mathbb{P}$ for every $\mathcal{G} \in \mathsf{inst}(\mathbb{P})$. ◀

The previous definition gives rise to the following static analysis problem.

**PC($\mathcal{Q}$,$\mathcal{P}$)**

| | |
|---|---|
| **Parameters:** | query class $\mathcal{Q}$ and policy class $\mathcal{P}$ |
| **Input:** | query $Q \in \mathcal{Q}$,<br>policy $\mathbb{P} \in \mathcal{P}$ |
| **Question:** | Is $Q$ parallel correct under $\mathbb{P}$? |

Following Definition 3.1.1, we also consider a dynamic variant of this problem, where the correct evaluation of the query is to be determined only for a *single* distribution, induced by a given global instance.

[1] Instances with facts over additional relations or facts with additional data values either have no effect or a trivialising effect on parallel correctness.

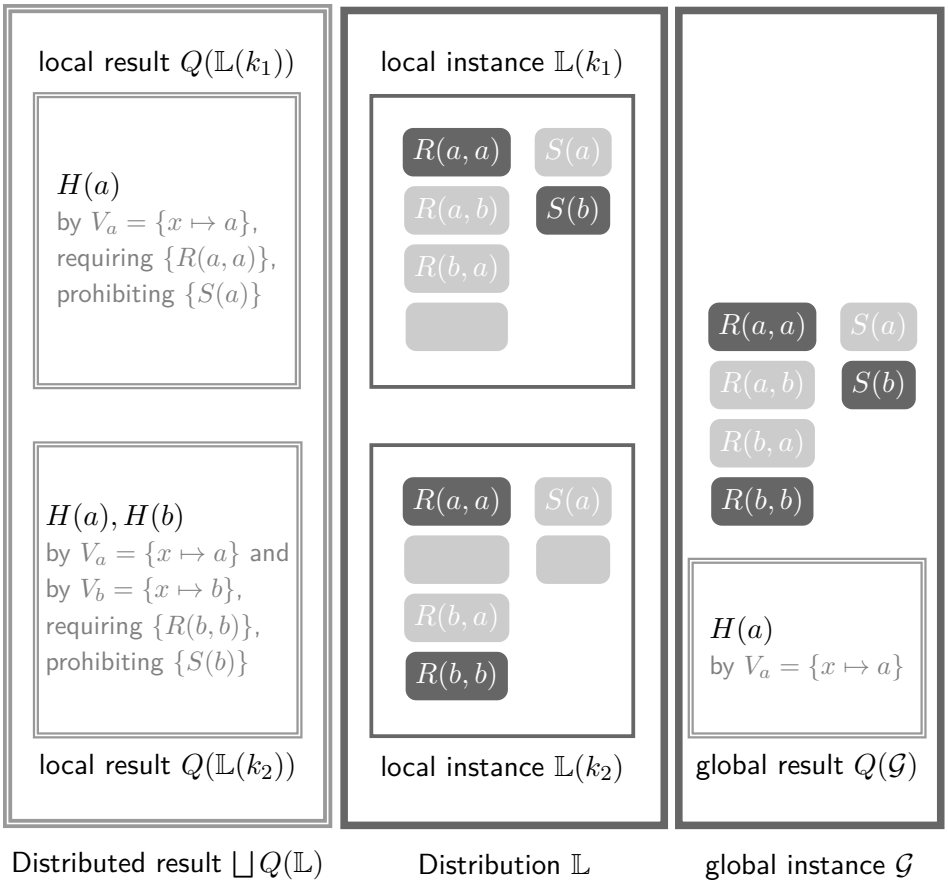


Figure 3.1.: Exemplary evaluation of query $Q : H(x) \leftarrow R(x,x), \neg S(x)$ with distributed and global result sets (left and right). Query $Q$ is *parallel-complete* on $\mathbb{D} = (\mathcal{G}, \mathbb{L})$ because the only fact $H(a)$ derived globally is also derived locally—even twice. Query $Q$ is however not *parallel-sound* on $\mathbb{D}$ because fact $H(b)$ is derived locally but not globally.

**$\mathrm{PC_i}(\mathcal{Q},\mathcal{P})$**

| | |
|---|---|
| **Parameters:** | query class $\mathcal{Q}$ and policy class $\mathcal{P}$ |
| **Input:** | query $Q \in \mathcal{Q}$,<br>policy $\mathbb{P} \in \mathcal{P}$,<br>instance $\mathcal{G}$ |
| **Question:** | Is $Q$ parallel-correct on $\mathcal{G}$ under $\mathbb{P}$? |

For both problems, static and dynamic, variants for parallel soundness and parallel completeness are defined analogously. They are denoted $\text{PSound}_{\text{i}}(\mathcal{Q},\mathcal{P})$ and $\text{PSound}(\mathcal{Q},\mathcal{P})$ or $\text{PComp}_{\text{i}}(\mathcal{Q},\mathcal{P})$ and $\text{PComp}(\mathcal{Q},\mathcal{P})$, respectively.

In the next section, we start the investigation of the complexity of parallel correctness for conjunctive queries and unions thereof. Afterwards, in the subsequent section, we extend our study to queries that also allow atomic negation.

## 3.2. Unions of conjunctive queries

Let us start with a simple but central observation for the negation-free class of unions of conjunctive queries: *each query $Q \in \mathsf{UCQ}^{\neq}_{dom}$ is trivially parallel-sound because it is monotonic.* Monotonicity implies that each local result set $Q((\mathbb{P} \triangleright \mathcal{G})(k))$ is a subset of the global result set $Q(\mathcal{G})$ because the local instance $(\mathbb{P} \triangleright \mathcal{G})(k)$ is a subset of the global instance $\mathcal{G}$. More precisely, Fact 2.1.4 guarantees that, whenever a fact is locally derivable, then it is also globally derivable—even by the same valuation.

Thus, for the queries considered in this section, *the parallel correctness problem is equivalent to the parallel completeness problem.* For the sake of clarity, we refer to the problem by the latter name.

Our study begins with the general case (arbitrary unions of conjunctive queries with disequalities and constants) in Section 3.2.1, where parallel completeness is $\Pi^p_2$-complete. It continues with the fragment of strongly minimal queries in Section 3.2.2, where the complexity decreases to coNP-completeness for queries *without* disequalities, albeit it remains $\Pi^p_2$-hard for queries *with* disequalities.

### 3.2.1. General case

Before we investigate the worst-case complexity of the problems PComp and $\text{PComp}_{\text{i}}$, we consider some notions that allow to characterise parallel completeness in terms of valuations instead of instances. These notions provide valuable insights into the problem on the one hand and facilitate the complexity analysis on the other hand. Moreover, variants of these notions are useful also for some other query classes (like polarised queries, considered in Section 3.3.3) and related problems (like parallel-completeness transfer, considered in Chapter 4).

#### Characterisation

Parallel completeness is defined with respect to a set of global instances, in the static variant, or a single global instance, in the dynamic variant. Usually, these instances are rather large—bounded by the size of the policy's universe as well as the number and arities of its relation symbols, and thus exponential in the worst case.

However, distribution policies are highly regular: they are defined on the granularity of a fact. Because of this regularity, parallel completeness can be characterised with respect to a set of 'very small' instances. These instances are the facts required by *minimal* valuations. The notion of minimality, in turn, is based on the preorder defined next.

Here and in the following, we rely on the valuation-subquery correspondance mentioned in Remark 2.1.1 to facilitate notation for *unions* of conjunctive queries.

**Definition* 3.2.1 (Preorder $\leq_Q^{\text{pos}}$).** For a query $Q \in \mathsf{UCQ}^{\neg,\neq}$, the preorder $\leq_Q^{\text{pos}}$ relates valuations for $Q$. If $V_1, V_2$ are valuations for $Q$, then $V_1 \leq_Q^{\text{pos}} V_2$ holds if

1. both valuations derive the same fact, $V_1(\mathsf{head}(Q)) = V_2(\mathsf{head}(Q))$, and
2. valuation $V_1$ requires only facts that $V_2$ requires, $V_1(\mathsf{pos}(Q)) \subseteq V_2(\mathsf{pos}(Q))$. ◀

Starting from preorder $\leq_Q^{\text{pos}}$, an equivalence relation $\equiv_Q^{\text{pos}}$ and a strict preorder $<_Q^{\text{pos}}$ are canonically defined such that

- $V_1 \equiv_Q^{\text{pos}} V_2$ holds if $V_1 \leq_Q^{\text{pos}} V_2$ and $V_2 \leq_Q^{\text{pos}} V_1$ hold and
- $V_1 <_Q^{\text{pos}} V_2$ holds if $V_1 \leq_Q^{\text{pos}} V_2$ and $V_1(\mathsf{pos}(Q)) \subsetneq V_2(\mathsf{pos}(Q))$ hold.

If it is clear from the context, the query-indicating index is sometimes omitted.

**Definition* 3.2.2 (Minimally requiring valuation).** For a fixed query $Q \in \mathsf{UCQ}^{\neg,\neq}$, a valuation $V$ is *minimally requiring* if it is minimal with respect to $\leq_Q^{\text{pos}}$, that is, if there is no valuation $U$ for $Q$ such that $U <_Q^{\text{pos}} V$. ◀

It is worth noting that for *unions of* queries, the notion of minimality of valuations can cross the boundaries of the constituent subqueries, as illustrated in the following example.

**Example 3.2.3.** Let $Q$ be the UCQ consisting of two conjunctive queries,

- $Q_1 = H(x, x) \leftarrow R(x, x)$ and
- $Q_2 = H(y, z) \leftarrow R(y, z), S(y, z)$.

Although valuation $V_2 = \{y \mapsto 0, z \mapsto 0\}$ is minimal for subquery $Q_2$—since it requires only one fact for each relation symbol in its body—it is not minimal for query $Q$. This is witnessed by another valuation, $V_1 = \{x \mapsto 0\}$, for which $V_1 <_Q V_2$ holds because $V_1$ derives the same fact, $V_1(\mathsf{head}(Q)) = H(1, 1) = V_2(\mathsf{head}(Q))$, but requires fewer facts, $V_1(\mathsf{pos}(Q_1)) = \{R(0, 0)\} \subsetneq \{R(0, 0), S(0, 0)\} = V_2(\mathsf{pos}(Q_2))$. ■

**Remark 3.2.4.** For negation-free queries in general and in this section in particular, we simplify notation a bit. For a query $Q \in \mathsf{UCQ}^{\neq}_{\text{dom}}$, we also write $\leq_Q$ instead of $\leq_Q^{\text{pos}}$. Furthermore, we mostly write just *minimal* instead of *minimally requiring*.

This is justified later by generalisations of the corresponding definitions (Section 3.3.3), where negated atoms are also taken into account.

For a fixed query $Q$ and an arbitrary valuation $V$ for $Q$, there is always a minimal valuation $U$ such that $U \leq V$. This minimal valuation is, however, not necessarily unique, as the following example illustrates. Consider the boolean query $H() \leftarrow R(x), R(y)$ and valuation $V \stackrel{\text{def}}{=} \{x \mapsto 1, y \mapsto 2\}$. Valuation $V$ is not minimal for the query because it requires two facts and thus strictly more than the following valuations. Valuations $U_1 \stackrel{\text{def}}{=} \{x \mapsto 1, y \mapsto 1\}$ and $U_2 \stackrel{\text{def}}{=} \{x \mapsto 2, y \mapsto 2\}$ are both minimal for the query because each valuation requires only a single fact. In particular, $U_1 < V$ and $U_2 < V$ hold since both valuations derive the same fact as $V$. Note that $U_1$ and $U_2$ are not only distinct but also require distinct sets of facts.

Determining whether $V_1 \leq_Q V_2$ holds for given valuations $V_1, V_2$ is possible in polynomial time, while determining whether a valuation is minimal for a given query is coNP-complete [AGK+17a].

Now, we have the concepts to characterise parallel completeness by a set of 'very small' instances. These instances are the facts required for minimal valuations.

Intuitively, it is sufficient to witness the derivation of each fact in a query's result set by a *minimal* valuation. Since such a valuation requires a minimal number of facts, it is always satisfied on some node's local instance if a non-minimal valuation deriving the same fact is satisfied. This idea motivates the following condition.

**Condition (PComp)**

**Assumptions: Let $Q$ be a $\mathsf{UCQ}^{\neq}$ and let $\mathbb{P}$ be a policy.**
For every minimal valuation for $Q$ over $\mathbb{P}$,
there is a node in $\mathsf{net}(\mathbb{P})$ that is responsible for all facts required by the valuation.

The next theorem states that this condition characterises parallel completeness for unions of conjunctive queries without negation.

**Theorem* 3.2.5 (Characterisation of parallel completeness).** For every query $Q \in \mathsf{UCQ}^{\neq}$ and every policy $\mathbb{P}$, query $Q$ is parallel-complete under $\mathbb{P}$ if and only if Condition (PComp) is satisfied for $Q$ and $\mathbb{P}$.

Extending the condition from *minimal* valuations to *all* valuations, as in Condition (PComp-naïve) below, yields a stronger condition (which is more restrictive on the set of valuations for the query) that is known to be sufficient, albeit *not* necessary.

**Condition (PComp-naïve)**

**Assumptions: Let $Q$ be a $\mathsf{UCQ}^{\neq}$ and let $\mathbb{P}$ be a policy.**
For every valuation for $Q$ over $\mathbb{P}$,
there is a node in $\mathsf{net}(\mathbb{P})$ that is responsible for all facts required by the valuation.

Policies that satisfy this condition distribute facts, in general, more generously than they have to ensure the correct evaluation of the query. The next definition captures this property, which we encounter again later.

**Definition* 3.2.6 (Strong parallel completeness).** A query $Q \in \mathsf{UCQ}^{\neq}$ is *strongly parallel-complete* under policy $\mathbb{P}$ if $Q$ and $\mathbb{P}$ satisfy Condition (PComp-naïve). ◀

Furthermore, it is not possible to decide parallel completeness of a *union of* conjunctive queries naïvely by deciding parallel completeness of its subqueries. More precisely, although a union of conjunctive queries is parallel-complete under a given policy $\mathbb{P}$ if all its subqueries are parallel-complete under $\mathbb{P}$, this is not a necessary condition. The latter is witnessed by the query in Example 3.2.3 (on page 26): query $Q$ can be parallel-complete under a policy $\mathbb{P}$ even though it has no node responsible for both facts $R(a,a), S(a,a)$ as long as it has a node that is responsible for the single fact in $R(a,a)$.

Building on the previous observations and, particularly, on the characterisation provided by Theorem 3.2.5, the computational complexity of several variants of the parallel completeness problem for unions of conjunctive queries is studied below.

### Complexity

The remainder of this section is devoted to the proof of the next theorem.

**Theorem 3.2.7.** $\textsc{PComp}(\mathcal{Q},\mathcal{P})$ and $\textsc{PComp}_{\mathrm{i}}(\mathcal{Q},\mathcal{P})$ are $\Pi_2^p$-complete for every query class $\mathcal{Q} \in \{\mathsf{CQ}, \mathsf{CQ}^{\neq}, \mathsf{UCQ}, \mathsf{UCQ}^{\neq}\}$ and every policy class $\mathcal{P} \in \mathfrak{P}$.

*Proof.* The upper bound is implied by the upper bound stated in Proposition 3.2.8 for the most general query class $\mathsf{UCQ}^{\neq}$ and the most general policy class $\mathcal{P}^c_{\mathrm{alg}}$, for arbitrary $c \geq 2$. The lower bound follows from the lower bound stated in Corollary 3.2.13 for the least general query class $\mathsf{CQ}$ and the least general policy class $\mathcal{P}_{\mathrm{list}}$. □

We start with the upper bounds stated in the previous theorem.

**Proposition 3.2.8.** $\textsc{PComp}(\mathsf{UCQ}^{\neq},\mathcal{P})$ and $\textsc{PComp}_{\mathrm{i}}(\mathsf{UCQ}^{\neq},\mathcal{P})$ are in $\Pi_2^p$ for every $\mathcal{P} \in \mathfrak{P}$.

*Proof.* We provide an algorithm for each problem. The general idea is to check parallel completeness either indirectly, via the characterisation by Theorem 3.2.5, for the static variant or directly on the given instance for the dynamic variant. We start with the latter, fixing a policy class $\mathcal{P}$ from $\mathfrak{P}$.

⋙ Dynamic variant. To check parallel completeness of query $Q$ on some instance $\mathcal{G}$ under a policy $\mathbb{P}$ over nodes $k_1, \ldots, k_r$ means to check whether

$$Q((\mathbb{P} \triangleright \mathcal{G})(k_1)) \cup \ldots \cup Q((\mathbb{P} \triangleright \mathcal{G})(k_r)) \supseteq Q(\mathcal{G}) \tag{3.1}$$

holds. For this, it is sufficient to show that, for each fact in the global result set $Q(\mathcal{G})$, there is a node where this fact is derived too. More particularly, if the derivation of a fact on $\mathcal{G}$ is witnessed by some valuation $V$, then its derivation on a local instance is equally witnessed by a—possibly different—valuation $W$.

Algorithm $\mathbb{A}_\mathsf{i}$ tests for the existence of such a valuation $W$ for every valuation $V$ that is relevant, that is, globally satisfying.

**Algorithm $\mathbb{A}_\mathsf{i}$**

**Input:** query $Q \in \mathsf{UCQ}^{\neq}$,
policy $\mathbb{P} \in \mathcal{P}$,
instance $\mathcal{G}$

**Quantified extra input:**

∀ valuation $V$ for $Q$ over $\mathsf{univ}(\mathbb{P})$

∃ valuation $W$ for $Q$ over $\mathsf{univ}(\mathbb{P})$, node $k \in \mathsf{net}(\mathbb{P})$, string $s$

1: **if** $V(\mathsf{pos}(Q)) \not\subseteq \mathcal{G}$ **then** {ignore valuation that is not globally satisfying}
2: **accept**
3: **if** $V(\mathsf{head}(Q)) = W(\mathsf{head}(Q))$ **then**
4: **if** $W(\mathsf{pos}(Q)) \subseteq \mathcal{G}$ and $W(\mathsf{pos}(Q)) \subseteq \mathbb{P}(k)$ **then**
5: **accept**
6: **reject**

The correctness of Algorithm $\mathbb{A}_\mathsf{i}$ is a direct consequence of Equation (3.1), translated to the level of valuations that derive the resulting facts, and the definition of right restrictions. The latter satisfies $(\mathbb{P} \triangleright \mathcal{G})(k) = \mathbb{P}(k) \cap \mathcal{G}$, where a suitable node is specified by the existentially quantified extra input. Hence, the condition in Line 4 holds if and only if valuation $W$ satisfies $Q$ on the local instance of node $k$. Because of Line 3, this valuation witnesses derivation of the fact derived by $V$.

It remains to discuss the worst-case complexity of the algorithm. There are possibly

exponentially many valuations to check. These are, however, handled by the quantified extra input: each valuation can be encoded by a string of polynomial length (bounded by the product of the number of variables of the query and the maximal encoding size of data values in the instance).

Finally, each test routine can be completed in polynomial time. First, the required facts, induced by the valuations, are computed. Then, it is tested whether they all occur in $\mathcal{G}$ and, for valuation $W$, in Line 4, also whether they all occur in $\mathbb{P}(k)$. The former test is obviously polynomial and the latter is too because policy class $\mathcal{P}$ is fixed. We argue the case where $\mathcal{P} = \mathcal{P}^c_{\text{alg}}$, for some $c \in \mathbb{N} - \{1\}$; for all other policy classes, the argument is then also valid. Here, policy $\mathbb{P}$ is represented by $(\mathbb{A}_{\mathbb{P}}, \beta)$, where $\mathbb{A}_{\mathbb{P}}$ is a nondeterministic polynomial time algorithm, which is invoked for each fact required by $W$. For each invocation of $\mathbb{A}_{\mathbb{P}}$, the (existentially quantified) extra input—the solution candidate—of the nondeterministic algorithm can be replaced by successive parts of string $s$.

⋙ Static variant. When parallel completeness should be checked for all instances induced by the policy, the previous approach is not applicable. Here, Condition (PComp) comes to help. Following this characterisation of parallel completeness, it is equivalent to check for every minimal valuation whether there is a node responsible for the facts required. That's precisely what Algorithm $\mathbb{A}$ does.

**Algorithm $\mathbb{A}$**

**Input:** query $Q \in \mathsf{UCQ}^{\neq}$,
policy $\mathbb{P} \in \mathcal{P}$

**Quantified extra input:**

∀ valuation $V$ for $Q$ over $\mathsf{univ}(\mathbb{P})$

∃ valuation $U$ for $Q$ over $\mathsf{univ}(\mathbb{P})$, node $k \in \mathsf{net}(\mathbb{P})$, string $s$

1: **if** $U(\mathsf{head}(Q)) = V(\mathsf{head}(Q))$ and $U(\mathsf{pos}(Q)) \subsetneq V(\mathsf{pos}(Q))$ **then**
2: **accept**
3: **if** $V(\mathsf{pos}(Q)) \subseteq \mathcal{G}$ and $V(\mathsf{pos}(Q)) \subseteq \mathbb{P}(k)$ **then**
4: **accept**
5: **reject**

The correctness of Algorithm $\mathbb{A}$ follows from Theorem 3.2.5, as it is a straightforward implementation of Condition (PComp). The test in Line 1 leads to the ignorance of each valuation $V$ that is not minimal—as witnessed by valuation $U$, which derives the same fact but requires fewer facts. For every other—minimal—valuation $V$, responsibility of the quantified node is tested in Line 3.

The analysis of the worst-case complexity of this algorithm follows the lines of the analysis for $\mathbb{A}_{\mathrm{i}}$ above. □

**Remark 3.2.9.** There is another algorithmic approach to decide PCOMP$_{\mathrm{i}}$, closer to Algorithm $\mathbb{A}$. In this approach, the test of responsibility for required facts is restricted to those *minimal* valuations that are valuations *over the global instance* $\mathcal{G}$. Intuitively, the existentially quantified extra input is then again needed to refute minimality of $V$ where appropriate.

The algorithms specified above are straightforward applications of the definition of parallel completeness or its characterisation. An obvious question is whether they can be improved with respect to their worst-case complexity. Theorem 3.2.5 answers this question negatively: the underlying problems are $\Pi_2^p$-hard.

We prove the lower bounds starting from the hardness of $\Pi_2$-QBF, the standard satisfiability problem for $\Pi_2^p$ [Sto76].

**$\Pi_2$-QBF**

**Input:** formula $\varphi = \forall \boldsymbol{x} \exists \boldsymbol{y}\ \psi(\boldsymbol{x}, \boldsymbol{y})$,
where $\psi$ is a propositional formula in 3-CNF
over propositions $\boldsymbol{x} = (x_1, \ldots, x_r)$ and $\boldsymbol{y} = (y_1, \ldots, y_s)$

**Question:** Does, for every truth assignment $\beta_{\boldsymbol{x}}$ on $\boldsymbol{x}$,
exist a truth assignment $\beta_{\boldsymbol{y}}$ on $\boldsymbol{y}$ such that $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$ satisfies $\psi$?

Note that we begin with a proof of $\Pi_2^p$-hardness for parallel completeness for conjunctive queries *with constants*. The use of constants allows a more direct 'translation' of the satisfiability question. Data values are however not necessary to achieve hardness, as we show in a subsequent step that culminates in Corollary 3.2.13.

But now, queries with constants first.

**Proposition 3.2.10.** PCOMP($\mathsf{CQ}_{\mathrm{dom}}$,$\mathcal{P}_{\mathrm{list}}$) is $\Pi_2^p$-hard, even if restricted to polices over single-node networks.

*Proof.* The mapping described next is a polynomial reduction from $\Pi_2$-QBF and therefore proves the stated hardness result.

Every input formula $\varphi = \forall \boldsymbol{x} \exists \boldsymbol{y}\ \psi$ for $\Pi_2$-QBF with a propositional (quantifier-free) subformula $\psi$ in 3-CNF can be mapped to a query $Q \in \mathsf{CQ}_{\mathrm{dom}}$ and a policy $\mathbb{P} \in \mathcal{P}_{\mathrm{list}}$ in the following way. Let $\boldsymbol{x} = (x_1, \ldots, x_r)$ and $\boldsymbol{y} = (y_1, \ldots, y_s)$ be the propositions of $\psi$, which consists of clauses $C_1, \ldots, C_p$, where $C_j = (\ell_{j,1} \vee \ell_{j,2} \vee \ell_{j,3})$ for every $j \in \{1, \ldots, p\}$.

The database schema underlying query and policy contains a binary relation $\mathtt{Neg}$ and ternary relations $\mathtt{C}_1, \ldots, \mathtt{C}_p$ for the clauses of $\psi$. Constants $0, 1 \in \mathsf{dom}$ are used as truth values. Furthermore, query $Q$ and policy $\mathbb{P}$ alike are based on a common set of facts,

$$\mathcal{F}_{\text{sat}} \stackrel{\text{def}}{=} \{\mathtt{Neg}(0,1), \mathtt{Neg}(1,0)\} \cup \{\mathtt{C}_j(\boldsymbol{b}) \mid j \in \{1,\ldots,p\} \text{ and } \boldsymbol{b} \in \mathbb{B}^+\},$$

which encodes complementary truth values (like for $x_1$ and $\neg x_1$) as well as satisfied clauses. To this end, the set $\mathbb{B}^+ \stackrel{\text{def}}{=} \{0,1\}^3 - \{(0,0,0)\}$ contains all triples of truth values with at least one true component.

Query $Q$ refers to variables $x_1, \ldots, x_r, y_1, \ldots, y_s$ and $\bar{x}_1, \ldots, \bar{x}_r, \bar{y}_1, \ldots, \bar{y}_s$ that represent the positive and negative literals, respectively, that may occur in $\psi$. The query's head $\mathsf{head}(Q) = H(x_1, \ldots, x_r)$ comprises all positive literals over the universally quantified propositions while its body $\mathsf{pos}(Q) = \mathcal{F}_{\text{sat}} \cup \mathcal{A}_{\text{lit}} \cup \mathcal{A}_{\text{clau}}$ comprises the aforementioned facts and, additionally, sets

$$\mathcal{A}_{\text{lit}} \stackrel{\text{def}}{=} \{\mathtt{Neg}(x_i, \bar{x}_i) \mid i \in \{1,\ldots,r\}\} \cup \{\mathtt{Neg}(y_i, \bar{y}_i) \mid i \in \{1,\ldots,s\}\} \text{ and}$$
$$\mathcal{A}_{\text{clau}} \stackrel{\text{def}}{=} \{\mathtt{C}_j(\ell_{j,1}, \ell_{j,2}, \ell_{j,3}) \mid j \in \{1,\ldots,p\}\}$$

of atoms. Here—with slight abuse of notation—each literal $\ell_{j,h} = x_i$ is interpreted as variable $x_i$ and each literal $\ell_{j,h} = \neg x_i$ as variable $\bar{x}_i$, and analogously for literals over propositions $y_1, \ldots, y_s$.

Furthermore, the single-node policy is $\mathbb{P} \stackrel{\text{def}}{=} \{k_{\text{sat}}\} \times \mathcal{F}_{\text{sat}}$. It marks node $k_{\text{sat}}$ as responsible for all facts in $\mathcal{F}_{\text{sat}}$ but no other facts. The policy's universe consists of the truth values 0 and 1.

Clearly, query $Q$ can be computed from $\varphi$ in polynomial time. The same holds for the representation of $\mathbb{P}$ as a list of pairs. Since the mapping is also obviously total, it remains to show that it satisfies the reduction property: $\varphi \in \Pi_2\text{-QBF}$ holds if and only if $(Q, \mathbb{P}) \in \text{PCOMP}(\mathsf{CQ}_{\text{dom}}, \mathcal{P}_{\text{list}})$.

⋙ Only if. Assume that $\varphi \in \Pi_2\text{-QBF}$. We show that $Q$ is parallel-complete under $\mathbb{P}$ then. To this end, let $V$ be an arbitrary minimal valuation for $Q$ over $\mathbb{P}$. The following argument shows that node $k_{\text{sat}}$ is responsible for all facts required by $V$.

Since $\mathsf{univ}(\mathbb{P}) = \{0,1\}$, valuation $V$ naturally induces a truth assignment $\beta_{\boldsymbol{x}}$ on $\boldsymbol{x}$, where $\beta_{\boldsymbol{x}}(x_i) = V(x_i)$ for every $i \in \{1,\ldots,r\}$. Since $\varphi \in \Pi_2\text{-QBF}$, there is a truth assignment $\beta_{\boldsymbol{y}}$ on $\boldsymbol{y}$ such that assignment $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$ satisfies subformula $\psi$.

This combined truth assignment in turn induces a valuation $V'$ that maps each variable $x_i$ and $y_i$ to the truth value defined by $\beta_{\boldsymbol{x}}$ and $\beta_{\boldsymbol{y}}$, respectively, as well as each variable $\bar{x}_i$ and $\bar{y}_i$ to the corresponding complementary values. The latter particularly implies that $V'(\mathcal{A}_{\text{lit}}) \subseteq \{\mathtt{Neg}(0,1), \mathtt{Neg}(1,0)\} \subseteq \mathcal{F}_{\text{sat}}$. Furthermore, for each $j \in \{1,\ldots,p\}$, valuation $V'$ maps at least one variable $\ell_{j,1}, \ell_{j,2}, \ell_{j,3}$ to 1 because $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$ satisfies $\psi$ and thus, in particular, every clause $C_j$. Therefore, also

$V'(\mathcal{A}_{\text{clau}}) \subseteq \mathcal{F}_{\text{sat}}$ and, summarizing, $V'(\mathsf{pos}(Q)) \subseteq \mathcal{F}_{\text{sat}}$. Hence, valuation $V'$ is minimal because, by definition of query $Q$, every valuation requires facts $\mathcal{F}_{\text{sat}}$.

Since valuations $V$ and $V'$ agree on the head variables $x_1, \ldots, x_r$, both valuations are minimal and $V$ requires all facts $\mathcal{F}_{\text{sat}} = V'(\mathsf{pos}(Q))$, we can infer $V \leq V'$. In particular, valuation $V$ requires the same facts as $V'$ and thus only facts for which node $k_{\text{sat}}$ is responsible: $V(\mathsf{pos}(Q)) \subseteq V'(\mathsf{pos}(Q)) \subseteq \mathcal{F}_{\text{sat}} = \mathbb{P}(k_{\text{sat}})$.

Hence, there is a node that is responsible for the facts required by every minimal valuation, that is, query $Q$ is parallel-complete under $\mathbb{P}$ by Condition (PComp).

⋙ If. For a proof by contraposition, assume that $\varphi \notin \Pi_2$-QBF. The following argument shows that query $Q$ is *not* parallel-complete under $\mathbb{P}$ in this case.

Since $\varphi \notin \Pi_2$-QBF, there is a truth assignment $\beta_{\boldsymbol{x}}$ on $\boldsymbol{x}$ such that there is no truth assignment $\beta_{\boldsymbol{y}}$ on $\boldsymbol{y}$ such that $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$ satisfies $\psi$. More specifically, let $\beta_{\boldsymbol{y}}$ be a truth assignment such that $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$ satisfies a maximal number of clauses. The valuation $V$ induced by this combined truth assignment maps $V(x_i) = \beta_{\boldsymbol{x}}(x_i)$ for every $i \in \{1, \ldots, r\}$ and $V(y_i) = \beta_{\boldsymbol{y}}(y_i)$ for every $i \in \{1, \ldots, s\}$ as well as all variables $\bar{x}_1, \ldots, \bar{x}_r$ and $\bar{y}_1, \ldots, \bar{y}_s$ to the corresponding complementary values. Obviously, $V$ is a valuation over $\mathbb{P}$.

Valuation $V$ is defined such that $V(\mathcal{A}_{\text{lit}}) \subseteq \{\mathtt{Neg}(0,1), \mathtt{Neg}(1,0)\}$ and every atom $\mathtt{C}_j(\ell_{j,1}, \ell_{j,2}, \ell_{j,3}) \in \mathcal{A}_{\text{clau}}$ is mapped to a fact $\mathtt{C}_j(\boldsymbol{b})$ for some $\boldsymbol{b} \in \{0,1\}^3$. Furthermore, valuation $V$ maps, for every $j \in \{1, \ldots, p\}$, relation atom $\mathtt{C}_j(\ell_{j,1}, \ell_{j,2}, \ell_{j,3}) \in \mathcal{A}_{\text{clau}}$ to a fact in $\mathcal{F}_{\text{sat}}$ only if assignment $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$ satisfies clause $C_j$. Therefore, there is at least one $j \in \{1, \ldots, p\}$ such that $V(\mathtt{C}_j(\ell_{j,1}, \ell_{j,2}, \ell_{j,3})) = \mathtt{C}_j(0,0,0) \notin \mathcal{F}_{\text{sat}}$ because truth assignment $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$ does not satisfy $\psi$. In particular, the only node $k_{\text{sat}}$ is *not* responsible for all facts required by $V$. The following argument shows however that $V$ is a *minimal* valuation.

For the sake of a contradiction, assume that valuation $V$ is not minimal. There exists then a valuation $U$ such that $U < V$, that is, a valuation that satisfies $U(\mathsf{head}(Q)) = V(\mathsf{head}(Q))$ and $U(\mathsf{pos}(Q)) \subsetneq V(\mathsf{pos}(Q))$. Valuation $U$ then particularly induces a truth assignment $\beta'_{\boldsymbol{x},\boldsymbol{y}}$ on $\boldsymbol{x}$ and $\boldsymbol{y}$ because $U(\mathcal{A}_{\text{lit}}) \subseteq V(\mathcal{A}_{\text{lit}}) = \{\mathtt{Neg}(0,1), \mathtt{Neg}(1,0)\}$. Assignment $\beta'_{\boldsymbol{x},\boldsymbol{y}}$ agrees with $\beta_{\boldsymbol{x}}$ on the universally quantified variables because the valuations agree on the head variables. Since both valuations require identical sets of facts on atoms $\mathcal{F}_{\text{sat}} \cup \mathcal{A}_{\text{lit}}$, we can infer $U(\mathcal{A}_{\text{clau}}) \subsetneq V(\mathcal{A}_{\text{clau}})$. Thus, valuation $U$ requires strictly fewer facts of the form $\mathtt{C}_j(0,0,0)$. This, in turn, implies that truth assignment $\beta'_{\boldsymbol{x},\boldsymbol{y}}$ satisfies more clauses than the truth assignment underlying valuation $V$, which leads to the desired contradiction because $\beta_{\boldsymbol{y}}$ was chosen to satisfy a maximal number of clauses (with respect to $\beta_{\boldsymbol{x}}$).

Therefore, a minimal valuation $V$ exists such that the only existing node $k_{\text{sat}}$ is not responsible for all its required facts. By Condition (PComp), query $Q$ is hence not parallel-complete under policy $\mathbb{P}$.

The provided mapping is thus indeed a polynomial reduction that proves the $\Pi_2^p$-

hardness of $\text{PCOMP}(\mathsf{CQ}_{\text{dom}}, \mathcal{P}_{\text{list}})$. □

**Remark 3.2.11.** The policy in the previous hardness proof is skipping (because no node is responsible for facts $\mathtt{Neg}(0,0)$, $\mathtt{Neg}(1,1)$ or $\mathtt{C}_j(0,0,0)$), Corollary 3.2.13 shows however that skipping is by no means necessary to achieve the hardness result; it only simplifys the argument a bit.

Next, we prove that the question of parallel completeness for queries *with constants* can be reduced to that for queries *without constants*. The growth of the size of the resulting query and the policy, and of the policy's network in particular, is comparatively small. Also, the new query and policy belong to the same classes as the original ones (e.g., a query from $\mathsf{CQ}_{\text{dom}}$ is transformed into a query from $\mathsf{CQ}$).

**Lemma 3.2.12 (Constant elimination).** Fix a policy class $\mathcal{P} \in \mathfrak{P}$. For every query $Q \in \mathsf{UCQ}^{\neq}_{\text{dom}}$ with constants $c_1, \ldots, c_s$ and every policy $\mathbb{P} \in \mathcal{P}$, there is a constant-free query $Q'$ and a complete policy $\mathbb{P}'$ such that the following properties are satisfied *if the maximum arity of relations is viewed as constant.*

1. $Q'$ is parallel-complete under $\mathbb{P}'$ if and only if $Q$ is parallel-complete under $\mathbb{P}$.
2. $Q'$ and $\mathbb{P}'$ have encoding sizes similar to $Q$ and $\mathbb{P}$. More precisely,
   a) $\|Q'\|$ is linear in $\|Q\|$ and
   b) $\|\mathbb{P}'\|$ is polynomial in $s + \|\mathbb{P}\|$.
3. $Q'$ belongs to the constant-free subclass $\mathcal{Q}$ if $Q$ belongs to $\mathcal{Q}_{\text{dom}}$.
4. $\mathbb{P}'$ also belongs to $\mathcal{P}$ and refers to only $s$ additional nodes.

Moreover, both query $Q'$ and policy $\mathbb{P}'$ are computable in polynomial time.

*Proof.* Let $Q$ and $\mathbb{P}$ be a query and policy, respectively, as in the statement of the lemma. Without loss of generality, assume that, first, query $Q$ uses variables $x_1, \ldots, x_r$, second, neither $Q$ nor $\mathbb{P}$ does refer to relations $\mathtt{Const}_1, \ldots, \mathtt{Const}_s$, and, third, the network of $\mathbb{P}$ is $\mathsf{net}(\mathbb{P}) = \{k_1, \ldots, k_n\}$. The query and policy described below satisfy all properties stated in the lemma.

Query $Q'$ is derived from $Q$ by replacing all constants by separate fresh variables $y_1, \ldots, y_s$, that are 'fixed' by unary relation atoms. More formally, if $Q$ is a single conjunctive query, the resulting query is defined as

$$\begin{aligned}
\mathsf{head}(Q') &\stackrel{\text{def}}{=} \sigma(\mathsf{head}(Q)) \\
\mathsf{pos}(Q') &\stackrel{\text{def}}{=} \sigma(\mathsf{pos}(Q)) \cup \{\mathtt{Const}_1(y_1), \ldots, \mathtt{Const}_s(y_s)\}, \\
\mathsf{diseq}(Q') &\stackrel{\text{def}}{=} \sigma(\mathsf{diseq}(Q))
\end{aligned}$$

where $\sigma$ is the substitution $[c_1/y_1, \ldots, c_s/y_s]$. For *unions of* conjunctive queries, each disjunct is modified in this way.

Accordingly, policy $\mathbb{P}'$ extends $\mathbb{P}$ in two ways. First, it lets all nodes of $\mathbb{P}$ be additionally responsible for those $\mathtt{Const}_i$-facts that are meant to represent the original constants. Second, it adds some nodes that are responsible for the other $\mathtt{Const}_i$-facts. Formally, the policy is defined as

$$\mathbb{P}' \stackrel{\text{def}}{=} \mathbb{P} \cup \big(\mathsf{net}(\mathbb{P}) \times \{\mathtt{Const}_1(c_1), \ldots, \mathtt{Const}_s(c_s)\}\big) \cup \mathbb{P}_1 \cup \cdots \cup \mathbb{P}_s$$

where, for each $i \in \{1, \ldots, s\}$, policy $\mathbb{P}_i \stackrel{\text{def}}{=} \{\ell_i\} \times (\mathcal{F} - \{\mathtt{Const}_i(c_i)\})$ marks a single fresh node $\ell_i$ responsible for the set $\mathcal{F}$ of all possible facts over the combined schema $\mathsf{sch}(\mathbb{P}) \cup \{\mathtt{Const}_1, \ldots, \mathtt{Const}_s\}$ and domain $\mathsf{univ}(\mathbb{P}) \cup \{c_1, \ldots, c_s\}$ *with the exception of the single fact* $\mathtt{Const}_i(c_i)$.

Policy $\mathbb{P}$ is complete. First, $\mathbb{P}'$ does not skip any fact over $\mathsf{sch}(\mathbb{P})$ because $s \geq 1$ and thus node $\ell_1 \in \mathsf{net}(\mathbb{P}')$, which is responsible for these facts by definition, exists. Second, node $k_1$ is responsible for fact $\mathtt{Const}_1(c_1)$ and node $\ell_1$ is responsible for every other fact $\mathtt{Const}_i(d)$, where either $i \neq 1$ or $d \neq c_1$.

Query $Q'$ and policy $\mathbb{P}'$ are defined such that they preserve parallel-completeness of $Q$ with respect to $\mathbb{P}$. In principle, this is a consequence of the following two properties, proven in the appendix (Lemma A.1) and Condition (PComp).

(P1) If $V'$ is a minimal valuation for $Q'$ and $V'(y_1, \ldots, y_s) = (c_1, \ldots, c_s)$, then $V'$ is also a minimal valuation for $Q$.

(P2) If $V$ is a minimal valuation for $Q$, then $V[y_1 \mapsto c_1, \ldots, y_s \mapsto c_s]$ is minimal for $Q'$.

To conclude the argument, we show that Property (1) holds: parallel completeness of $Q$ under $\mathbb{P}$ implies parallel completeness of $Q'$ under $\mathbb{P}'$ and vice versa.

First, assume that $Q$ is parallel-complete under $\mathbb{P}$ and let $V'$ be an arbitrary minimal valuation for $Q'$ over $\mathbb{P}'$. Two cases can be distinguished, depending on the tuple $\boldsymbol{d} \stackrel{\text{def}}{=} (d_1, \ldots, d_s) = V'(y_1, \ldots, y_s)$ of data values induced by $V'$: either $\boldsymbol{d} = (c_1, \ldots, c_s)$ or not. In the latter case, there is a position $i \in \{1, \ldots, s\}$ such that $d_i \neq c_i$ and node $\ell_i$ is responsible for all facts $V'(\mathsf{pos}(Q'))$. In the former case, $V'(\sigma(\mathsf{pos}(Q))) = V'(\mathsf{pos}(Q))$ and satisfaction of $\mathsf{diseq}(Q')$ by $V'$ trivially implies satisfaction of $\mathsf{diseq}(Q)$ by $V'$. Thus, valuation $V'$ is consistent for $Q$. Since, by Property (P1), valuation $V'$ is also minimal for $Q$, there is a node $k_i \in \mathsf{net}(\mathbb{P})$ responsible for facts $V'(\mathsf{body}(Q))$ under $\mathbb{P}$—because $Q$ is parallel-complete under $\mathbb{P}$ by assumption. Like every node from $\mathsf{net}(\mathbb{P})$, in the extended policy $\mathbb{P}'$ node $k_i$ is also responsible for the facts $\{\mathtt{Const}_1(c_1), \ldots, \mathtt{Const}_s(c_s)\}$ additionally required by $V'$ for $Q'$. Hence, node $k_i$ is responsible for all facts $V'(\mathsf{pos}(Q'))$, implying that $Q'$ is parallel-complete under $\mathbb{P}'$.

Second, assume that $Q'$ is parallel-complete under $\mathbb{P}'$ and let $V$ be an arbitrary minimal valuation for $Q$. By Property (P2), the induced valuation $V' = V[y_1 \mapsto c_1, \ldots, y_r \mapsto c_r]$ is minimal for $Q'$ and also consistent, satisfying $\mathsf{diseq}(Q')$. Thus there is a node $k \in \mathsf{net}(\mathbb{P}')$ that is responsible for all facts $V'(\mathsf{pos}(Q'))$ because $Q'$ is assumed to be parallel-complete under $\mathbb{P}$. Obviously, valuation $V'$ requires facts $\{\mathtt{Const}_1(c_1), \ldots, \mathtt{Const}_s(c_s)\}$ and thus node $k$ has to be from $\mathsf{net}(\mathbb{P})$. Since $\mathbb{P}'$ extends $\mathbb{P}$ on $k$ only with respect to $\mathtt{Const}_i$-facts and $V(\mathsf{pos}(Q)) \subseteq V'(\mathsf{pos}(Q'))$, node $k$ is responsible for all facts required by $V$ for $Q$ also under the original policy $\mathbb{P}$. Thus, query $Q$ is parallel-complete under $\mathbb{P}$.

It remains to argue that query $Q'$ and policy $\mathbb{P}'$ satisfy Properties (2) – (4). The encoding size of query $Q'$ is obviously linear in $\|Q\|$. The encoding size of $\mathbb{P}'$ increases by the use of $s$ additional nodes and the large set of facts that these nodes are responsible for.[2] Each of the new nodes is responsible for at most $s + \|\mathbb{P}\| \cdot (\|\mathbb{P}\| + s)^r$ facts if $r$ is the maximum arity of relations because there at most $\|\mathbb{P}\|$ different relations and at most $\|\mathbb{P}\| + s$ data values. Furthermore, nodes $k_1, \ldots, k_n$ are responsible for only $s$ additional facts. Therefore, if $r$ is seen as constant, Property (2) holds.

Property (3) readily follows from the fact that query $Q'$ basically 'copies' query $Q$, introducing only relation atoms (no disequality atoms, no negated atoms, ...).

Similarly, Property (4) holds because the extension of policy $\mathbb{P}$ to $\mathbb{P}'$ can be already achieved for a list-based representation (with a polynomial increase of the encoding size) and only nodes $\ell_1, \ldots, \ell_s$ are added to the network of $\mathbb{P}$.

Lastly, it is not hard to see that the complexity of the derivation of query $Q'$ and policy $\mathbb{P}'$ only depends polynomially on their encoding sizes, which are polynomial with respect to $Q$ and $\mathbb{P}$. □

As motivated before, the previous lemma gives us the same lower bound of $\Pi_2^p$ for the parallel completeness problem on conjunctive queries *without constants*.

**Corollary 3.2.13.** $\text{PCOMP}(\mathsf{CQ}, \mathcal{P}_{\text{list}})$ is $\Pi_2^p$-hard, even if restricted to complete policies and networks over only three nodes.

*Proof.* Hardness follows from the $\Pi_2^p$-hardness of $\text{PC}(\mathsf{CQ}_{\text{dom}}, \mathcal{P}_{\text{list}})$ over single-node networks via constant elimination as described above: application of the Lemma 3.2.12 yields an extended policy $\mathbb{P}'$ with two additional nodes, induced by the constants 0 and 1 used in the hardness proof for Proposition 3.2.10, where the arity of relations is bounded by 3 (a constant). □

[2] The analysis for the encoding size of $\mathbb{P}'$ even holds for explicit representations. For the more elaborate representations (rule-based, algorithmic), the increase is only linear in the product of the number of relations and constants—and, in particular, independent of the arity.

The lower bound provided by Proposition 3.2.10 refers to the static variant of parallel completeness. It results from the need to test responsibility for required facts only for *minimal* valuations. The next proposition, which concludes the proof for Theorem 3.2.7, states that the complexity—unfortunately—does *not* decrease for the dynamic variant.

**Proposition 3.2.14.** $\text{PCOMP}_\text{i}(\mathtt{CQ},\mathcal{P}_\text{list})$ is $\Pi_2^p$-hard, even if restricted to complete policies and networks over only two nodes.

*Proof.* Hardness follows again by a polynomial reduction from $\Pi_2$-QBF. Indeed, a slight variation of the reduction provided in the proof of Proposition 3.2.10 shows $\Pi_2^p$-hardness of $\text{PCOMP}_\text{i}$ for queries *with constants* over single-node networks. Finally, we adapt this mapping to satisfy the conditions in the statement to prove.

⋙ With constants. For any given input $\varphi$ for $\Pi_2$-QBF, let $Q'$ be the query and $\mathbb{P}'$ be the policy derived from $\varphi$ by the reduction used in the proof of Proposition 3.2.10. Then, query $Q'$ uses only two constants $c_0 = 0$ and $c_1 = 1$ and policy $\mathbb{P}' = \{k_\text{sat}\} \times \mathcal{F}_\text{sat}$ is defined over one node.

Let $\mathcal{F}_\text{sat}$ be the set of facts defined in the proof of Proposition 3.2.10, which contains, in particular, all encodings of satisfied clauses. Moreover, consider an additional set of facts, $\mathcal{F}_\text{unsat} \stackrel{\text{def}}{=} \{\mathtt{C}_1(0,0,0),\ldots,\mathtt{C}_p(0,0,0)\}$, which encodes unsatisfied clauses in formula $\varphi$.

These sets of facts define policy $\mathbb{P}''$, which results by addition of a new node $k_\text{unsat}$ to $\mathbb{P}'$, yielding

$$\mathbb{P}'' \stackrel{\text{def}}{=} \{k_\text{sat}\} \times \mathcal{F}_\text{sat} \cup \{k_\text{unsat}\} \times (\mathcal{F}_\text{unsat} \cup \{\mathtt{Neg}(0,0), \mathtt{Neg}(1,1)\}),$$

and the global instance $\mathcal{G}' \stackrel{\text{def}}{=} \mathcal{F}_\text{sat} \cup \mathcal{F}_\text{unsat}$ expected for $\text{PC}_\text{i}$. Policy $\mathbb{P}''$ is easily checked to be complete over universe $\{0,1\}$.

The mapping from $\varphi$ to $(Q',\mathbb{P}'',\mathcal{G}')$ as described above is clearly total and computable in polynomial time. Furthermore, it yields a valid input for $\text{PCOMP}_\text{i}$ for queries *with constants*. As the following argument shows, it also satisfies the reduction property: $\varphi \in \Pi_2\text{-QBF}$ holds if and only if $(Q',\mathbb{P}'',\mathcal{G}') \in \text{PCOMP}_\text{i}(\mathtt{CQ}_\text{dom},\mathcal{P}_\text{list})$.

» Only if. Assume that $\varphi \in \Pi_2$-QBF. Let $f$ be an arbitrary fact from the global result $Q'(\mathcal{G}')$. Let $V$ be a *minimal* valuation that witnesses derivation of $f$ on $\mathcal{G}'$. The argument in the proof of Proposition 3.2.10 shows that node $k_\text{sat}$ is responsible for all facts required by $V$. Therefore, fact $f$ is also derived locally at this node because $\mathcal{G}'$ contains already all facts that $k_\text{sat}$ is responsible for and thus $(\mathbb{P}'' \triangleright \mathcal{G}')(k_\text{sat}) = \mathbb{P}'(k_\text{sat})$ holds. Hence, query $Q'$ is parallel-complete on $\mathcal{G}'$ under policy $\mathbb{P}''$.

» If. Towards a proof by contraposition, assume that $\varphi \notin \Pi_2$-QBF. Using this assumption, the proof of Proposition 3.2.10 shows that there is a minimal valuation $V$ for query $Q'$ that requires only facts from $\mathcal{G}'$ and thus proves global derivation of a fact $f = H(b_1, \ldots, b_r)$ where $(b_1, \ldots, b_r) = \beta_{\boldsymbol{x}}(x_1, \ldots, x_r)$ for a truth assignment $\beta_{\boldsymbol{x}}$ that witnesses $\varphi \notin \Pi_2$-QBF. The same argument shows that $V$ requires at least one $\mathtt{C}_j(0,0,0)$-fact and, in particular, that every other valuation agreeing with $V$ on the head variables $x_1, \ldots, x_r$ (irrespective of its minimality) requires one or more facts $\mathtt{C}_{j_1}(0,0,0), \ldots, \mathtt{C}_{j_s}(0,0,0)$ too. Therefore, no valuation derives fact $f$ on the local instance of $k_{\text{sat}}$, which is disjoint from $\mathcal{F}_{\text{unsat}}$. We turn to the only node remaining for policy $\mathbb{P}''$: the local instance of $k_{\text{unsat}}$ does not contain facts $\mathtt{Neg}(0,1)$ and $\mathtt{Neg}(1,0)$ required for atoms $\mathcal{F}_{\text{sat}}$ in $Q'$. Hence, fact $f$ is derived there neither, and query $Q'$ is thus *not* parallel-complete on $\mathcal{G}'$ under $\mathbb{P}''$.

This proves that $\text{PCOMP}_\text{i}(\mathtt{CQ}_{\text{dom}}, \mathcal{P}_{\text{list}})$ is $\Pi_2^p$-hard, even for complete policies and networks over only two nodes. It remains to extend this hardness result onto constant-free queries.

⋙ Without constants. Eventually, a query $Q$ free from constants $c_0 = 0$ and $c_1 = 1$ can be derived from $Q'$ along the lines of the constant-elimination principle (Lemma 3.2.12). Furthermore, policy $\mathbb{P}''$ can be extended to policy

$$\mathbb{P} \stackrel{\text{def}}{=} \mathbb{P}'' \cup \{k_{\text{sat}}\} \times \{\mathtt{Const}_0(0), \mathtt{Const}_1(1)\} \cup \{k_{\text{unsat}}\} \times \{\mathtt{Const}_0(1), \mathtt{Const}_1(0)\},$$

which is also complete over universe $\{0,1\}$. Lastly, instance $\mathcal{G}'$ is extended to $\mathcal{G} \stackrel{\text{def}}{=} \mathcal{G}' \cup \{\mathtt{Const}_0(0), \mathtt{Const}_1(1)\}$. This yields a mapping from an arbitrary formula $\varphi$ for $\Pi_2$-QBF to a valid input $(Q, \mathbb{P}, \mathcal{G})$ for $\text{PC}_\text{i}(\mathtt{CQ}, \mathcal{P}_{\text{list}})$.

The new facts do not affect the above argument for the reduction property. First, valuations as defined for *(only if)* additionally require facts $\mathtt{Const}_0(0)$ and $\mathtt{Const}_1(1)$, also available on $k_{\text{sat}}$. Second, facts $\mathtt{Const}_0(1)$ and $\mathtt{Const}_1(0)$ cannot compensate the missing $\mathtt{Neg}$-facts for the valuation used for *(if)*.

Therefore, $\text{PC}_\text{i}(\mathtt{CQ}, \mathcal{P}_{\text{list}})$ is $\Pi_2^p$-hard, even for complete policies over only two nodes. □

The previous results settle the complexity of the static and dynamic variant of the parallel completeness problem for conjunctive queries. Both variants are $\Pi_2^p$-complete. Allowing queries to contain also disequalities or unions or even both does not change that picture. The picture *does* change, however, if we turn to fragments of $\mathsf{UCQ}^{\neq}$—fragments in particular that trivialise the question for minimality of valuations, as discussed next.

### 3.2.2. Strongly minimal queries

The algorithms presented above indicate that the complexity of the parallel completeness problem is significantly determined by the need to verify the minimality for the tested valuations. The corresponding lower bounds support this indication.

From another point of view, this observation provides a lever to lower the complexity of the parallel completeness problem, namely by restriction to fragments where the need to test minimality vanishes because every valuation is guaranteed to be minimal. This is captured by the following definition.

**Definition* 3.2.15 (Strongly minimally requiring).** A query $Q \in \mathsf{UCQ}^{\neg,\neq}$ is *strongly minimally requiring* if every valuation for it is minimally requiring. ◀

As for minimal valuations, we simply say *strongly minimal* for negation-free queries.

It has been shown that deciding strong minimality is coNP-complete [AGK+17a]. The lower bound already holds for conjunctive queries and the upper bound is obvious: a nondeterministic algorithm can—if existent—easily guess two valuations that contradict strong minimality (to decide the complement problem).

For some queries, however, it is easy to determine that they are strongly minimal. Prominent examples are full queries (without projection variables) and queries without self-joins (where every relation symbol occurs in at most one atom). The following lemma provides a sufficient condition that generalises these simple observations.

**Lemma* 3.2.16 (Sufficient condition for strong minimality).** A query $Q \in \mathsf{CQ}$ is strongly minimal if every variable $x$ of the query is a head variable or there is a relation symbol $R$ and a position $p$ such that $x$ is the only variable at position $p$ in the $R$-atoms of $\mathsf{pos}(Q)$.

As an example, query $H(x, y) \leftarrow R(x), R(y), S(u, x), S(u, u), T(v)$ is strongly minimal by the previous lemma: the non-head variable $u$ is the only variable that occurs in position 1 of the query's $S$-atoms and non-head variable $v$ is the only variable that occurs in a $T$-atom.

Strong minimality leads to a lower complexity for the parallel completeness problem because Condition (PComp) and Condition (PComp-naïve) are obviously equivalent for this query class. The latter condition however can be tested by simple adaptations of Algorithms $\mathbb{A}_{\mathrm{i}}$ and $\mathbb{A}$ that do not need an existentially quantified extra input (if not for the policy).

**Proposition* 3.2.17.** Both $\textsc{PComp}(\mathsf{CQ}_{[\mathsf{sm}]},\mathcal{P})$ and $\textsc{PComp}_{\mathrm{i}}(\mathsf{CQ}_{[\mathsf{sm}]},\mathcal{P})$ are in coNP for every policy class $\mathcal{P} \in \mathfrak{P}_{\mathrm{det}}$.

Note that the result above only holds for policies that are represented by *deterministic* polynomial time algorithms. Of course, these upper bounds obviously also hold for the larger class $\mathsf{UCQ}^{\neq}{}_{[\mathsf{sm}]}$.

We consider strongly minimal queries again when we study parallel-correctness *transfer* (Section 4.2.2). Meanwhile, we extend our study of the parallel correctness problem to queries with negation.

## 3.3. Unions of conjunctive queries with negation

In the previous section, we have considered the parallel correctness problem for CQs and UCQs, possibly with disequalities. The situation was rather comfortable. First, these queries are trivially parallel-sound because they are monotonic. Second, the characterisation in terms of minimal valuations by Theorem 3.2.5 offered a means to verify or falsify parallel completeness by sets of 'very small' instances.

The situation changes when queries contain negation. Now, parallel soundness is not guaranteed but has to be checked on its own. Furthermore, the instances that have to be considered may become 'quite large'. Both aspects are addressed in Section 3.3.1. Later, in Sections 3.3.2 and 3.3.3, we show that the second aspect is irrelevant for some queries, where only *one* 'small' instance is relevant or negation is used in a 'controlled' fashion, respectively.

### 3.3.1. General case

The following example demonstrates that, as to be expected, conjunctive queries with negation are not guaranteed to be parallel-sound. Beyond that, we argue that the size of an instance witnessing violation of parallel soundness can be necessarily exponential—in the size of the query.

**Example 3.3.1 (Exponentially large counterexamples).** In the following, we define a sequence $(Q_1, \mathbb{P}_1), (Q_2, \mathbb{P}_2), \ldots$ of query-policy pairs. For every $n \in \mathbb{N}$, let

$$Q_n \stackrel{\text{def}}{=} H() \leftarrow R(x_1), \ldots, R(x_n), R(y_1), \ldots, R(y_n), S(x_1, \ldots, x_n), \neg S(y_1, \ldots, y_n)$$

be a conjunctive query with negation over the unary relation $R$ and the $n$-ary relation $S$. Furthermore, let $\mathbb{P}_n$ be the policy over network $\{k_0, k_1\}$, where node $k_0$ is solely responsible for fact $S(0, \ldots, 0)$ while node $k_1$ is responsible for all other facts over the policy's universe $\{0, 1\}$. More concretely,

$$\begin{aligned}\mathbb{P}_n \stackrel{\text{def}}{=}\ & \{k_0\} \times \{S(0, \ldots, 0)\} \\ & \cup\ \{k_1\} \times \big(\{S(\boldsymbol{b}) \mid \boldsymbol{b} \in \{0,1\}^n - \{(0, \ldots, 0)\}\} \cup \{R(0), R(1)\}\big).\end{aligned}$$

Clearly, policy $\mathbb{P}_n$ is complete and can be represented by a linear number of rules: for every $i \in \{1, \ldots, n\}$, there is a rule $S(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) \to k_1$ to mark node $k_1$ responsible for every $S$-fact but $S(0, \ldots, 0)$. Two more rules address relation $R$.

For every $n \in \mathbb{N}$, query $Q_n$ is *not* parallel-sound under $\mathbb{P}_n$, as the following argument shows. Let global instance $\mathcal{G}_n \stackrel{\text{def}}{=} \mathsf{facts}(\mathbb{P}_n)$ comprise all facts over the schema of $\mathbb{P}_n$ such that $\mathbb{P}_n \triangleright \mathcal{G}_n = \mathbb{P}_n$ holds in particular.

On the one hand, no fact is derived globally because *no* valuation $V$ over $\{0,1\}$ satisfies $Q_n$ globally on $\mathcal{G}_n$ since it prohibits fact $V(S(y_1,\ldots,y_n)) = S(b_1,\ldots,b_n)$, for some $b_1,\ldots,b_n \in \{0,1\}$, which is present in $\mathcal{G}_n$.

On the other hand, there is a valuation that satisfies $Q_n$ locally. One such valuation is $W \stackrel{\text{def}}{=} \{x_1 \mapsto 1,\ldots,x_n \mapsto 1, y_1 \mapsto 0,\ldots,y_n \mapsto 0\}$, which requires facts $W(\mathsf{pos}(Q_n)) = \{R(0), R(1), S(1,\ldots,1)\} \subseteq \mathbb{P}_n(k_1)$ and prohibits fact $W(\mathsf{neg}(Q_n)) = S(0,\ldots 0) \notin \mathbb{P}_n(k_1)$.

Therefore, the distributed result is not a subset of the global result,

$$\bigsqcup Q_n(\mathbb{P}_n \triangleright \mathcal{G}_n) = \bigsqcup Q_n(\mathbb{P}_n) = \{H()\} \not\subseteq \emptyset = Q_n(\mathcal{G}_n), \tag{3.2}$$

witnessing that $Q_n$ is not parallel-sound under $\mathbb{P}_n$.

Furthermore, instance $\mathcal{G}_n$ is the only instance over $\{0,1\}$ with this property. For a proof by contradiction, assume existence of a global instance $\mathcal{G}'_n \subsetneq \mathcal{G}_n$ that satisfies $Q_n$ locally but not globally. A locally satisfying valuation $V$ has to map at least two variables $x_i$ and $y_i$ to different values (otherwise it is inconsistent). This valuation then requires facts $R(0), R(1)$ and $S(a_1,\ldots,a_n)$ for some $a_1,\ldots,a_n \in \{0,1\}$, which have to be in local instance $(\mathbb{P}_n \triangleright \mathcal{G}'_n)(k_1)$ and thus also in global instance $\mathcal{G}'_n$. These observations with assumption $\mathcal{G}'_n \subsetneq \mathcal{G}_n$ imply that there is a fact $S(b_1,\ldots,b_n) \notin \mathcal{G}'_n$ for some $b_1,\ldots,b_n \in \{0,1\}$. Hence, valuation $W \stackrel{\text{def}}{=} \{x_1 \mapsto a_1,\ldots,x_n \mapsto a_n, y_1 \mapsto b_1,\ldots,y_n \mapsto b_n\}$ is consistent, because $(a_1,\ldots,a_n) \neq (b_1,\ldots,b_n)$, and satisfies query $Q_n$ globally. Therefore, the locally derived fact $H()$ is also derived globally, contradicting Equation (3.2). Since the query is boolean, this is the only derivable fact. Thus, as claimed, there is no other instance than $\mathcal{G}_n$ satisfying this property. In particular, there is no *smaller* instance.

In summary, the sequence $(Q_n, \mathbb{P}_n)_{n\in\mathbb{N}}$ of query-policy pairs requires a sequence $(\mathcal{G}_n)_{n\in\mathbb{N}}$ of global instances of size $\|\mathcal{G}_n\| \in \Omega(\mathsf{exp}(\|Q_n\|))$ to witness the violation of parallel soundness. ■

The previous example diminishes the hope to characterise parallel soundness by (minimal) valuations in a fashion highly analogous to that in Condition (PComp)—at least in the general case. Indeed, the arguments below show that this is not specific to parallel *soundness* but affects parallel completeness and parallel correctness alike.

It is, however, notable that the argument requires the arity of relation $S$ to grow unboundedly. Correspondingly, we will later consider two scenarios separately: the special case, where the arity of the queries' relations is bounded in advance, and the general case, where there is no such bound.

Helpful to achieve an upper bound in both cases is the next lemma.

**Lemma 3.3.2 (Witnesses with small domain).** Let $Q \in \mathsf{UCQ}^{\neg,\neq}$ be a query and let $\mathbb{P}$ be an arbitrary policy. The following statements hold if $d$ is the maximum number of variables in disjuncts of $Q$.

1. If $Q$ is not parallel-sound under $\mathbb{P}$,
   then there is an instance $\mathcal{G}^*$ with at most $d$ data values witnessing that.
2. If $Q$ is not parallel-complete under $\mathbb{P}$,
   then there is an instance $\mathcal{G}^*$ with at most $d$ data values witnessing that.

*Proof.* Let $Q$ be a query over some schema $\mathcal{S}$. The following argument proves Statement 2, the proof of Statement 1 is analogous.

Assume that $Q$ is not parallel-complete under $\mathbb{P}$. By definition of parallel-completeness, there is global instance $\mathcal{G}$ witnessing incompleteness: $\bigsqcup Q(\mathbb{P} \triangleright \mathcal{G}) \not\supseteq Q(\mathcal{G})$. Thus, there exists a fact $f \in Q(\mathcal{G})$ that is derived globally by some valuation $V$ but on none of the resulting local instances. The set $U = \mathsf{dom}(V(\mathsf{pos}(Q)))$ of data values referred to by $V$ then clearly contains at most $d$ data values. The same is true for the induced subinstance $\mathcal{G}^* \stackrel{\text{def}}{=} \mathcal{G} \cap \mathsf{facts}(\mathcal{S}, U)$.

Instance $\mathcal{G}^*$ also witnesses the violation of parallel-completeness of $Q$ under $\mathbb{P}$. This is a consequence of the following observation: for every fact $f \in \mathsf{facts}(\mathcal{S}, U)$ and every node $k \in \mathsf{net}(\mathbb{P})$ the following equivalences,

$$f \in \mathcal{G} \iff f \in \mathcal{G}^* \text{ and } f \in (\mathbb{P} \triangleright \mathcal{G})(k) \iff f \in (\mathbb{P} \triangleright \mathcal{G}^*)(k), \tag{3.3}$$

hold.

First, valuation $V$ from above refers only to facts in $\mathsf{facts}(\mathcal{S}, C)$ and is thus, by the left-hand side of Observation (3.3), satisfying $Q$ on $\mathcal{G}^*$ because it is satisfying on $\mathcal{G}$. Second, there is no valuation that derives $f$ on the local instance $(\mathbb{P} \triangleright \mathcal{G}^*)(k)$ of some node $k$. Assume, towards a contradiction, that such a valuation $W$ exists. Then, $W$ requires facts from $\mathcal{G}^* \subseteq \mathsf{facts}(\mathcal{S}, U)$ and, since $Q$ is a query with *safe* negation—and therefore every variable in a negated atom also occurs in a positive atom—, prohibits only facts from $\mathsf{facts}(\mathcal{S}, U)$. The right-hand side of Observation (3.3) implies then, that $W$ is also satisfying on local instance $(\mathbb{P} \triangleright \mathcal{G})(k)$, contradicting the choice of fact $f$.

Therefore, the restricted global instance contains only $d$ data values and also witnesses violation of parallel completeness, $\bigsqcup Q(\mathbb{P} \triangleright \mathcal{G}^*) \not\supseteq Q(\mathcal{G}^*)$, as claimed. □

This upper bound *on the number of data values* in instances that witness the violation of parallel soundness or parallel completeness immediately implies an upper bound on the size of the instances, that is, *on the number of facts* they contain. The latter, in turn, is the key to the following upper bounds for the associated decision problems.

**Proposition 3.3.3.** For every $\mathcal{P} \in \mathfrak{P}$, the following problems are in coNEXPTIME.

- PSound($\mathsf{UCQ}^{\neg,\neq},\mathcal{P}$)
- PComp($\mathsf{UCQ}^{\neg,\neq},\mathcal{P}$)
- PC($\mathsf{UCQ}^{\neg,\neq},\mathcal{P}$)

If the input is restricted to schemas with a fixed maximal arity, these problems are in $\Pi_2^p$.

*Proof.* We describe an algorithm that decides PSound($\mathsf{UCQ}^{\neg,\neq},\mathcal{P}$) for a fixed policy class $\mathcal{P} \in \mathfrak{P}$. Without loss of generality, we assume that $\mathcal{P} = \mathcal{P}^c_{\text{alg}}$, for some $c$ (all other representations can be easily transformed into a nondeterministic algorithm with a sufficient runtime bound).

Algorithm $\mathbb{A}^{\text{sound}}_{\neg}$ below tests whether each locally satisfying valuation has a corresponding globally satisfying valuation. It is formulated for the restricted case, using two types of quantified extra input. The generalisation is described at the end of this proof.

**Algorithm $\mathbb{A}^{\text{sound}}_{\neg}$**

**Input:** query $Q \in \mathsf{UCQ}^{\neg,\neq}$,
policy $\mathbb{P} \in \mathcal{P}$

**Quantified extra input:**

∀ valuation $V$ for $Q$ over $\mathbb{P}$, instance $\mathcal{G}^*$ over $\mathsf{dom}(V(\mathsf{pos}(Q)))$, node $k \in \mathsf{net}(\mathbb{P})$

∃ valuation $W$ for $Q$ over $\mathbb{P}$

1: **if** $V$ satisfies $Q$ on $(\mathbb{P} \triangleright \mathcal{G}^*)(k)$ **then**
2: **if** $W$ derives the same fact as $V$ and $W$ satisfies $Q$ on $\mathcal{G}^*$ **then**
3: **accept**
4: **else**
5: **reject**
6: **accept**

The correctness of Algorithm $\mathbb{A}^{\text{sound}}_{\neg}$ results from the following observations. If query $Q$ is *not* parallel-sound under policy $\mathbb{P}$, then there is an instance $\mathcal{G}$ over $\mathbb{P}$ and a fact $f$ such that this fact is derived by a locally satisfying valuation $V$ (on some node $k$) but by no globally satisfying valuation. A closer look into the proof of Lemma 3.3.2 reveals that there is even an instance $\mathcal{G}^*$ that contains only data values referred to by valuation $V$.

As part of the universally quantified extra input, valuation $V$, instance $\mathcal{G}^*$ and node $k$ are considered by the algorithm. In Line 1, the algorithm tests whether $V$ is locally satisfying. If $V$ is not locally satisfying, nothing has to be checked further and the algorithm accepts in Line 6. Otherwise, the algorithm tests whether the existentially quantified valuation $W$ witnesses the derivation of $V(\mathsf{head}(Q))$ on the global instance. Since no such valuation $W$ can be found, the algorithm rejects in Line 5.

If query $Q$ is parallel-sound on the contrary, then obviously either Line 2 is not reached or a valuation $W$ that satisfies the test in Line 2 exists and leads to acceptance in Line 3.
For schemas with a fixed upper bound $\alpha$ on the arity of their relations, $\mathbb{A}_{\neg}^{\text{sound}}$ is a $\Pi_2^p$-algorithm: the number of facts in instance $\mathcal{G}^*$ is bounded by $rd^\alpha$ where $r$ is the number of relation symbols and $d$ is the maximal number of variables in $Q$. This gives a bound $\|Q\|^{1+\alpha}$, which is polynomial. Otherwise, the arity $\alpha$ is only bounded by the size of the query, which yields an exponential bound $\|Q\|^{1+\|Q\|}$ on the size of $\mathcal{G}^*$.

Furthermore, towards the coNEXPTIME-upper bound, the existential quantified extra input $W$ can be replaced by a deterministic procedure that checks all valuations over $\mathbb{P}$ deterministically in exponential time. □

So far, we have seen that the violation of parallel correctness and its variants can be witnessed by at most exponentially large instances in the general case. Moreover, Example 3.3.1 demonstrates that sometimes there are not (significantly) smaller instances with that property. Of course, there could be a different, more efficient approach to decide parallel correctness. However, the algorithm above seems to be optimal because the problems are complete for the respective classes.

**Theorem 3.3.4.** The following problems are coNEXPTIME-complete, for every query class $\mathcal{Q} \in \{\mathsf{CQ}^\neg, \mathsf{CQ}^{\neg,\neq}, \mathsf{UCQ}^\neg, \mathsf{UCQ}^{\neg,\neq}\}$ and every $\mathcal{P} \in \mathfrak{P} - \{\mathcal{P}_{\text{list}}\}$.

- PSound$(\mathcal{Q},\mathcal{P})$
- PComp$(\mathcal{Q},\mathcal{P})$
- PC$(\mathcal{Q},\mathcal{P})$

If the input is restricted to schemas with a fixed maximal arity, these problems are $\Pi_2^p$-complete.

*Proof.* The upper bounds readily follow from Proposition 3.3.3, as they already hold for the most general class $\mathsf{UCQ}^{\neg,\neq}$ of queries. The lower bounds follow from the respective lower bounds supplied by Theorem* 3.3.5 below and the polynomial reducibility, stated in the subsequent Proposition 3.3.6. □

Note that the $\Pi_2^p$-hardness of parallel *completeness* and parallel *correctness* is already implied by the corresponding lower bounds for conjunctive queries *without negation* (Theorem 3.2.7).

The hardness results stated above—and others to follow—are proven by reductions from a classical static analysis problem in database theory: query containment. This problem is formally defined below.

| Contain$(\mathcal{Q}_1, \mathcal{Q}_2)$ | |
|---|---|
| **Parameters:** | query classes $\mathcal{Q}_1$ and $\mathcal{Q}_2$ |
| **Input:** | queries $Q_1 \in \mathcal{Q}_1$ and $Q_2 \in \mathcal{Q}_2$ |
| **Question:** | Does $Q_1 \sqsubseteq Q_2$ hold? |

Although well-studied over the past decades, some relatively recent results shed new light on the exact complexity of the containment problem for conjunctive queries with negation, unions of them and several fragments—described in more detail later.

Most notably, for conjunctive queries *with negation*, Ketsman has shown the problem to be coNEXPTIME-complete—if the schemas underlying the input queries are not assumed to obey a fixed upper arity bound [GKNS19]. This contrasts with a previously stated $\Pi_2^p$-completeness [MST12, Ull00]—which albeit holds if there *exists* a fixed arity bound.

**Theorem* 3.3.5.** Problem Contain$(\mathsf{CQ}^\neg, \mathsf{CQ}^\neg)$ is coNEXPTIME-complete in general, even if restricted to boolean queries. If the input is restricted to schemas with a fixed maximal arity, the problem is $\Pi_2^p$-complete.

This complexity results allows us to establish lower bounds that match the upper bounds provided by Proposition 3.3.3 via the next result.

**Proposition 3.3.6.** Contain$(\mathsf{CQ}^\neg, \mathsf{CQ}^\neg)$, restricted to boolean queries, is polynomially reducible to each of the following problems.

- PSound$(\mathsf{CQ}^\neg, \mathcal{P}_{\text{rule}})$
- PComp$(\mathsf{CQ}^\neg, \mathcal{P}_{\text{rule}})$
- PC$(\mathsf{CQ}^\neg, \mathcal{P}_{\text{rule}})$

In particular, the reduction yields boolean queries and policies over only four nodes.

*Proof.* We provide a single mapping that serves as a reduction for all three problems alike, parallel completeness, parallel soundness and parallel correctness. This mapping is described next.

Let $Q_1$ and $Q_2$ be two boolean queries from $\mathsf{CQ}^\neg$ forming some input of Contain. We assume, without loss of generality, that both queries are formulated over disjoint sets of variables and that the schema $\mathcal{S}_1$ of $Q_1$ subsumes the schema of $Q_2$ (otherwise $Q_1$ is trivially *not* contained in $Q_2$). Furthermore, let $m$ be the maximum number of variables in $Q_1$ or $Q_2$. Without restriction, $m \geq 3$ is assumed (otherwise atoms $R(x_1, x_2, x_3)$ and $R(y_1, y_2, y_2)$ with a fresh relation relation symbol $R$ and fresh variables could be added to $Q_1$ and $Q_2$, respectively). Finally, both queries are assumed to be consistent, that is, none of them contains the same relation atom positively *and* negated. This latter condition can be tested in polynomial time. If it is violated, the queries are mapped to a fixed yes-instance (if $Q_1$ is inconsistent) or to a fixed no-instance (otherwise).

The idea underlying the reduction mapping is to combine queries $Q_1$ and $Q_2$ into a single query $Q$ such that it can, although connected *conjunctively*, emulate derivation by its constituents 'independently'—under a suitable policy. To this end, the schema of $Q$ is defined as the union $\mathcal{S} \cup \mathcal{T}$ of two schemas, without restriction assumed to be disjoint. Schema $\mathcal{S}$ is suited to the application of the $\mathsf{ext}$-operator (defined in Section 2.1.1) to the atoms of queries $Q_1$ and $Q_2$ and thus results from $\mathcal{S}_1$ by increasing each relation symbol's arity by 1. Schema $\mathcal{T}$ contains unary relation symbols $\mathtt{Start}_1$, $\mathtt{Start}_2$, $\mathtt{Stop}$ and $\mathtt{Type}$, which are meant to guide the emulated derivation.

The boolean query $Q$ with $\mathsf{head}(Q) \stackrel{\text{def}}{=} H()$ and

$$\mathsf{pos}(Q) \stackrel{\text{def}}{=} \mathsf{ext}_{\alpha_1}(\mathsf{pos}(Q_1)) \cup \mathsf{ext}_{\alpha_2}(\mathsf{pos}(Q_2)) \cup \mathcal{A}^+ \text{ and}$$
$$\mathsf{neg}(Q) \stackrel{\text{def}}{=} \mathsf{ext}_{\alpha_1}(\mathsf{neg}(Q_1)) \cup \mathsf{ext}_{\alpha_2}(\mathsf{neg}(Q_2)) \cup \mathcal{A}^-$$

comprises the bodies of queries $Q_1$ and $Q_2$ extended by different fresh 'label' variables $\alpha_1$ and $\alpha_2$ as well as the additional sets of atoms

$$\mathcal{A}^+ \stackrel{\text{def}}{=} \{\mathtt{Start}_1(\alpha_1), \mathtt{Start}_2(\alpha_1), \mathtt{Start}_2(\alpha_2), \mathtt{Type}(\tau)\} \text{ and}$$
$$\mathcal{A}^- \stackrel{\text{def}}{=} \{\neg\mathtt{Stop}(\alpha_1), \neg\mathtt{Stop}(\alpha_2)\}$$

over schema $\mathcal{T}$, where $\tau$ is another fresh variable.

Policy $\mathbb{P}$ is defined as the union $\mathbb{P}_\mathcal{S} \cup \mathbb{P}_\mathcal{T}$ of two policies over a common network $N \stackrel{\text{def}}{=} \{c, s, r_1, r_2\}$ and universe $U \stackrel{\text{def}}{=} \mathsf{dom}_\mathsf{m}$, which contains at least as many data values as there are variables in any disjunct of $Q_1$. The policy is defined such that at most one node can be responsible for every valuation $V$ for $Q$: the responsible node is determined by both values $V(\alpha_1)$ and $V(\tau)$ and thus governed by sub-policy $\mathbb{P}_\mathcal{T}$, defined by

$$\begin{aligned}
\mathbb{P}_{\mathcal{T}}(c) &\stackrel{\text{def}}{=} \{\mathtt{Start}_1(1), \mathtt{Type}(1)\} \\
&\quad \cup \{\mathtt{Start}_2(1), \mathtt{Stop}(1)\}, \\
\mathbb{P}_{\mathcal{T}}(s) &\stackrel{\text{def}}{=} \{\mathtt{Start}_1(1), \mathtt{Type}(2)\} \\
&\quad \cup \{\mathtt{Start}_2(1), \ldots, \mathtt{Start}_2(m), \mathtt{Stop}(1)\}, \\
\mathbb{P}_{\mathcal{T}}(r_1) &\stackrel{\text{def}}{=} \{\mathtt{Start}_1(1), \mathtt{Type}(3), \ldots, \mathtt{Type}(m)\} \\
&\quad \cup \{\mathtt{Start}_2(1), \ldots, \mathtt{Start}_2(m), \mathtt{Stop}(1), \ldots, \mathtt{Stop}(m)\} \text{ and} \\
\mathbb{P}_{\mathcal{T}}(r_2) &\stackrel{\text{def}}{=} \{\mathtt{Start}_1(2), \ldots, \mathtt{Start}_1(m), \mathtt{Type}(1), \ldots, \mathtt{Type}(m)\} \\
&\quad \cup \{\mathtt{Start}_2(1), \ldots, \mathtt{Start}_2(m), \mathtt{Stop}(1), \ldots, \mathtt{Stop}(m)\}.
\end{aligned}$$

Sub-policy $\mathbb{P}_{\mathcal{S}} \stackrel{\text{def}}{=} N \times \mathsf{facts}(\mathcal{S}, U)$ trivially marks each of the four nodes responsible for all facts over the extended input schema $\mathcal{S}$. Policy $\mathbb{P}$ is easily verified to be complete.

Query $Q$ and policy $\mathbb{P}$ can be computed in polynomial time from $(Q_1, Q_2)$ and the mapping thus defined is clearly total. Hence, it remains to prove the reduction property: $(Q_1, Q_2) \in \textsc{Contain}(\mathtt{CQ}^{\neg}, \mathtt{CQ}^{\neg})$ holds if and only if both $(Q, \mathbb{P}) \in \textsc{PComp}(\mathtt{CQ}^{\neg}, \mathcal{P}_{\text{rule}})$ and $(Q, \mathbb{P}) \in \textsc{PSound}(\mathtt{CQ}^{\neg}, \mathcal{P}_{\text{rule}})$ hold. The hardness result for $\textsc{PC}(\mathtt{CQ}^{\neg}, \mathcal{P}_{\text{rule}})$ is then implied as it is defined as the intersection of the former problem variants.

For the proof of the equivalence, two observations are helpful. They build on the following distinction of valuations for the derived query $Q$. In the context of this proof, a valuation $V$ for $Q$ is *trivial* if $V(\alpha_1) \geq 2$ or $V(\tau) \geq 3$.

First, policy $\mathbb{P}$ is 'well-behaved' with respect to *all* trivial valuations, regardless of whether containment $Q_1 \sqsubseteq Q_2$ holds or not. A trivial valuation $V$ affects neither parallel completeness nor parallel soundness because it requires—among others—facts $V(\mathtt{Start}_1(\alpha_1))$ and $V(\mathtt{Type}(\tau))$. There is only one node responsible for these facts, either node $r_1$ or node $r_2$. But this node is then responsible for all facts required or prohibited by the valuation because, by definition of $\mathbb{P}$, it is responsible for all $\mathtt{Start}_2$-, all $\mathtt{Stop}$-facts and all $\mathcal{S}$-facts. Hence, the following property holds.

(P0) A trivial valuation that is locally satisfying
is also globally satisfying, and vice versa.

Second, policy $\mathbb{P}$ is 'well-behaved' with respect to *all* non-trivial valuations if and only if $Q_1 \sqsubseteq Q_2$ does hold. This is shown below and it is mainly a consequence of the next two properties of reduced instances (defined in Chapter 2).

(P1) If a non-trivial valuation $V$ satisfies query $Q$ on instance $\mathcal{G}$,
then $V$ also satisfies both $Q_1$ on $\mathsf{red}_1(\mathcal{G})$ and $Q_2$ on $\mathsf{red}_a(\mathcal{G})$ for $a = V(\alpha_2)$.

(P2) If $\mathcal{G} \supseteq \{\mathtt{Start}_1(1), \mathtt{Start}_2(a), \mathtt{Type}(b)\}$ and $\mathcal{G} \cap \{\mathtt{Stop}(1), \mathtt{Stop}(a)\} = \emptyset$ and
if valuations $V_1$ and $V_2$ satisfy $Q_1$ and $Q_2$ on $\mathsf{red}_1(\mathcal{G})$ and $\mathsf{red}_a(\mathcal{G})$, respectively,
then valuation $W_{a,b} \stackrel{\text{def}}{=} V_1 \cup V_2 \cup \{\alpha_1 \mapsto 1, \alpha_2 \mapsto a, \tau \mapsto b\}$ satisfies $Q$ on $\mathcal{G}$.

Moreover, for $a = 1$, valuation $W_{a,b}$ refers only to facts that node $c$ (if $b = 1$) or node $s$ (if $b = 2$) is responsible for.

Properties (P1) and (P2) follow immediately from the construction of $Q$, whose body contains the extended bodies of $Q_1$ and $Q_2$, and the definition of extended and reduced instances. With these observations, it is not hard to prove the implications.

The general idea underlying the construction of policy $\mathbb{P}$—and the choice of its network in particular—is that node $c$ provides a counterexample for parallel completeness and node $s$ provides a counterexample for parallel soundness whenever $Q_1 \not\sqsubseteq Q_2$ holds.

⋙ Only if. Assume that containment holds, $Q_1 \sqsubseteq Q_2$. The following arguments prove that query $Q$ is then both parallel-complete and parallel-sound under policy $\mathbb{P}$. To this end, let $\mathcal{G}$ be an arbitrary instance over the schema and domain of $\mathbb{P}$ and let $\mathbb{L} \stackrel{\text{def}}{=} \mathbb{P} \triangleright \mathcal{G}$ be the induced distributed database.

» Parallel completeness. Let $V$ be an arbitrary valuation that satisfies $Q$ on $\mathcal{G}$. If $V$ is trivial, then it is also locally satisfying by Property (P0). Thus, let $V$ be non-trivial in the following.

By Property (P1), valuation $V$ satisfies query $Q_1$ on the reduced instance $\mathcal{I}_1 \stackrel{\text{def}}{=} \mathsf{red}_1(\mathcal{G})$. Containment $Q_1 \sqsubseteq Q_2$ additionally implies existence of a valuation $V_2$ that satisfies $Q_2$ on $\mathcal{I}_1$. Non-triviality of $V$ furthermore implies that $V(\alpha_1) = 1$ and therefore instance $\mathcal{G}$ contains facts $\texttt{Start}_1(1)$, $\texttt{Start}_2(1)$ and $\texttt{Type}(b)$, for some $b \stackrel{\text{def}}{=} V(\tau) \in \{1, 2\}$. Also, $\mathcal{G}$ does not contain $\texttt{Stop}(1)$ because $V$ satisfies $Q$.

By Property (P2), a valuation $W_{1,b}$ (induced by $V$, $V_2$ and $b$) that satisfies $Q$ on $\mathcal{G}$ is guaranteed to exist such that it only refers to facts that either node $c$ or node $s$ are responsible for. Hence, $W_{1,b}$ satisfies $Q$ locally on $\mathbb{L}(c)$ or $\mathbb{L}(s)$.

Therefore, query $Q$ is parallel-complete under policy $\mathbb{P}$.

» Parallel soundness. Let $V$ be an arbitrary valuation that satisfies $Q$ on the local instance $\mathbb{L}(k)$ of some node $k \in \mathsf{net}(\mathbb{P})$. If $V$ is trivial, then it is also globally satisfying by Property (P0). Thus, again, assume $V$ to be non-trivial in the following.

Satisfaction of $Q$ implies that the local instance $\mathbb{L}(k)$ contains facts $\texttt{Start}_1(1)$, $\texttt{Start}_2(1)$ and $\texttt{Type}(b)$, for $b \stackrel{\text{def}}{=} V(\tau)$, but not $\texttt{Stop}(1)$. This also holds for the global instance $\mathcal{G}$ since $\mathbb{P}$ is responsible for these facts on the relevant node (either node $c$, if $b = 1$, or node $s$, if $b = 2$).

By Property (P1), valuation $V$ satisfies $Q_1$ on the reduced instance $\mathcal{I}_1 \stackrel{\text{def}}{=} \mathsf{red}_1(\mathbb{L}(k))$ and containment $Q_1 \sqsubseteq Q_2$, once more, implies existence of valuation $V_2$ that satisfies $Q_2$ on $\mathcal{I}_1$.

Again, due to Property (P2), there is a valuation $W_{1,b}$ that satisfies $Q$ on $\mathcal{G}$ globally. Therefore, query $Q$ is parallel-sound under policy $\mathbb{P}$.

⋙ If. For a proof by contraposition, assume that containment $Q_1 \sqsubseteq Q_2$ does *not* hold. It is well-known that there exists a 'small' instance witnessing $Q_1 \not\sqsubseteq Q_2$. More

precisely, a minimal instance with at most as many data values as query $Q_1$ has variables is guaranteed to exist (Fact 2.1.3). Indeed, these data values can be assumed to be a subset of the universe $\mathsf{univ}(\mathbb{P}) = \{1, \ldots, m\}$ because both queries are generic and $|\mathsf{var}(Q_1)| \leq m$. In the following, let $\mathcal{I}_1$ be such an instance—over the schema and domain of policy $\mathbb{P}$ such that $Q_1(\mathcal{I}_1) \not\subseteq Q_2(\mathcal{I}_1)$.

The choice of instance $\mathcal{I}_1$ implies that there is a valuation $V_1$ that satisfies $Q_1$ on $\mathcal{I}_1$ although there is no valuation that satisfies $Q_2$ on $\mathcal{I}_1$. There is however some satisfying valuation $V_2$ for $Q_2$ over $\mathsf{univ}(\mathbb{P})$ because query $Q_2$ is consistent and contains no more than $m$ variables. This valuation is naturally satisfying for $Q_2$ on the instance $\mathcal{I}_2 \stackrel{\text{def}}{=} V(\mathsf{pos}(Q_2))$ that is induced by the facts it requires.

The following arguments show that query $Q$ is neither parallel-complete nor parallel-sound under policy $\mathbb{P}$.

» Parallel completeness. A global instance that witnesses that query $Q$ is *not* parallel-complete under policy $\mathbb{P}$ is

$$\mathcal{G} \stackrel{\text{def}}{=} \mathsf{ext}_1(\mathcal{I}_1) \cup \mathsf{ext}_2(\mathcal{I}_2) \\ \cup \{\mathtt{Start}_1(1), \mathtt{Start}_2(1), \mathtt{Start}_2(2), \mathtt{Type}(1)\},$$

derived from $\mathcal{I}_1$ and $\mathcal{I}_2$, containing some additional $\mathcal{T}$-atoms that enforce the violation of parallel completeness.

First, by Property (P2), valuation $W_{2,1}$ satisfies $Q$ on $\mathcal{G}$, globally.

Second, there is no locally satisfying valuation for $Q$. Assume, for the sake of a contradiction, that such a valuation $V$ exists. Since $\mathtt{Start}_2(1)$ and $\mathtt{Start}_2(2)$ are the only $\mathtt{Start}_2$-facts in $\mathcal{G}$, either $V(\alpha_2) = 1$ or $V(\alpha_2) = 2$ holds. Furthermore, $\mathtt{Type}(1)$ is the only $\mathtt{Type}$-fact in $\mathcal{G}$. It follows from the definition of $\mathbb{P}$ that only node $c$ is responsible for both facts $V(\mathtt{Start}_2(\alpha_2))$ and $\mathtt{Type}(1)$. In particular, $V(\alpha_2) = 1$ holds because node $c$ is *not* responsible for fact $\mathtt{Start}_2(2)$. Then, by Property (P1), valuation $V$ satisfies $Q_2$ on $\mathsf{red}_1(\mathcal{G})$. Since $\mathsf{red}_1(\mathcal{G})$ equals instance $\mathcal{I}_1$ with respect to $\mathcal{S}$-facts and, furthermore, $\mathcal{T}$-facts are irrelevant for $Q_2$, this implies that $Q_2$ is satisfied on $\mathcal{I}_1$. This contradicts the choice of $\mathcal{I}_1$ and thus concludes the argument.

Therefore, query $Q$ is *not* parallel-complete under policy $\mathbb{P}$.

» Parallel soundness. Similar to the previous argument, another global instance witnesses that query $Q$ is *not* parallel-sound under policy $\mathbb{P}$. This instance is

$$\mathcal{G} \stackrel{\text{def}}{=} \mathsf{ext}_1(\mathcal{I}_1) \cup \mathsf{ext}_2(\mathcal{I}_2) \\ \cup \{\mathtt{Start}_1(1), \mathtt{Start}_2(1), \mathtt{Start}_2(2), \mathtt{Type}(2), \mathtt{Stop}(2)\},$$

which differs from the global instance above only with respect to the $\mathtt{Type}$-facts and the additional $\mathtt{Stop}$-fact.

First, let valuation $W$ be induced by valuations $V_1$ and $V_2$ as valuation $W_{2,2}$ in Property (P2). Clearly, valuation $W$ satisfies $Q$ on local instance $\mathbb{L}(s)$ because

it requires $\mathcal{T}$-facts $\texttt{Start}_2(2)$ and $\texttt{Type}(2)$, for which node $s$ is responsible, and prohibits fact $\texttt{Stop}(2)$, for which this node is *not* responsible.

Second, there is no globally satisfying valuation for $Q$ on $\mathcal{G}$. Towards a contradiction, assume that such a valuation $V$ exists. Then, either $V(\alpha_2) = 1$ or $V(\alpha_2) = 2$ because $\texttt{Start}_2(1)$ and $\texttt{Start}_2(2)$ are the only $\texttt{Start}_2$-facts in $\mathcal{G}$. The latter option however cannot hold since this would render the valuation inconsistent. Hence, $V(\alpha_2) = 1$ and, by Property (P1), valuation $V$ satisfies query $Q_2$ on $\mathsf{red}_1(\mathcal{G}) = \mathcal{I}_1$. Again, this contradicts the choice of $\mathcal{I}_1$ and concludes the argument.

Therefore, query $Q$ is *not* parallel-sound under policy $\mathbb{P}$.

This completes the proof, showing that containment holds if and only if query $Q$ is parallel-complete, parallel-sound and parallel-correct under policy $\mathbb{P}$. □

The relatively high complexity of parallel correctness and its variants has motivated the study of fragments with better worst-case bounds. Since the lower bounds in Proposition 3.3.6 already hold for the simple rule-based representations of policies, restrictions on the query classes are a natural target.

Following established patterns in the literature, we consider *full* queries in Section 3.3.2. Afterwards, we turn to *polarised* queries, with a restricted form of negation, in Section 3.3.3.

### 3.3.2. Full queries

In Section 3.2.2, we have discussed that the complexity of parallel completeness decreases from $\Pi_2^p$ to coNP for conjunctive queries (without negation) that are strongly minimal. For these queries, every valuation is guaranteed to be minimally requiring. Full queries, that is, queries without existentially quantified variables, form an important special case of strongly minimal queries—often considered in other contexts in the literature too.

Interestingly, fullness is such a strong restriction that allowing negation in queries does not change the complexity of parallel completeness. At least this holds when policies can be evaluated without nondeterminism. Parallel soundness, however, is not trivial anymore (but has the same worst-case complexity as parallel completeness).

**Proposition 3.3.7.** The following problems are coNP-hard, even if the input is restricted to complete policies over only two nodes.

- $\textsc{PSound}(\mathsf{UCQ}^{\neg[\mathsf{full}]}, \mathcal{P}_{\mathrm{rule}})$
- $\textsc{PComp}(\mathsf{UCQ}^{\neg[\mathsf{full}]}, \mathcal{P}_{\mathrm{rule}})$
- $\textsc{PC}(\mathsf{UCQ}^{\neg[\mathsf{full}]}, \mathcal{P}_{\mathrm{rule}})$

*Proof.* We provide polynomial reductions to all three variants. The reductions start from Contain($\mathsf{CQ}^\neg_{[\mathsf{full}]}, \mathsf{UCQ}^\neg_{[\mathsf{full}]}$), which is coNP-complete [GKNS19]. The input $(Q_1, Q_2)$ for the containment problem consists of a (single) conjunctive query $Q_1$ with negation and a union $Q_2$ of conjunctive queries with negation. It is mapped to a single query $Q$ and a policy $\mathbb{P} \in \mathcal{P}_{\text{rule}}$. Assume, without loss of generality, that both input queries are formulated over a schema $\mathcal{S}$ that does not contain relation symbol `Global`. Furthermore, let $s$ denote the number of variables in query $Q_1$. While the derived query $Q$ differs, all reductions use the same non-skipping policy

$$\mathbb{P} \stackrel{\text{def}}{=} \{k\} \times \mathsf{facts}(\mathcal{S}, \mathsf{dom}_s) \cup \{\ell\} \times \{\texttt{Global}\}.$$

This policy is defined over a schema that is extended by a nullary relation `Global`. The purpose of the policy is to forward all facts relevant for the original queries to node $k$ and to separate them from `Global`-facts. Such a fact serves as a flag to 'activate' or 'deactivate' the derivation of facts based on an idea explained below. Policy $\mathbb{P}$ can be described by a linear number of rules—linear in the size of the schema, and thus in the size of the queries. Since all rules, except for $\texttt{Global} \to \ell$, are of the form $R(x_1, \ldots, x_k) \to k$, the policy can be computed in polynomial time.

For a query $Q \in \mathsf{CQ}^\neg$, let $Q^+$ and $Q^-$ denote the query resulting from addition of relation atom `Global` to $\mathsf{pos}(Q)$ and $\mathsf{neg}(Q)$, respectively. For a query $Q \in \mathsf{UCQ}^\neg$, the atom is added to each subquery in the respective fashion. Then, if the original query does not refer to relation `Global`, the modified query either behaves like the original one or like a trivial one, depending on the instance. More precisely, the modified queries satisfy the following equalities.

$$Q^+(\mathcal{G} \cup \{\texttt{Global}\}) = Q(\mathcal{G}) \tag{3.4}$$
$$Q^-(\mathcal{G} \cup \{\texttt{Global}\}) = \emptyset \tag{3.5}$$
$$Q^+(\mathcal{G} - \{\texttt{Global}\}) = \emptyset \tag{3.6}$$
$$Q^-(\mathcal{G} - \{\texttt{Global}\}) = Q(\mathcal{G}) \tag{3.7}$$

We start with parallel soundness. After that, we consider parallel completeness and parallel correctness, using the same reduction for both. In each case, query $Q$ is clearly computable from input queries $Q_1$ and $Q_2$ in polynomial time and the mapping is total. Hence, after each definition, we argue only the satisfaction of the reduction property.

⋙ Hardness of PSound($\mathsf{UCQ}^\neg_{[\mathsf{full}],\mathcal{P}_{\mathsf{rule}}}$). Let the derived query be $Q \stackrel{\text{def}}{=} Q_1^- \cup Q_2^+$. Query $Q$ is parallel-sound under policy $\mathbb{P}$ if and only if containment $Q_1 \sqsubseteq Q_2$ holds.

» If. For a direct proof, assume that $Q_1 \sqsubseteq Q_2$ holds and let $\mathcal{G}$ be an arbitrary instance over $\mathbb{P}$. On the one hand, for the local instance $\mathbb{L}(k) \stackrel{\text{def}}{=} (\mathbb{P} \triangleright \mathcal{G})(k) =$

$\mathcal{G} - \{\texttt{Global}\}$, we have

$$Q(\mathbb{L}(k)) = (Q_1^- \cup Q_2^+)(\mathbb{L}(k)) = Q_1^-(\mathbb{L}(k)) \cup \emptyset = Q_1(\mathbb{L}(k)) = Q_1(\mathcal{G})$$

by Equations (3.6) and (3.7) and because $Q_1$ ignores fact `Global`. On the other hand, the global result set depends on the occurrence of fact `Global`.

› 1. Case ($\texttt{Global} \in \mathcal{G}$). Then, $Q(\mathcal{G}) = (Q_1^- \cup Q_2^+)(\mathcal{G}) = \emptyset \cup Q_2^+(\mathcal{G}) = Q_2(\mathcal{G})$ by Equations (3.4) and (3.5). The containment relationship between the input queries furthermore ensures $Q_2(\mathcal{G}) \supseteq Q_1(\mathcal{G}) = Q(\mathbb{L}(k))$.

› 2. Case ($\texttt{Global} \notin \mathcal{G}$). Now, $Q(\mathcal{G}) = (Q_1^- \cup Q_2^+)(\mathcal{G}) = \emptyset \cup Q_1^-(\mathcal{G}) = Q_1(\mathcal{G})$ by Equations (3.6) and (3.7). As argued above, $Q_1(\mathcal{G}) = Q(\mathbb{L}(k))$ holds.

Therefore, query $Q$ is parallel-sound under $\mathbb{P}$ in both cases.

» Only if. For a proof by contraposition, assume now that $Q_1 \not\sqsubseteq Q_2$. Fact 2.1.3 guarantees existence of an instance $\mathcal{G}$ with at most $s$ data values such that $Q_1(\mathcal{G}) \not\subseteq Q_2(\mathcal{G})$. Since both queries are free of constants and thus generic, we can assume without restriction that $\mathcal{G}$ is an instance over $\mathsf{dom}_s$, the universe of policy $\mathbb{P}$. Even more, we may assume that $\mathcal{G}$ contains fact `Global`, since it is ignored by queries $Q_1$ and $Q_2$. Just like above, we can conclude $Q(\mathbb{L}(k)) = Q_1(\mathcal{G})$ and, as in the first case, $Q(\mathcal{G}) = Q_2(\mathcal{G})$. This implies $Q(\mathbb{L}(k)) \not\subseteq Q(\mathcal{G})$ and thus witnesses that query $Q$ is *not* parallel-sound under $\mathbb{P}$, as desired.

»» Hardness of $\text{PCOMP}(\text{UCQ}^{\neg[\text{full}],\mathcal{P}_{\text{rule}}})$. In principle, a reduction 'symmetric' to the previous one, that is, $Q \stackrel{\text{def}}{=} Q_1^+ \cup Q_2^-$, with an analogous argument would prove the stated hardness result. Instead, we consider query $Q \stackrel{\text{def}}{=} Q_1^+ \cup Q_2$, which does *not* introduce negation and therefore allows us to draw stronger conclusions (cf. Corollary 3.3.8 below). Query $Q$ is parallel-complete under policy $\mathbb{P}$ if and only if containment $Q_1 \sqsubseteq Q_2$ holds.

Clearly, on the local instance of node $k$, which never contains fact `Global`, the derived query is equivalent to $Q_2$. More precisely, $Q((\mathbb{P} \triangleright \mathcal{G})(k)) = Q_2(\mathcal{G})$ holds for every global instance $\mathcal{G}$.

» If. Assume that $Q_1 \sqsubseteq Q_2$ holds. This implies $Q_1^+ \sqsubseteq Q_1 \sqsubseteq Q_2$, which gives the equivalence $Q = Q_1^+ \cup Q_2 \equiv Q_2$. In particular, the global result set $Q(\mathcal{G}) = Q_2(\mathcal{G})$ is identical to the local result set $Q((\mathbb{P} \triangleright \mathcal{G})(k))$. Therefore, query $Q$ is parallel-complete under policy $\mathbb{P}$.

» Only if. Assume $Q_1 \not\sqsubseteq Q_2$ towards a proof by contraposition. As in the argument for the first reduction, this implies the existence of an instance $\mathcal{G}$ over $\mathbb{P}$ that contains fact `Global` and witnesses non-containment, that is, $Q_1(\mathcal{G}) \not\subseteq Q_2(\mathcal{G})$. The global result set is then $Q(\mathcal{G}) = Q_1^+(\mathcal{G}) \cup Q_2(\mathcal{G})$ by Equation (3.4) and hence a strict superset of the local result set $Q_2(\mathcal{G})$. Therefore, query $Q$ is *not* parallel-correct under policy $\mathbb{P}$.

⋙ Hardness of $\mathrm{PC}(\mathsf{UCQ}^{\neg[\mathsf{full}],\mathcal{P}_{\mathsf{rule}}})$. The previous reduction for parallel completeness is also a reduction for parallel correctness: query $Q$ is parallel-sound under $\mathbb{P}$ because the policy forwards all—in particular, all prohibited—facts over schema $\mathcal{S}$ to node $k$.

This completes the coNP-hardness proofs for all three variants. □

Remarkably, the reduction provided for parallel completeness neither introduces negation nor new variables in query $Q$. It only requires that the targeted query class is closed under, first, disjunction and, second, conjunction with atoms. Hence, deciding parallel completeness for such query classes is at least as hard as deciding the corresponding containment problem. Since this problem is undecidable for the relational algebra and for Datalog [SSS10], this implies the next result.

**Corollary* 3.3.8.** Parallel completeness and parallel correctness are undecidable for the relational algebra and for Datalog.

Coming back to conjunctive queries (with union and negation), the lower bounds stated in Proposition 3.3.7 are tight by the next proposition.

**Proposition* 3.3.9.** The following problems are in coNP.

- $\mathrm{PSound}(\mathsf{UCQ}^{\neg[\mathsf{full}],\mathcal{P}_{\mathrm{rule}}})$
- $\mathrm{PComp}(\mathsf{UCQ}^{\neg[\mathsf{full}],\mathcal{P}_{\mathrm{rule}}})$
- $\mathrm{PC}(\mathsf{UCQ}^{\neg[\mathsf{full}],\mathcal{P}_{\mathrm{rule}}})$

Indeed, a closer look into the proof of Proposition 3.3.9 reveals that the coNP-upper bound can be extended to all policy families in $\mathfrak{P}_{\mathrm{det}}$ [GKNS19].

As shown above, the absence of existential variables simplifies the reasoning process for parallel correctness. Intuitively, this is because, for full queries, each fact in the result set is derived via a *unique* valuation. This is not the case for polarised queries—which we consider next—, where the complexity accordingly increases again. Nevertheless, it does not raise to coNEXPTIME, as for conjunctive queries with negation in general.

### 3.3.3. Polarised queries

We have started our study of the parallel correctness problem and its variants for the non-monotonic class $\mathrm{UCQ}^{\neg,\neq}$ with the observation that certain queries only have exponential-size instances witnessing the violation of the respective property (Example 3.3.1).

This size bound depends on two aspects: first, on the arity of the relation symbols and, second, on the simultaneous occurrence of a positive and a negated atom

referring to the same relation symbol. Both aspects are reflected in the two $S$-atoms of the queries in the example sequence $Q_1, Q_2, Q_3, \ldots$ where

$$Q_n = H() \leftarrow R(x_1), \ldots, R(x_n), R(y_1), \ldots, R(y_n), S(x_1, \ldots, x_n), \neg S(y_1, \ldots, y_n)$$

for every $n \in \mathbb{N}$.

Theorem 3.3.4 states that, with an *a priori* bound on the arity, the complexity remains on the second level of the polynomial hierarchy (in $\Pi_2^p$). In the following, we demonstrate that this is also the case if negation is restricted to relations that do not occur positively—indicating that, indeed, *both* aspects are crucial for the coNEXPTIME-hardness result.

Intuitively, reasoning about parallel soundness is complicated for queries like $Q_n$ because of the following problem. Assume that there is a valuation $V$ and a node such that the node is responsible for all facts required by $V$ but not for all facts prohibited by $V$. This valuation is a candidate for a local derivation without a corresponding global derivation—but only if *itself* is not satisfying globally. That is, the global instance has to contain a prohibited fact, here: $g = V(S(y_1, \ldots, y_n))$. Addition of fact $g$ however may spawn another valuation $V'$ that derives the same fact as $V$ and does *not prohibit* but *require* fact $g = V'(S(x_1, \ldots, x_n))$. If this valuation is to be unsatisfying, the global instance thus has to contain another fact $g' = V'(S(y_1, \ldots, y_n))$. This process then may have to be repeated and is bounded in the worst case only by the number of facts representable over the considered schema.

An obvious way to prevent the repetitive steps completely is by establishing a syntactic restriction on the queries as defined next.

**Definition 3.3.10 (Polarised query).** A query $Q \in \mathsf{UCQ}^{\neg,\neq}$ is *polarised*[3] if there is a bipartition $\mathcal{S}^+ \uplus \mathcal{S}^-$ of its schema such that every relation symbol from $\mathcal{S}^+$ occurs only positively and every relation symbol from $\mathcal{S}^-$ occurs only negated in $Q$. ◀

Let $\mathcal{S}^+(Q)$ and $\mathcal{S}^-(Q)$ denote the respective (disjoint) subschemas. If the query is clear from the context, we usually simply write $\mathcal{S}^+$ and $\mathcal{S}^-$.

Note, in particular, that for unions of conjunctive queries, the definition demands from every disjunct to adhere to the same division of relation symbols. That is, query $Q = Q_1 \cup Q_2$ with subqueries $Q_1 = H() \leftarrow R(x), \neg S(x)$ and $Q_2 = H() \leftarrow S(y), \neg T(y)$ is *not* polarised although both its disjuncts are.

In the following, we study conditions and provide characterisations of both parallel completeness and parallel soundness for polarised queries. Eventually, we complete the picture by showing that both problems are $\Pi_2^p$-complete.

[3] The name is inspired by the name of a graph concept introduced in the context of the containment problem for conjunctive queries with negation [LM07]. There, the queries that we call 'polarised' correspond to '*pure* polarised' graphs.

### Characterisation

For conjunctive queries *without negation*, Condition (PComp) provides a characterisation of parallel completeness in terms of minimally requiring valuations. An obvious question is whether this notion of minimality, which simply ignores prohibited facts, is also suitable for conjunctive queries *with negation*. Unsurprisingly, this is not the case, as the following example demonstrates.

**Example 3.3.11.** Let $Q$ be the polarised query

$$H(x,y) \leftarrow R(x,x), R(x,y), R(u,v), \neg T(u), \neg T(v)$$

and let

$$\begin{aligned}\mathbb{P} \stackrel{\text{def}}{=} &\{k_1\} \times \{R(1,1), R(1,2), T(1), T(2)\} \\ &\cup \{k_2\} \times \{R(2,1), R(2,2), T(1), T(2)\}\end{aligned}$$

be a non-skipping policy with universe $\{1,2\}$. This policy $\mathbb{P}$ has, for each minimally requiring valuation, a node that is responsible for the required facts. However, query $Q$ is not parallel-complete under $\mathbb{P}$, as argued in the following.

First, minimally requiring valuations that can derive $H(1,1)$ or $H(2,2)$ solely require fact $R(1,1)$ or $R(2,2)$, which can be found on node $k_1$ or $k_2$, respectively. Furthermore, minimally requiring valuations that derive $H(1,2)$ require only facts $R(1,1)$ and $R(1,2)$, which can be found on node $k_1$. The argument for minimally requiring valuations that derive $H(2,1)$ is analogous (with node $k_2$).

Second, query $Q$ is not parallel-complete under policy $\mathbb{P}$ on instance

$$\mathcal{G} \stackrel{\text{def}}{=} \{R(1,1), R(1,2), R(2,2), T(1)\},$$

as witnessed by fact $H(1,2)$, which is derived globally but not locally. Valuation $V = \{x \mapsto 1, y \mapsto 2, u \mapsto 2, v \mapsto 2\}$ requires all three $R$-facts in $\mathcal{G}$ and prohibits only $T(2)$, which is not present in $\mathcal{G}$. Hence $V$ derives $H(1,2)$ for query $Q$ on the global instance $\mathcal{G}$. However, there is no locally satisfying valuation that derives the same fact, as the following argument by contradiction shows. Assume that there is a valuation $V$ that derives $H(1,2)$ on node $k_1$ or on node $k_2$. Since both nodes are responsible for the prohibited *and present* fact $T(1)$, valuation $V$ maps $u$ and $v$ to 2 and thus requires fact $R(2,2)$. Since the valuation derives $H(1,2)$, it additionally requires fact $R(1,1)$. Obviously, neither node $k_1$ nor node $k_2$ is responsible for both these facts, contradicting the assumption. Therefore, query $Q$ is not parallel-complete under policy $\mathbb{P}$. ■

The negative outcome of the previous example motivates the quest for a different notion of minimality for valuations, and hence for a different preorder on valuations.

We extend Definition 3.2.1 of preorder $\leq_Q^{\text{pos}}$ by a third condition. This condition takes into account also the negated atoms of the query or, more precisely, the facts prohibited by the valuations.

**Definition 3.3.12 (Preorder $\leq_Q$).** For a query $Q \in \mathsf{UCQ}^{\neg,\neq}$, the preorder $\leq_Q$ relates valuations for $Q$. If $V, V'$ are valuations for $Q$, then $V \leq_Q V'$ holds if

1. both valuations derive the same fact, $V(\mathsf{head}(Q)) = V'(\mathsf{head}(Q))$;
2. valuation $V$ requires only facts that $V'$ requires, $V(\mathsf{pos}(Q)) \subseteq V'(\mathsf{pos}(Q))$; and
3. valuation $V$ prohibits only facts that $V'$ prohibits, $V(\mathsf{neg}(Q)) \subseteq V'(\mathsf{neg}(Q))$. ◀

It is obvious that the extended definition agrees with the restricted definition on every query $Q \in \mathsf{UCQ}^{\neq}$, where $\mathsf{neg}(Q) = \emptyset$. This justifies the use of the same notation above.

As before, preorder $\leq_Q$ naturally induces an equivalence relation $\equiv_Q$ and a strict preorder $<_Q$. In particular, a valuation can be strictly smaller for one of two reasons, fewer required or fewer prohibited facts. More precisely, $V_1 <_Q V_2$ holds

- if $V_1 \leq_Q V_2$ and $V_1(\mathsf{pos}(Q)) \subsetneq V_2(\mathsf{pos}(Q))$ holds; or
- if $V_1 \leq_Q V_2$ and $V_1(\mathsf{neg}(Q)) \subsetneq V_2(\mathsf{neg}(Q))$ holds.

Again, this preorder induces a notion of minimality.

**Definition 3.3.13 (Minimal valuation).** For a fixed query $Q \in \mathsf{UCQ}^{\neg,\neq}$, a valuation is *minimal* if it is minimal with respect to $\leq_Q$, that is, if there is no valuation $U$ for $Q$ such that $U <_Q V$. ◀

As before, every fact $f$ that is derived by an *arbitrary* valuation on some instance $\mathcal{I}$, is also derived by a *minimal* valuation on $\mathcal{I}$. Testing minimality remains coNP-complete: The upper bound is easily extended and hardness already holds for CQs (cf. Section 3.2.1).

Note that the set of required facts and the set of prohibited facts can behave in an antagonistic fashion, as the following example indicates. In particular, for a given polarised query with negated atoms, a minimally requiring valuation is not necessarily minimal and vice versa.

**Example 3.3.14.** Consider again the polarised query from Example 3.3.11.

$$Q = H(x, y) \leftarrow R(x, x), R(x, y), R(u, v), \neg T(u), \neg T(v)$$

For data values $a$ and $b$, the mapping $V_{a,b} \stackrel{\text{def}}{=} \{x \mapsto 1, y \mapsto 2, u \mapsto a, v \mapsto b\}$ defines a valuation that can, if satisfying, derive fact $H(1, 2)$ for $Q$.

Valuation $V_{1,2}$ is *not* minimal because there is a valuation $V_{1,1}$ that requires the same set $V_{1,1}(\mathsf{pos}(Q)) = \{R(1, 1), R(1, 2)\} = V_{1,2}(\mathsf{pos}(Q))$ of facts while it prohibits strictly less facts: $V_{1,1}(\mathsf{neg}(Q)) = \{T(1)\} \subsetneq \{T(1), T(2)\} = V_{1,2}(\mathsf{neg}(Q))$. Moreover,

valuation $V_{1,1}$ *is* minimal because it prohibits only one fact and requires only facts that are necessarily required for the derivation of $H(1,2)$. However, both valuations are minimally requiring.

A third valuation, namely $V_{2,2}$, is also minimal for $Q$ although it requires strictly more facts than $V_{1,1}$. Indeed, it requires facts $V_{2,2}(\mathsf{pos}(Q)) = \{R(1,1), R(1,2), R(2,2)\}$ and prohibits $V_{2,2}(\mathsf{neg}(Q)) = \{T(2)\}$. This valuation is minimal because it only prohibits a single fact that is different from $T(1)$ and every such valuation requires three facts because variables $u$ and $x$ have to be mapped differently. However, it is not minimally requiring (because $V_{1,1} <^{\text{pos}} V_{2,2}$). ■

Interestingly, this—canonical—generalisation of minimality to valuations for queries with negation is adequate to formulate a characterisation for parallel *soundness* but not for parallel *completeness*. We turn to the positive result first and consider the negative result thereafter.

**Condition (PSound-pol)**

**Assumptions: Let $Q$ be a polarised $\mathsf{UCQ}^{\neg,\neq}$ and let $\mathbb{P}$ be a policy.**
For every minimal valuation $V$ for $Q$ over $\mathbb{P}$ and every node $k \in \mathsf{net}(\mathbb{P})$
that is responsible for the required facts, $V(\mathsf{pos}(Q)) \subseteq \mathbb{P}(k)$,
this node is also responsible for the prohibited facts, $V(\mathsf{neg}(Q)) \subseteq \mathbb{P}(k)$.

This condition yields a characterisation of parallel soundness for polarised queries.

**Proposition 3.3.15 (Characterisation of parallel soundness).** Let $Q$ be an arbitrary polarised query from $\mathsf{UCQ}^{\neg,\neq}$ and let $\mathbb{P}$ be an arbitrary policy. Query $Q$ is parallel-sound under $\mathbb{P}$ if and only if Condition (PSound-pol) is satisfied for $Q$ and $\mathbb{P}$.

*Proof.* We prove both implications of the equivalence separately.

⋙ Only if. Assume that $Q$ is parallel-sound under $\mathbb{P}$. We show that $Q$ and $\mathbb{P}$ satisfy Condition (PSound-pol). Let $V$ be an arbitrary minimal valuation for $Q$ and let $k \in \mathsf{net}(\mathbb{P})$ be an arbitrary node that is responsible for all facts required by $V$. If there is no such node, then valuation $V$ is trivially conform with Condition (PSound-pol).

It suffices to show that node $k$ is responsible for all facts prohibited by the valuation, $V(\mathsf{neg}(Q)) \subseteq \mathbb{P}(k)$. Towards a proof by contradiction, let us assume that this is not the case. Then, $\mathcal{F}^- \stackrel{\text{def}}{=} V(\mathsf{neg}(Q)) \cap \mathbb{P}(k)$ is a *strict* subset of $V(\mathsf{neg}(Q))$. Now, we define a global instance $\mathcal{G} \stackrel{\text{def}}{=} \mathcal{G}^+ \cup \mathcal{G}^-$ where $\mathcal{G}^+ \stackrel{\text{def}}{=} V(\mathsf{pos}(Q))$ and $\mathcal{G}^- \stackrel{\text{def}}{=} \mathsf{facts}(\mathcal{S}^-, \mathsf{univ}(\mathbb{P})) - \mathcal{F}^-$.

Obviously, valuation $V$ satisfies query $Q$ on the local instance $(\mathbb{P} \triangleright \mathcal{G})(k)$ of node $k$. Furthermore, since $Q$ is parallel-sound under $\mathbb{P}$ by assumption, there is a valuation $W$ that derives the same fact as $V$ and satisfies $Q$ globally, on $\mathcal{G}$. This implies that

$W(\mathsf{pos}(Q)) \subseteq \mathcal{G}^+ = V(\mathsf{pos}(Q))$ and $W(\mathsf{neg}(Q)) \cap \mathcal{G}^- = \emptyset$ because query $Q$ is polarised. The latter furthermore implies $W(\mathsf{neg}(Q)) \subseteq \mathcal{F}^- = V(\mathsf{neg}(Q)) \cap \mathbb{P}(k) \subsetneq V(\mathsf{neg}(Q))$ and thus contradicts the assumed minimality of valuation $V$.

Therefore, Condition (PSound-pol) holds if query $Q$ is parallel-sound under policy $\mathbb{P}$.

⋙ If. Assume that Condition (PSound-pol) is satisfied for query $Q$ and policy $\mathbb{P}$. We show that $Q$ is parallel-sound under $\mathbb{P}$. To this end, let $\mathcal{G}$ be an arbitrary instance over the domain of $\mathbb{P}$ and let $V$ be a valuation that satisfies $Q$ on the local instance $(\mathbb{P} \triangleright \mathcal{G})(k)$ of some node $k$.

Then, there exists a minimal valuation $V^*$ such that $V^* \leq V$. In particular, this valuation satisfies $V^*(\mathsf{pos}(Q)) \subseteq V(\mathsf{pos}(Q)) \subseteq (\mathbb{P} \triangleright \mathcal{G})(k) \subseteq \mathcal{G}$ and $V^*(\mathsf{neg}(Q)) \cap (\mathbb{P} \triangleright \mathcal{G})(k) \subseteq V(\mathsf{neg}(Q)) \cap (\mathbb{P} \triangleright \mathcal{G})(k) = \emptyset$. Indeed, the latter implies $V^*(\mathsf{neg}(Q)) \cap \mathcal{G} = \emptyset$ because $V^*(\mathsf{neg}(Q)) \subseteq \mathbb{P}(k)$ by Condition (PSound-pol). Thus, valuation $V^*$ satisfies $Q$ on the global instance and witnesses that the derivation of fact $V(\mathsf{head}(Q))$ is correct.

Therefore, query $Q$ is parallel-sound under policy $\mathbb{P}$ if Condition (PSound-pol) holds.

Hence, query $Q$ is parallel-sound under policy $\mathbb{P}$ if and only if Condition (PSound-pol) is satisfied. □

Although we have just shown that the notion of minimality that is based on the generalised preorder (Definition 3.3.12) is appropriate for a characterisation of parallel soundness, it is unfortunately *not* appropriate for a characterisation of parallel completeness.

The reason for this is that—depending on the policy—not necessarily all valuations that are minimal *in this sense* are relevant. The next example illustrates that.

**Example 3.3.16.** Consider the polarised query

$$Q \stackrel{\text{def}}{=} H(x, y) \leftarrow R(x, y), R(u, v), \neg S(x), \neg S(v)$$

and policy $\mathbb{P} \stackrel{\text{def}}{=} (\{k\} \times \{R(1,1), R(2,2)\}) \cup (\{\ell\} \times \{R(1,2)\}) \cup (\{m\} \times \{R(2,1)\})$ with universe $\{1, 2\}$, which partitions all $R$-facts over three nodes and *skips all $S$-facts*. Query $Q$ is parallel-complete under this policy. However, it is not true that for every minimal valuation there is a node that is responsible for the required facts.

First, query $Q$ is parallel-complete under $\mathbb{P}$ because, for every valuation $V$ that derives a fact $H(a, b)$ globally, there is a valuation $V'$ that agrees with $V$ on the head variables $x, y$ and maps $V'(u) = V(x)$ and $V'(v) = V(y)$. Valuation $V'$ requires only the single fact $R(a, b)$ that is also required by $V$ and can be found in one of the local instances by definition of $\mathbb{P}$. Furthermore, valuation $V'$ satisfies $Q$ on the

corresponding node because the local instance does not contain any (prohibited) $S$-fact.

Second, valuation $W^* \stackrel{\text{def}}{=} \{x \mapsto 1, y \mapsto 2, u \mapsto 2, v \mapsto 1\}$, which requires facts $R(1,2)$ and $R(2,1)$ and prohibits fact $S(1)$, is minimal since every valuation that derives $H(1,2)$ and prohibits $S(1)$ requires two facts. However, no node in the network of $\mathbb{P}$ is responsible for both required facts. ■

Note that, in the example above, it is not essential that policy $\mathbb{P}$ is skipping. Alternatively, there could be a fourth node responsible for both $S$-facts, allowing to come to the same conclusion.

The notion of minimality based on preorder $\leq$ can thus be too fine to capture parallel completeness for some policies. Next, we adjust Definition 3.3.12 to the possible influence of the policy. The basic idea is to distinguish, for a fixed policy $\mathbb{P}$ and a fixed node $k$ in its network, the facts prohibited by a valuation $V$ for a query $Q$:

- the set of facts $V(\mathsf{neg}(Q)) \cap \mathbb{P}(k)$ that are *effectively prohibited* and
- the set of facts $V(\mathsf{neg}(Q)) - \mathbb{P}(k)$ that are *ineffectively prohibited*.

Ineffectively prohibited facts can be present in the global instance while the valuation remains satisfying on the local instance, to the possible benefit of parallel completeness. The following preorder extends the previous one in that respect—by Property (3b).

**Definition 3.3.17 ($\mathbb{P}$-relativised preorder).** For a query $Q \in \mathsf{UCQ}^{\neg,\neq}$ and a fixed policy $\mathbb{P}$, the preorder $\leq_Q^{\mathbb{P}}$ relates valuations for $Q$. If $V, V'$ are valuations for $Q$, then $V \leq_Q^{\mathbb{P}} V'$ holds if

1. both valuations derive the same fact,
   $V(\mathsf{head}(Q)) = V'(\mathsf{head}(Q))$;
2. valuation $V$ requires only facts that $V'$ requires,
   $V(\mathsf{pos}(Q)) \subseteq V'(\mathsf{pos}(Q))$; and,
3. one of the following properties is satisfied,
   a) valuation $V$ prohibits only facts that $V'$ prohibits,
      $V(\mathsf{neg}(Q)) \subseteq V'(\mathsf{neg}(Q))$; or
   b) there is a node $k \in \mathbb{P}^{-1}(V(\mathsf{pos}(Q)))$ such that
      - $V(\mathsf{neg}(Q)) \cap \mathbb{P}(k) \subseteq V'(\mathsf{neg}(Q))$ and
      - $V(\mathsf{neg}(Q)) \cap \mathbb{P}(k) \subseteq V'(\mathsf{neg}(Q)) \cap \mathbb{P}(\ell)$
        for every node $\ell \in \mathbb{P}^{-1}(V'(\mathsf{pos}(Q)))$. ◀

This relation is indeed a preorder, as shown in the appendix (Lemma A.2). Clearly, if $V \leq_Q V'$, then also $V \leq_Q^{\mathbb{P}} V'$. As usual, an equivalence relation and strict preorder can be derived from the $\mathbb{P}$-relativised preorder. In particular, $V <_Q^{\mathbb{P}} V'$ holds

- if $V <_Q V'$ or

- if $V \leq_Q^{\mathbb{P}} V'$ and
  - if $V(\mathsf{pos}(Q)) \subsetneq V'(\mathsf{pos}(Q))$ or
  - if there is a node $k \in \mathbb{P}^{-1}(V(\mathsf{pos}(Q)))$ such that
    - $V(\mathsf{neg}(Q)) \cap \mathbb{P}(k) \subsetneq V'(\mathsf{neg}(Q))$ and
    - $V(\mathsf{neg}(Q)) \cap \mathbb{P}(k) \subsetneq V'(\mathsf{neg}(Q)) \cap \mathbb{P}(\ell)$ for every $\ell \in \mathbb{P}^{-1}(V'(\mathsf{pos}(Q)))$.

Obviously, this definition is rather involved due to the interaction between valuations and the policy. Nevertheless, it reflects parallel completeness, as argued below. Furthermore, in some situations—with additional *a priori* knowledge about the policy (e.g., if it is skipping all facts over certain relations in $\mathcal{S}^-$)—it becomes simpler. In extreme cases, for instance if all $\mathcal{S}^-$-facts are skipped, it even becomes equivalent to $\leq^{\text{pos}}$. In any case, this preorder yields another notion of minimality.

**Definition 3.3.18 ($\mathbb{P}$-minimal valuation).** For a fixed query $Q \in \mathsf{UCQ}^{\neg,\neq}$ and a fixed policy $\mathbb{P}$, a valuation $V$ for $Q$ is $\mathbb{P}$-*minimal* for $Q$ if there is no valuation $U$ for $Q$ such that $U <_Q^{\mathbb{P}} V$. ◀

Both notions, minimality and $\mathbb{P}$-minimality, are incomparable. This is witnessed by the following two examples. The first example shows that the minimal valuation $W^*$ that is problematic in Example 3.3.16 is not $\mathbb{P}$-minimal.

**Example 3.3.19.** Valuation $W^* \stackrel{\text{def}}{=} \{x \mapsto 1, y \mapsto 2, u \mapsto 2, v \mapsto 1\}$, which requires facts $R(1,2)$ and $R(2,1)$ and prohibits the single fact $S(1)$ only, is minimal for query $Q \stackrel{\text{def}}{=} H(x,y) \leftarrow R(x,y), R(u,v), \neg S(x), \neg S(v)$, as we have argued in Example 3.3.16.

However, for the policy $\mathbb{P}$ considered there—which skips all $S$-facts—, valuation $W^*$ is not $\mathbb{P}$-minimal because $W <^{\mathbb{P}} W^*$ for valuation $W \stackrel{\text{def}}{=} \{x \mapsto 1, y \mapsto 2, u \mapsto 1, v \mapsto 2\}$. First, valuation $W$ requires fewer facts, namely only $R(1,2)$. Second, valuation $W$ effectively prohibits only facts effectively prohibited by $W^*$, that is,

$$W(\mathsf{neg}(Q)) \cap \mathbb{P}(\ell) = \{S(1), S(2)\} \cap \emptyset = \emptyset = \{S(1)\} \cap \emptyset = W^*(\mathsf{neg}(Q)) \cap \mathbb{P}(\ell)$$

holds, on the only node $\ell$ responsible for the required fact. ■

The second example demonstrates conversely that a $\mathbb{P}$-minimal valuation is not necessarily minimal.

**Example 3.3.20.** For the polarised query

$$H(x,y) \leftarrow R(x), R(y), R(u), R(v), \neg S(u), \neg S(v),$$

both valuations, $V_1$ and $V_2$, where $V_a \stackrel{\text{def}}{=} \{x \mapsto 1, y \mapsto 1, u \mapsto 1, v \mapsto a\}$ for $a \in \{1,2\}$, are $\mathbb{P}$-minimal given the $S$-fact skipping policy $\mathbb{P} \stackrel{\text{def}}{=} \{k\} \times \{R(1), R(2)\}$ with universe $\{1,2\}$. They are $\mathbb{P}$-minimal because they require the facts $R(1)$ and $R(2)$, necessary for the derivation of $H(1,2)$, and effectively prohibit no fact on node $k$.

Valuation $V_2$ is not minimal though because valuation $V_1$ prohibits fewer facts and thus satisfies $V_1 < V_2$. ■

Testing $\mathbb{P}$-minimality of a valuation is in coNP for policies from $\mathfrak{P}_{\text{det}}$ (as before, the universally quantified extra input is used to consider alternative valuations, and the properties from Definition 3.3.17 can be checked on the polynomial-size network in polynomial time).

Now that we have seen that neither *minimally requiring* valuations nor *minimal* valuations are suitable to characterise parallel completeness for polarised queries, we can finally provide a characterisation. This characterisation is, as to be expected, based on $\mathbb{P}$-minimal valuations.

**Condition (PComp-pol)**

**Assumptions: Let $Q$ be a polarised $\text{UCQ}^{\neg,\neq}$ and let $\mathbb{P}$ be a policy.**
For every $\mathbb{P}$-minimal valuation $V$ for $Q$ over $\mathbb{P}$,
there is a node in $\mathsf{net}(\mathbb{P})$ that is responsible for all facts required by the valuation.

**Proposition 3.3.21 (Characterisation of parallel completeness).** Let $Q$ be an arbitrary polarised query from $\text{UCQ}^{\neg,\neq}$ and let $\mathbb{P}$ be an arbitrary policy. Query $Q$ is parallel-complete under $\mathbb{P}$ if and only if Condition (PComp-pol) is satisfied for $Q$ and $\mathbb{P}$.

*Proof.* We prove both implications separately.

⋙ Only if. We assume that the polarised query $Q$ is parallel-complete under policy $\mathbb{P}$ and show that Condition (PComp-pol) holds. To this end, let $V$ be an arbitrary $\mathbb{P}$-minimal valuation for $Q$. Furthermore, let universe $U$ be the set of data values referred to by $V$.

From valuation $V$, we define the global instance $\mathcal{G} \stackrel{\text{def}}{=} \mathcal{G}^+ \uplus \mathcal{G}^-$ where $\mathcal{G}^+ \stackrel{\text{def}}{=} V(\mathsf{pos}(Q))$ consists of all facts required by $V$ and $\mathcal{G}^- \stackrel{\text{def}}{=} \mathsf{facts}(\mathcal{S}^-, U) - V(\mathsf{neg}(Q))$ is the complement of all facts prohibited by $V$. The global instance is designed such that valuation $V$ satisfies query $Q$ on $\mathcal{G}$. Parallel completeness of $Q$ under $\mathbb{P}$ thus guarantees the existence of a node $k \in \mathsf{net}(\mathbb{P})$ and of a valuation $W$ for $Q$ such that $W$ derives the same fact as $V$ on the local instance of node $k$. More precisely, the following three properties are satisfied: first, $W(\mathsf{head}(Q)) = V(\mathsf{head}(Q))$; second, $W(\mathsf{pos}(Q)) \subseteq (\mathbb{P} \triangleright \mathcal{G})(k)$; and third, $W(\mathsf{neg}(Q)) \cap (\mathbb{P} \triangleright \mathcal{G})(k) = \emptyset$.

The second property implies $W(\mathsf{pos}(Q)) \subseteq \mathcal{G}$ in particular and, because of polarisation, even $W(\mathsf{pos}(Q)) \subseteq \mathcal{G}^+ = V(\mathsf{pos}(Q))$. Obviously, valuation $V$ is conform with Condition (PComp-pol) if $W(\mathsf{pos}(Q)) = V(\mathsf{pos}(Q))$ or, independently, $\mathbb{P}^{-1}(V(\mathsf{pos}(Q))) \neq \emptyset$ holds. Next, we show that another case does not exist. Towards

a contradiction, assume that $W(\mathsf{pos}(Q)) \subsetneq V(\mathsf{pos}(Q))$ and $\mathbb{P}^{-1}(V(\mathsf{pos}(Q))) = \emptyset$. Since valuation $V$ is $\mathbb{P}$-minimal, and Properties (1) and (2) in Definition 3.3.17 are satisfied by $W$ and $V$, Property (3b) has to be violated. This particularly implies $W(\mathsf{neg}(Q)) \cap \mathbb{P}(k) \not\subseteq V(\mathsf{neg}(Q))$ because the second subproperty trivially holds since $\mathbb{P}^{-1}(V(\mathsf{pos}(Q))) = \emptyset$. Hence, valuation $W$ prohibits a fact that is present in $\mathbb{P}(k)$ and $\mathcal{G} \supseteq \mathcal{G}^-$, contradicting its choice. The arbitrarily chosen $\mathbb{P}$-minimal valuation $V$ hence satisfies the considered condition.

Therefore, Condition (PComp-pol) is satisfied for query $Q$ and policy $\mathbb{P}$.

⋙ If. We now assume that Condition (PComp-pol) is satisfied for $Q$ and $\mathbb{P}$ and prove that this implies that $Q$ is parallel-complete under $\mathbb{P}$. Let $\mathcal{G}$ be an arbitrary instance over $\mathbb{P}$ and let $V$ be a valuation that satisfies $Q$ on $\mathcal{G}$, deriving some fact $f$.

Let $V^*$ be a $\mathbb{P}$-minimal valuation such that $V^* \leq^{\mathbb{P}} V$. This implies $V^*(\mathsf{pos}(Q)) \subseteq V(\mathsf{pos}(Q)) \subseteq \mathcal{G}$. By Condition (PComp-pol), we can additionally conclude that there is a node $k \in \mathsf{net}(\mathbb{P})$ where $V^*(\mathsf{pos}(Q)) \subseteq \mathbb{P}(k)$ holds. Furthermore, $V^* \leq^{\mathbb{P}} V$ implies that valuations $V^*$ and $V$ satisfy Property (3a) or Property (3b) in Definition 3.3.17. In the former case, where $V^*(\mathsf{neg}(Q)) \subseteq V(\mathsf{neg}(Q))$ holds, valuation $V^*$ is clearly satisfying on node $k$ since valuation $V$ is satisfying on the global instance and thus no fact from $V(\mathsf{neg}(Q))$ is present in $\mathcal{G}$. Hence, valuation $V^*$ derives $f$ on node $k$. In the latter case, there is a node $\ell \in \mathbb{P}^{-1}(V^*(\mathsf{pos}(Q)))$ such that $V^*(\mathsf{neg}(Q)) \cap \mathbb{P}(\ell) \subseteq V(\mathsf{neg}(Q))$, that is, $V^*(\mathsf{neg}(Q)) \cap (\mathbb{P} \triangleright \mathcal{G})(\ell) = \emptyset$. Hence, valuation $V^*$ derives fact $f$ on node $\ell$.

Therefore, every globally derived fact is derived also locally, that is, query $Q$ is parallel-complete under policy $\mathbb{P}$.

Hence, query $Q$ is parallel-complete under $\mathbb{P}$ if and only if Condition (PComp-pol) is satisfied. □

This completes the picture on characterisations of parallel soundness and parallel completeness for polarised queries. A characterisation of parallel correctness trivially results as the conjunction of both Conditions (PSound-pol) and (PComp-pol).

### Complexity

The conditions established above allow to decide parallel correctness and its variants along the same lines as for negation-free conjunctive queries, as described in Section 3.2.1. Notably, without difference in the worst-case complexity (at least for 'deterministic' policies) and without restriction to queries over schemas that obey a fixed arity bound.

**Theorem 3.3.22.** For every $\mathcal{P} \in \mathfrak{P}_{\text{det}}$, the following problems are $\Pi_2^p$-complete for every query class $\mathcal{Q} \in \{\mathsf{CQ}^\neg[\mathsf{pol}], \mathsf{CQ}^{\neg,\neq}[\mathsf{pol}], \mathsf{UCQ}^\neg[\mathsf{pol}], \mathsf{UCQ}^{\neg,\neq}[\mathsf{pol}]\}$.

- $\textsc{PSound}(\mathcal{Q},\mathcal{P})$

- PComp$(\mathcal{Q},\mathcal{P})$
- PC$(\mathcal{Q},\mathcal{P})$

*Proof.* The lower bound for PComp and PC follows from the hardness result for parallel completeness for conjunctive queries, which already holds for policies from $\mathcal{P}_{\text{list}}$ (Proposition 3.2.13). For PSound, the lower bound is proven below, in Corollary 3.3.24.

The upper bounds follow by simple adaptations of algorithm $\mathbb{A}$ (page 30) such that now Conditions (PSound-pol) and (PComp-pol) are checked instead of Condition (PComp) for soundness and completeness, respectively. Clearly, parallel correctness can then be checked by successive application of both algorithms. □

The $\Pi_2^p$-hardness of PSound for *unions* of conjunctive queries with negation would be implied by a $\Pi_2^p$-hardness result for the containment problem and the reduction in the proof of Proposition 3.3.7. Indeed, hardness already holds for polarised queries *without disjunction*, as proven in the appendix (Proposition A.3).

**Proposition 3.3.23.** PSound$(\mathsf{CQ}^{\neg}_{\text{dom}},\mathcal{P}_{\text{list}})$, restricted to polarised queries, is $\Pi_2^p$-hard, even over a single-node network.

Just like for parallel completeness and negation-free queries, the lower bound can be achieved without the use of constants.

**Corollary 3.3.24.** PSound$(\mathsf{CQ}^{\neg}_{[\text{pol}]},\mathcal{P}_{\text{rule}})$ is $\Pi_2^p$-hard, even over a three-node network.

*Proof.* The reduction in the constant-elimination lemma (Lemma 3.2.12) does not only preserve parallel completeness but also parallel soundness, as shown in the appendix (Lemma A.4). □

**Possible generalisations.** At the beginning of this section, we have identified two sources for the rise of complexity from $\Pi_2^p$ for conjunctive queries without negation to coNEXPTIME for conjunctive queries with negation: the unbounded arity of at least one relation and the simultaneous occurrence of a positive and a negated atom referring to a common relation. Indeed, both aspects need to coincide on the same relation.

In this case, the naïve approach to construct a global instance that witnesses the violation of the considered parallel property by addition of ineffectively prohibited facts can lead to an exponentially long chain of alternative valuations that derive the same fact. The syntactical restriction to polarised queries is an obvious way to prevent such chains *completely*, which again allows more efficient algorithms.

However, it is a comparatively strong restriction and it is not hard to see that other restrictions (orthogonal to full and polarised queries) allow equally efficient algorithms. From a complexity-theoretic point of view, it suffices if the *length of chains is constrained by a constant or even a polynomial*. Examples include the following queries.

- *Partially polarised queries:* queries that need to be 'polarised' only on those relations with an arity higher than an *a priori* bound $\alpha$. As an example, for arity bound $\alpha = 2$, the query

$$H() \leftarrow R(x, y), \neg R(y, x), S(z_1, \ldots, z_n), \neg T(z_1, \ldots, z_n)$$

  is 'partially polarised' with respect to relations $S$ and $T$.
- *Negation-full queries:* queries that only contain head variables in negated atoms. As an example, if the boolean query $Q_n$ from Example 3.3.1 is altered by making $y_1, \ldots, y_n$ head variables, this yields a non-full query

$$\begin{aligned} H(y_1, \ldots, y_n) \leftarrow\ & R(x_1), \ldots, R(x_n), R(y_1), \ldots, R(y_n), \\ & S(x_1, \ldots, x_n), \neg S(y_1, \ldots, y_n) \end{aligned}$$

  where the length of the chains is bounded by 2.

The exact boundaries for fragments of query classes determined by the complexity of the parallel correctness problem and its variants are currently unknown. In the next and last section of this chapter, we turn away from such considerations and towards *other settings and related problems*.

## 3.4. Related work and bibliographical remarks

Parallel correctness is a fundamental property that can be studied in several settings. Possible parameters are the query language, the database model, the communication model, the policy formalism and the distributed evaluation process.

This work focuses on basically one such setting: conjunctive queries—sometimes with unions, disequalities and negation—for relational databases under set semantics in the one-round MPC model with distribution policies that are defined on the granularity of a fact. Here, distributed evaluation happens independently on each node, and is defined as the union of the local results. Meanwhile, some other settings have been studied. We give a brief overview of results in publications known to the author next.

**Bag semantics [KNV18].** As often in the literature, this thesis follows Codd's initial approach and views relations in the usual mathematical sense, as *sets*. In practical database management systems however, they are often modelled as *multisets* (or *bags*), where each fact is associated with a multiplicity—representing the number of occurrences.

The following is known about parallel correctness of queries from $\mathsf{CQ}^{\neq}$ under bag semantics due to Ketsman, Neven and Vandevoort [KNV18].

Since each valuation contributes to the multiplicity of a derived fact in this setting, parallel correctness is not characterised in terms of *minimal* valuations only. Instead, a query is parallel-correct under a policy if, for *each* valuation, there is *exactly one* node that is responsible for the required facts.

Deciding parallel correctness under bag semantics is coNP-complete for conjunctive queries and policies from $\mathfrak{P}_{\text{det}}$: for each valuation, the number of nodes that are responsible for the required facts has to be counted.

In general, parallel correctness under bag semantics implies parallel correctness under set semantics but not vice versa. Both notions are equivalent in a restricted case however: for strongly minimal queries under policies that are non-replicating (where at most one node is responsible for each fact). Let us consider this point in a little more detail.

> **Parallel correctness under bag semantics is stronger than under set semantics.** The asymmetrical relationship between the two notions is conform with the following point of view. For bag semantics, parallel completeness could be defined as an approximation from above—each fact is derived locally at least as often as globally—and parallel soundness as an approximation from below—each fact is derived locally at most as often as globally. Then, different from set semantics, parallel soundness is non-trivial under bag semantics, even for queries without negation. Hence, parallel completeness is a strictly weaker property than parallel correctness under bag semantics.
> We note that the previous argument is one reason for the difference between the settings, but it is not the only one. Indeed, the implication from set semantics to bag semantics even fails if restricted to parallel *completeness*. For instance, the boolean query $H() \leftarrow R(u), R(v)$ is parallel-complete under policy $\mathbb{P} = \{(k, R(a)), (\ell, R(b))\}$ with respect to set semantics but not with respect to bag semantics: for the global instance $\mathcal{G} = \{R(a), R(b)\}$ there are four globally satisfying valuations but only two locally satisfying valuations (one on each node). Problematic here is that non-minimal valuations are neglected. Circumventing this problem, we claim that parallel completeness under set semantics implies parallel completeness (in the sense of an overapproximation, as described above) under bag semantics *for strongly minimal queries*.

We return to the results of Ketsman et al. Interestingly, the parallel-correctness property renders some queries 'non-parallelisable' under bag semantics. In general,

this could mean that there is no *asymptotical* runtime benefit in distributing the data because, for every size bound $n \in \mathbb{N}$, there exists a database $\mathbb{D}$ of size $\Theta(n)$ such that at least one node $k$ receives $\Omega(n)$ facts. Here, however, this holds in an even stronger sense that prevents already a *speed-up by a constant factor* because some node receives basically all relevant facts!

As an example, for query $H(x, y) \leftarrow R(x, u), R(u, y)$, all the facts required by some valuation have to be located on the *same* node. The exact complexity of deciding whether a query is 'parallelisable' is currently unknown, but we provide an upper bound in Section 5.2.

Furthermore, if the distributed evaluation is coordinated such that each valuation contributes only once to the result—based on an order on the network—, then parallel correctness can be characterised by Condition (PComp-naïve), that is, by extension of Condition (PComp) to incorporate *all*—not just minimal—valuations. In this scenario, queries are trivially 'parallel-sound' (in the sense of an underapproximation).

**Datalog [KAK18, NSSV19].** Parallel correctness has been studied also for Datalog programs and a more elaborate policy formalism [KAK18]. In the work of Ketsman, Albarghouthi and Koutris, 'production' policies guide the local derivations and 'consumption' policies guide the distribution of facts, *possibly over multiple rounds*.

Since Datalog programs are monotonic, parallel correctness reduces to parallel completeness. In terms of expressibility, Datalog is incomparable to conjunctive queries with negation. It is, however, more expressive than unions of conjunctive queries (without negation). Indeed, it is so expressive that the parallel completeness problem is undecidable—already for very simplistic policies that distribute facts by the relation symbol only. As noted in the context of Corollary* 3.3.8, this complexity/undecidability transfer follows a more general pattern.

The possibility for 'recursion' in Datalog programs raises further questions. In particular, it raises the question whether the program can be evaluated within a fixed number of communication rounds, regardless of the global instance. This is the 'distributed' analogon of the classical *boundedness* problem for Datalog, where the bound refers to the height of derivation trees, which is known to be undecidable [GMSV93]. Some variants of the distributed boundedness problem are also undecidable, again already for fairly simple Datalog programs and policies—at least for more than one communication round—, others are decidable in polynomial time [KAK18, Theorem 7.4].

The aforementioned undecidability of parallel completeness results from the undecidability of query containment for Datalog programs. Follow-up work by Neven, Schwentick, Spinrath and Vandevoort has shown that undecidability also results for weaker fragments with a *decidable* containment problem, namely for 'monadic'

and 'frontier-guarded' Datalog [NSSV19]. There, infeasibility arises not only from the queries but also from their interaction with policies. Notably, the facts are not necessarily distributed in an instance-independent fashion. To a certain extent this is already true in the setting of Ketsman et al. because the communication of derived facts naturally depend on facts in derivation trees for that fact. In some settings studied by Neven et al., however, instance-dependency is also an explicit property of the distribution policies.

The studied classes of communication policies are either hash-based or building on 'data-moving constraints'[4]. For restricted forms of policies, that solely rely on hash-based policies or on 'modest' data-moving constraints, the problem is 2EXPTIME-complete however for both fragments of Datalog. The same holds for the boundedness problem, but only for 'frontier-guarded' Datalog and only under the restriction to communication policies defined by modest data-moving constraints. For general data-moving constraints the problems are undecidable.

Another remarkable point was revealed by the authors: although each monadic Datalog program can be transformed into an equivalent frontier-guarded Datalog program, parallel completeness is undecidable for the former and decidable for the latter under certain conditions [NSSV19, Proposition 26].[5]

**Document spanners [DKM$^+$19].** A large portion of the data analysed today is not relational but textual. *Document spanners* have been introduced as a means to query text documents declaratively [FKRV15]. In practice, information is extracted from texts often in a spatially restricted fashion: many queries address only small segments of the document (e.g., paragraphs, sentences or $N$-grams). This offers the possibility to parallelise the extraction process.

Doleschal et al. have studied the problems *split correctness* and *splittability* for different representations of spanners [DKM$^+$19]. Notably, the distribution of segments in this framework is guided by spanners too—unary spanners, called *splitters*. The splittability problem is a generalisation of parallel correctness. It asks, for a given query $Q$ (spanner) and a policy $\mathbb{P}$ (splitter), whether there is a—possibly different—query $Q'$ (spanner) such that, for each document $D$, query $Q$ yields the same result on $D$ as query $Q'$ on the documents $D_1, \ldots, D_r$ induced by $D$ under $\mathbb{P}$. The special case where $Q'$ is required to equal $Q$, is called *self-splittability* and corresponds with the notion of parallel correctness in this thesis. In the split correctness problem, query $Q'$ is not existentially quantified but provided as part of the input.

[4] Data-moving constraints correspond to a restricted form of *distribution tuple-generating dependencies*, defined and studied in Chapter 5.

[5] This gap between the serial and the parallel setting results by the addition of relations in the standard transformation from monadic to frontier-guarded Datalog, which is not faithfully reflected in the policy.

Split correctness is PSPACE-complete when spanners are defined by regular expression formulas or variable-set automata (even if functional and deterministic). Interestingly, the lower bound for regular expression formulas is again obtained via a reduction from the corresponding containment problem. Furthermore, the restriction of split correctness to policies (splitters) that are guaranteed to induce *disjoint* distributions is in PTIME.

Under the disjointness restriction, splittability is also PSPACE-complete, again for regular expression formulas and variable-set automata. However, the authors leave open the problem to determine the decidability (and complexity) of splittability in general. The picture is more complete on its variant, self-splittability. This problem is PSPACE-complete in general and in PTIME for functional deterministic variable-set automata.

**Bibliographical remarks.** The notions of parallel correctness on a single instance (similar to Definition* 3.1.1) and on all instances induced by a policy (Definition* 3.1.3) have been introduced by Ameloot et al. [AGK+15, AGK+17a] for conjunctive queries. Subsequently, they have been refined to parallel soundness and completeness for conjunctive queries with negation [GKNS16, GKNS19].

Parallel completeness has been characterised via minimal valuations (Theorem* 3.2.5, Definition* 3.2.2) for conjunctive queries first [AGK+15], and later generalised to unions of conjunctive queries with disequalities [GKNS16]. Strongly minimal queries (Section 3.2.2) have been considered from the very start [AGK+15] and *strong* parallel completeness has been introduced as 'strong saturation' [AGK+17a] by the same authors.

The upper and lower complexity bounds in Theorem 3.2.7 are due to the author. The upper bound was given first for CQs [AGK+15] and then generalised to UCQs with disequalities [GKNS16]. The hardness proofs here (Propositions 3.2.10 and 3.2.14) are similar to the original ones [AGK+15] but use constants in the first place; the results for constant-free queries are then obtained again by the use of the previously unpublished 'constant elimination' technique (Lemma 3.2.12). This—hopefully—makes the core argument more clear and is also useful in other places (e.g., in the proof of Proposition 3.3.23).

The coNEXPTIME-bounds for UCQs with negation [GKNS16] (Theorem 3.3.4) are also due to the author. Notably, an early version of the lower bound proof inspired the coNEXPTIME-hardness proofs for the containment problem by Ketsman (Theorem* 3.3.5) and has been changed afterwards to the reduction from containment. (Technically, the reduction in this thesis has been slightly improved to require only a constant rather than a linear number of nodes.)

Furthermore, the coNP-lower bounds for full UCQs with negation (Proposition 3.3.7) have been originally contributed by the author [GKNS16]. In this thesis, we have

changed the second reduction (to parallel *completeness*) such that it resembles more the reduction for Datalog by Ketsman et al. [KAK18, Theorem 5.2]. Both later reductions base on the same main idea—provided in the original proof by the author. The latter proofs, however, result in a stronger version of Corollary* 3.3.8, which the author deems a notable result that seemingly has not been stated explicitly before.

Section 3.3.3 exclusively contains previously unpublished results by the author.

# 4. Parallel-correctness transfer

Like parallel correctness, the *transfer* of parallel correctness is a basic property that allows high-level optimisation in distributed query evaluation. In both cases, the goal is to determine whether the communication phase can be omitted without impairing the evaluation—which would reduce the overall computation time. Let us quickly consider the differences between both approaches.

- *Parallel correctness.* Here, the policy $\mathbb{P}$ for the last computation round is assumed to be known and query $Q_2$ is to be evaluated in the next round.

  If a query $Q_2$ is parallel-correct under policy $\mathbb{P}$, then the current distribution is suitable for the evaluation of $Q_2$.

- *Parallel-correctness transfer.* Now, it is known that query $Q_1$ has been evaluated correctly in the previous round and query $Q_2$ is to be evaluated in the next round.

  If parallel correctness transfers from $Q_1$ to $Q_2$, then the current distribution is suitable for the evaluation of $Q_2$.

Optimisation based on parallel-correctness transfer is thus possible on a higher level, even before concrete policies have been determined.

Clearly, transfer of parallel correctness demands the transfer of its constituent properties, parallel soundness and parallel completeness. The former is trivial for monotonic queries but deserves a separate investigation for non-monotonic queries.

**Structure of this chapter.** We provide a formal definition of transfer in Section 4.1. Then, we start with the study of (unions of) conjunctive queries in Section 4.2 and continue with the extension of this query class to UCQs with negation in Section 4.3. Section 4.4 concludes the chapter with an overview of related results.

For queries in $\mathsf{UCQ}^{\neq}$, parallel-correctness transfer is equivalent to parallel-completeness transfer, which again can be characterised by minimally requiring valuations (Theorem* 4.2.3). Using this characterisation, we prove the problem to be $\Pi_3^p$-complete in general (Theorem 4.2.4). The complexity decreases by one or two levels, however, if query $Q_1$ is known to be strongly minimal. While it has been shown to be $\mathsf{NP}$-complete for unions of conjunctive queries without disequalities (Proposition* 4.2.8), complexity increases again to $\Pi_2^p$-completeness if disequalities occur (Proposition 4.2.15).

For queries in $\mathsf{UCQ}^{\neg,\neq}$, parallel-soundness transfer is not trivial anymore. However, the complexity analysis for—monotonic—CQs and UCQs relies crucially on the characterisation via minimal valuations. Lacking a comparable characterisation for general CQs and UCQs with negation, we turn immediately to a fragment: for polarised queries, we provide a characterisation in Section 3.3.3. Furthermore, we prove the $\Pi_3^p$-completeness of parallel-*soundness* transfer for this class of queries (Theorem 4.3.4). Parallel-*completeness* transfer remains open for this fragment.

## 4.1. Definition

We start with a formal definition of the central notions in this chapter.

**Definition* 4.1.1 (Parallel-correctness transfer).** Let $Q_1$ and $Q_2$ be queries. Parallel correctness *transfers from* $Q_1$ *to* $Q_2$ if the following implication holds for every policy $\mathbb{P}$:

if $Q_1$ is parallel-correct under $\mathbb{P}$, then $Q_2$ is parallel-correct under $\mathbb{P}$.

*Transfer of parallel soundness* and *parallel completeness* is defined analogously. ◀

**Example 4.1.2.** Parallel completeness transfers from query $Q_1 = H() \leftarrow R(x, y), S(y)$ to query $Q_2 = H() \leftarrow R(y, y), S(y)$. Both queries are strongly minimal by Lemma 3.2.16, that is, all its valuations are minimal. The argument below implies transfer by Condition (PComp).

Let $\mathbb{P}$ be a policy under which $Q_1$ is parallel-complete. Then, for each valuation $V_1$ for $Q_1$, there exists a node that is responsible for the facts required by $V_1$. Furthermore, each valuation $V_2$ for $Q_2$ can be identified with a valuation for $Q_1$—which maps both variables $x$ and $y$ to the same value. Hence, for each valuation $V_2$ for $Q_2$, there exists a node that is responsible for the facts required by $V_2$. ■

In the previous example, the queries satisfy the containment relation $Q_2 \sqsubseteq Q_1$. Though it might be tempting to think of a general relationship between containment and parallel-correctness transfer, both concepts are incomparable [AGK+17a]: neither does containment guarantee transfer of parallel-correctness nor vice versa. For instance, for query $Q_1' = H() \leftarrow R(x, y)$, containment $Q_2 \sqsubseteq Q_1 \sqsubseteq Q_1'$ holds but parallel correctness does neither transfer from $Q_1'$ to query $Q_1$ nor to query $Q_2$ since the latter queries require $S$-facts to be placed on some nodes while the former does not.

Parallel-correctness transfer can, however, be seen as a generalisation of parallel correctness. The following definition generalises the variants of the parallel-correctness property from a single policy to a *family* of policies.

**Definition* 4.1.3 (Parallel correctness under a policy family).** Let $\mathcal{P}$ be a family of policies. Query $Q$ is *parallel-correct under* $\mathcal{P}$ if it is parallel-correct under every policy in $\mathcal{P}$. The notions of *parallel-soundness* and *parallel-completeness under a policy family* are defined analogously. ◀

A particularly interesting case arises from families of policies that benefit a fixed query $Q$. More precisely, let $\mathcal{P}_{\text{sound}}(Q)$ and $\mathcal{P}_{\text{comp}}(Q)$ denote the families of policies under which query $Q$ is parallel-sound and parallel-complete, respectively. Similarly, let $\mathcal{P}_{\text{cor}}(Q) \stackrel{\text{def}}{=} \mathcal{P}_{\text{sound}}(Q) \cap \mathcal{P}_{\text{comp}}(Q)$ denote the family of policies under which query $Q$ is parallel-correct.

Then, parallel-correctness transfers from a query $Q_1$ to a query $Q_2$ if and only if $Q_2$ is parallel-correct under $\mathcal{P}_{\text{cor}}(Q_1)$. The next table lists the correspondences more generally.

| **property** | **condition** |
|---|---|
| parallel-soundness transfer | $\mathcal{P}_{\text{sound}}(Q_2) \subseteq \mathcal{P}_{\text{sound}}(Q_1)$ |
| parallel-completeness transfer | $\mathcal{P}_{\text{comp}}(Q_2) \subseteq \mathcal{P}_{\text{comp}}(Q_1)$ |
| parallel-correctness transfer | $\mathcal{P}_{\text{cor}}(Q_2) \subseteq \mathcal{P}_{\text{cor}}(Q_1)$ |

Below, we study the complexity of the following decision problem.

PC-T($\mathcal{Q}_1$,$\mathcal{Q}_2$)

| | |
|---|---|
| **Parameters:** | query classes $\mathcal{Q}_1$ and $\mathcal{Q}_2$ |
| **Input:** | query $Q_1 \in \mathcal{Q}_1$,<br>query $Q_2 \in \mathcal{Q}_2$ |
| **Question:** | Does parallel-correctness transfer from $Q_1$ to $Q_2$? |

If both query classes $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are identical, we simply write PC-T($\mathcal{Q}_1$) or omit the class completely, if it is clear from the context. Of course, variants for the transfer of parallel soundness and parallel completeness are defined analogously and denoted PSound-T and PComp-T, respectively.

## 4.2. Unions of conjunctive queries

In Section 3.2, we started the study of parallel correctness with the observation that, because of monotonicity, unions of conjunctive queries are trivially parallel-sound. Clearly, the same observation also trivialises parallel-soundness transfer: parallel soundness transfers from $Q_1$ to $Q_2$ for *all* queries $Q_1$ and $Q_2$ from $\mathsf{UCQ}^{\neq}$. Hence, we consider, being equivalent to the transfer of parallel-correctness, the transfer

of *parallel completeness* in this section only. For ease of description, we make an additional assumption.

**Remark 4.2.1 (Irredundancy of UCQs).** In the following, we silently assume that every union $Q_1 \cup \ldots \cup Q_n$ of conjunctive queries is *irredundant*: no subquery $Q_i$ is contained in another subquery $Q_j$ for $i, j \in \{1, \ldots, n\}$ where $i \neq j$.

Deciding this property is coNP-complete [CM77] and thus not harder than most of the problems considered below.

We begin with the general case (arbitrary unions of conjunctive queries with disequalities) in Section 4.2.1. Building on a characterisation via minimally requiring valuations, we obtain that parallel-completeness transfer is $\Pi_3^p$-complete. Afterwards, we consider the restriction to strongly minimal queries (for input $Q_1$, the right-hand query $Q_2$ can be arbitrary) in Section 4.2.2. Under this restriction, parallel-completeness transfer is NP-complete for queries *without disequalities* and $\Pi_2^p$-complete for queries *with disequalities*.

## 4.2.1. General case

We quickly recapitulate how parallel-completeness transfer can be characterised and then determine the complexity of the associated decision problem.

### Characterisation

Minimality of valuations is the key notion used to tackle parallel completeness for UCQs in Section 3.2.1. Not astonishingly, they are also useful to characterise the *transfer of* parallel completeness. Since transfer relates two queries, it seems natural to relate valuations for these queries. The next definition formalises a corresponding notion.

**Definition* 4.2.2 (Covering valuation).** Let $Q$ and $Q'$ be queries from $\mathsf{UCQ}^{\neq}$. A valuation $V'$ for $Q'$ *covers* a valuation $V$ for $Q$ if it satisfies both of the following conditions.

1. *Fact condition:* $V'$ requires all facts required by $V$, that is, $V(\mathsf{pos}(Q)) \subseteq V'(\mathsf{pos}(Q'))$.
2. *Domain condition:* $V'$ refers to the same data values as $V$, that is, $\mathsf{dom}(V(\mathsf{pos}(Q))) = \mathsf{dom}(V'(\mathsf{pos}(Q')))$.

A valuation $V'$ that satisfies the fact condition, is called *fact-cover* of $V$. ◀

Using the notions of 'minimality' and 'covers', a simple characterisation of parallel-completeness transfer can be formulated.

**Condition (PComp-T)**

**Assumptions: Let $Q_1$ and $Q_2$ be queries from $\mathsf{UCQ}^{\neq}$.**
For every minimal valuation $V_2$ for $Q_2$,
there is a minimal valuation $V_1$ for $Q_1$ that covers $V_2$.

This condition is indeed a characterisation, as stated in the following theorem.

**Theorem* 4.2.3 (Characterisation of parallel-completeness transfer).** For all queries $Q_1, Q_2$ from $\mathsf{UCQ}^{\neq}$, parallel completeness transfers from $Q_1$ to $Q_2$ if and only if Condition (PComp-T) is satisfied.

Condition (PComp-T) provides a valuable tool to establish matching upper and lower complexity bounds for PCOMP-T, to which we turn now.

### Complexity

The statement in Condition (PComp-T) directly suggests a $\forall\exists$-structure and thus the second level of the polynomial hierarchy. Indeed, the requirement for valuation $V_1$ to be minimal makes an additional level of quantification necessary.

**Theorem 4.2.4.** Decision problem PCOMP-T($\mathcal{Q}$) is $\Pi_3^p$-complete for every query class $\mathcal{Q} \in \{\mathsf{CQ}, \mathsf{CQ}^{\neq}, \mathsf{UCQ}, \mathsf{UCQ}^{\neq}\}$.

*Proof.* The upper bound results from the upper bound provided by Proposition 4.2.6 for the most general query class $\mathsf{UCQ}^{\neq}$. The lower bound follows from the lower bound from Proposition 4.2.7 for the least general query class $\mathsf{CQ}$. □

Two ideas are decisive for the algorithm for PCOMP-T($\mathsf{UCQ}^{\neq}$) presented below. First, Condition (PComp-T) frees us from the consideration of (possibly infinitely many) policies. Second, although the condition formally refers to an infinite set of valuations, it suffices to test the existence of covers for a finite subset of valuations. The latter idea is formalised in the next lemma.

**Lemma 4.2.5.** Let $Q_1$ and $Q_2$ be queries from $\mathsf{UCQ}^{\neq}$ with a maximum number $s$ of variables (per disjunct). The following conditions are equivalent, for $\mathsf{dom}_s = \{1, \ldots, s\}$.

(1) For every minimal valuation $V_2$ for $Q_2$ over $\mathsf{dom}$,
there is a minimal valuation $V_1$ for $Q_1$ over $\mathsf{dom}$ such that $V_2$ is covered by $V_1$.

(2) For every minimal valuation $W_2$ for $Q_2$ over $\mathsf{dom}_s$,
there is a minimal valuation $W_1$ for $Q_1$ over $\mathsf{dom}_s$ such that $W_2$ is covered by $W_1$.

*Proof.* We sketch both implications independently.

⋙ Condition (1) implies Condition (2). For a direct proof, assume that Condition (1) holds. Let $W_2$ be a minimal valuation for query $Q_2$ over $\mathsf{dom_s}$. By Condition (1), there is a valuation $V_1$ for query $Q_1$ over the unrestricted domain $\mathsf{dom}$ that covers $W_2$. The notion of a cover guarantees, first, $W_2(\mathsf{pos}(Q_2)) \subseteq V_1(\mathsf{pos}(Q_1))$ and, second, $\mathsf{dom}(W_2(\mathsf{pos}(Q_2))) = \mathsf{dom}(V_1(\mathsf{pos}(Q_1)))$. The latter particularly implies that valuation $V_1$ refers to data values from $\mathsf{dom_s}$ only. Therefore, Condition (2) holds.

⋙ Condition (2) implies Condition (1). Assume now that Condition (2) holds and let $V_2$ be a minimal valuation for query $Q_2$ over the unrestricted domain $\mathsf{dom}$. Since $Q_2$ contains at most $s$ variables, there exists a permutation $\pi : \mathsf{dom} \to \mathsf{dom}$ such that

$$\pi\big(\mathsf{dom}(V_2(\mathsf{pos}(Q_2)))\big) \subseteq \mathsf{dom_s}$$

holds. Consequently, valuation $W_2 \stackrel{\text{def}}{=} \pi \circ V_2$ is a minimal valuation for $Q_2$ that refers only to values from $\mathsf{dom_s}$. Hence, by Condition (2), there is a valuation $W_1$ for query $Q_1$ that covers $W_2$. Then, a valuation $V_1 \stackrel{\text{def}}{=} \pi^{-1} \circ W_1$ that covers $V_2$ exists because application of the bijection $\pi^{-1}$ preserves the set relationships guaranteed by $W_1$ covering $W_2$, yielding

$$V_2(\mathsf{pos}(Q_2)) = (\pi^{-1} \circ W_2)(\mathsf{pos}(Q_2)) \subseteq (\pi^{-1} \circ W_1)(\mathsf{pos}(Q_1)) = V_1(\mathsf{pos}(Q_1))$$

and, similarly, $\mathsf{dom}(V_2(\mathsf{pos}(Q_2))) = \mathsf{dom}(V_1(\mathsf{pos}(Q_1)))$. Therefore, Condition (1) holds.

Since both conditions mutually imply each other, they are equivalent. □

As mentioned above, we are now ready to prove the upper bound for parallel-completeness transfer constructively.

**Proposition 4.2.6.** Decision problem PCOMP-T($\mathsf{UCQ}^{\neq}$) is in $\Pi_3^p$.

*Proof.* The following algorithm, $\mathbb{A}^{\text{trans}}$, decides parallel-completeness transfer for unions of conjunctive queries with disequalities, using three levels of quantified extra input.

Algorithm $\mathbb{A}^{\text{trans}}$ is correct by Theorem* 4.2.3 and Lemma 4.2.5. We briefly argue how transfer of parallel completeness is *verified—falsification* of transfer follows in a similar fashion.

Condition (PComp-T) states a requirement only for minimal valuations for $Q_2$. If a valuation $V_2$ for $Q_2$ is not minimal, then there is a valuation $W_2$ witnessing that. Since it is of size linear in $|Q_2|$, it can be 'guessed' as part of the existential input. In the positive case, the algorithm accepts in Line 2 because a non-minimal valuation $V_2$ does not contradict the condition.

If valuation $V_2$ is minimal however, there has to be a valuation $V_1$ for $Q_1$ that is minimal and a cover of $V_2$. This valuation $V_1$ is also 'guessed' with the existentially quantified extra input. Since it satisfies the conditions of a cover and, by minimality, there is no valuation $W_1$ that derives the same fact with fewer required facts, the 'reject' statement in Line 4 is not reached, and the algorithms accepts.

**Algorithm** $\mathbb{A}^{\mathsf{trans}}$

**Input:** query $Q_1 \in \mathsf{UCQ}^{\neq}$,
query $Q_2 \in \mathsf{UCQ}^{\neq}$

**Quantified extra input:**

- $\forall$ valuation $V_2$ for $Q_2$ over $\mathsf{dom_s}$ (where $s = \|Q_1\| + \|Q_2\|$)
- $\exists$ valuation $W_2$ for $Q_2$ and $V_1$ for $Q_1$ over $\mathsf{dom_s}$
- $\forall$ valuation $W_1$ for $Q_1$ over $\mathsf{dom_s}$

1: **if** $W_2 <^{\text{pos}}_{Q_2} V_2$ **then** {ignore non-minimal valuations for $Q_2$ }
2: **accept**
3: **if** $W_1 <^{\text{pos}}_{Q_1} V_1$ or $V_1$ is not a cover of $V_2$ **then**
4: **reject**
5: **accept**

Clearly, for each combination of extra input, the test can be accomplished in polynomial time. For the comparisons with respect to preorder $<^{\text{pos}}$ and the 'cover conditions', the sets of induced facts have to be computed and tested for equality or subset relationship, respectively. We have described the details for similar tests in Algorithm $\mathbb{A}$ for parallel completeness (page 30). □

The lower bound stated in the next proposition results by a polynomial reduction from the following problem, known to be $\Pi_3^p$-complete [Sto76].

$\Pi_3$-QBF

**Input:** Formula $\varphi = \forall \boldsymbol{x} \exists \boldsymbol{y} \forall \boldsymbol{z}\ \psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$,
where $\psi$ is a propositional formula in 3-DNF over propositions
$\boldsymbol{x} = (x_1, \ldots, x_r)$, $\boldsymbol{y} = (y_1, \ldots, y_s)$ and $\boldsymbol{z} = (z_1, \ldots, z_t)$

**Question:** Does, for every truth assignment $\beta_{\boldsymbol{x}}$ on $\boldsymbol{x}$,
exist a truth assignment $\beta_{\boldsymbol{y}}$ on $\boldsymbol{y}$ such that,
for every truth assignment $\beta_{\boldsymbol{z}}$ on $\boldsymbol{z}$,
truth assignment $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}} \cup \beta_{\boldsymbol{z}}$ satisfies $\psi$?

**Proposition 4.2.7.** Decision problem PCOMP-T(CQ) is $\Pi_3^p$-hard.

*Proof.* We define a mapping from inputs for $\Pi_3$-QBF to inputs of PCOMP-T(CQ) and show that it is a polynomial reduction.

Every input formula $\varphi = \forall \boldsymbol{x} \exists \boldsymbol{y} \forall \boldsymbol{z}\ \psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ for $\Pi_3$-QBF with a quantifier-free subformula $\psi$ in 3-DNF is mapped to a pair $(Q_1, Q_2)$ of queries in the following way. Let $\boldsymbol{x} = (x_1, \ldots, x_r)$, $\boldsymbol{y} = (y_1, \ldots, y_s)$ and $\boldsymbol{z} = (z_1, \ldots, z_t)$ be the propositions of $\varphi$ and let $\psi$ consist of clauses $C_1, \ldots, C_p$ where $C_j = (\ell_{j,1} \wedge \ell_{j,2} \wedge \ell_{j,3})$ for every $j \in \{1, \ldots, p\}$. Without loss of generality, we assume that each proposition occurs in at least one clause.

The schema for the queries contains unary relations $\texttt{Res}, \texttt{False}, \texttt{True}$ for 'result' and truth values and also unary relations $\texttt{XVal}_h, \texttt{YVal}_i$, for every $h \in \{1, \ldots, r\}$ and every $i \in \{1, \ldots, s\}$, for mappings of the propositions. Additionally, there is a binary relation $\texttt{Neg}$, a ternary relation $\texttt{Or}$ and a quaternary relation $\texttt{And}$ to guide the 'evaluation' of the represented subformula $\psi$.

Query $Q_2$ refers to 'truth variables' $w_0, w_1$, intended to represent truth values *false* and *true*, and it refers to variables $x_1, \ldots, x_r$ for the respective propositions. The query is defined by

$$\begin{aligned} \mathsf{head}(Q_2) &\stackrel{\text{def}}{=} H(w_0, w_1, x_1, \ldots, x_r) \text{ and} \\ \mathsf{pos}(Q_2) &\stackrel{\text{def}}{=} \mathcal{A}_{\text{fix}} \cup \mathcal{A}_{\boldsymbol{y}}^{0,1} \cup \{\texttt{Res}(w_1)\}, \end{aligned}$$

where the sets of atoms

$$\begin{aligned} \mathcal{A}_{\text{fix}} &\stackrel{\text{def}}{=} \{\texttt{False}(w_0), \texttt{True}(w_1)\} \cup \{\texttt{XVal}_h(x_h) \mid h \in \{1, \ldots, r\}\} \text{ and} \\ \mathcal{A}_{\boldsymbol{y}}^{0,1} &\stackrel{\text{def}}{=} \{\texttt{YVal}_i(w_0), \texttt{YVal}_i(w_1) \mid i \in \{1, \ldots, s\}\} \end{aligned}$$

are intended to fix the values of the query's variables, in the case of $\mathcal{A}_{\text{fix}}$, and provide both truth values for variables $y_1, \ldots, y_s$, in the case of $\mathcal{A}_{\boldsymbol{y}}^{0,1}$. Note also that query $Q_2$ is strongly minimal because it is full.

Query $Q_1$ refers to the same variables as $Q_2$ and some additional variables. First, it refers to 'literal variables' $\pi, \bar{\pi}$ for every $\pi \in \{x_1, \ldots, x_r, y_1, \ldots, y_s, z_1, \ldots, z_t\}$. Second, it refers to variables $\alpha_1, \ldots, \alpha_p$ and $\omega_1, \ldots, \omega_p$, with the intention that variable $\alpha_j$ represents the truth value of clause $C_j$ and variable $\omega_j$ represents the truth value of the partial disjunction $C_1 \vee \ldots \vee C_j$ for every $j \in \{1, \ldots, p\}$. The query is defined by

$$\begin{aligned} \mathsf{head}(Q_1) &\stackrel{\text{def}}{=} H(w_0, w_1, x_1, \ldots, x_r, y_1, \ldots, y_s) \text{ and} \\ \mathsf{pos}(Q_1) &\stackrel{\text{def}}{=} \mathcal{A}_{\text{fix}} \cup \mathcal{A}_{\boldsymbol{y}} \cup \mathcal{A}_{\text{sat}} \cup \mathcal{A}_{\psi} \cup \{\texttt{Res}(w_0), \texttt{Res}(\omega_p)\}, \end{aligned}$$

where $\mathcal{A}_{\text{fix}}$ is the set of atoms defined before and the other sets of atoms are defined next. Set $\mathcal{A}_{\boldsymbol{y}} \stackrel{\text{def}}{=} \{\texttt{YVal}_i(y_i), \texttt{YVal}_i(\bar{y}_i) \mid i \in \{1, \ldots, s\}\}$ is intended to force a mapping of positive and negated literal variables to complementary truth values. Set

$$\begin{aligned}\mathcal{A}_{\text{sat}} \stackrel{\text{def}}{=} &\{\mathtt{Neg}(w_0, w_1), \mathtt{Neg}(w_1, w_0)\}\\ &\cup \{\mathtt{And}(\boldsymbol{w}, w_0) \mid \boldsymbol{w} \in \mathbb{W}^-\} \cup \{\mathtt{And}(w_1, w_1, w_1, w_1)\}\\ &\cup \{\mathtt{Or}(w_0, w_0, w_0), \mathtt{Or}(w_0, w_1, w_1), \mathtt{Or}(w_1, w_0, w_1), \mathtt{Or}(w_1, w_1, w_1)\}\end{aligned}$$

represents the logical functions negation, conjunction and disjunction if the last position is considered as the output for the previous positions. To this end, the set $\mathbb{W}^- \stackrel{\text{def}}{=} \{w_0, w_1\}^3 - \{(w_1, w_1, w_1)\}$ contains all triples of truth variables with at least one false component. Finally, set

$$\begin{aligned}\mathcal{A}_{\psi} \stackrel{\text{def}}{=} &\{\mathtt{Neg}(\pi, \bar{\pi}) \mid \text{for each proposition } \pi \text{ in } \psi\}\\ &\cup \{\mathtt{And}(\ell_{j,1}, \ell_{j,2}, \ell_{j,3}, \alpha_j) \mid j \in \{1, \ldots, p\}\}\\ &\cup \{\mathtt{Or}(\alpha_1, \alpha_1, \omega_1), \mathtt{Or}(\omega_1, \alpha_2, \omega_2), \ldots, \mathtt{Or}(\omega_{p-1}, \alpha_p, \omega_p)\}\end{aligned}$$

is intended to represent the literals of subformula $\psi$ as well as its (conjunctive) clauses and the results of their partial disjunctions.

The pair $(Q_1, Q_2)$ of queries can obviously be computed from formula $\varphi$ in polynomial time and the mapping defined in this way is total. Before we continue with a proof of the reduction property, we shortly illustrate the mapping and its workings.

**Illustration.** The claimed reduction maps the following $\Pi_3$-QBF-formula

$$\varphi = \forall x_1 \exists y_1, y_2 \forall z_1 \big((x_1 \wedge y_1 \wedge z_1) \vee (\neg x_1 \wedge y_2 \wedge z_1)\big)$$

to a pair $(Q_1, Q_2)$ of conjunctive queries. The latter query $Q_2$, with head $H(w_0, w_1, x_1)$ and body

$$\begin{aligned}&\mathtt{False}(w_0), \mathtt{True}(w_1), \mathtt{XVal}_1(x_1), \mathtt{Res}(w_1),\\ &\mathtt{YVal}_1(w_0), \mathtt{YVal}_1(w_1), \mathtt{YVal}_2(w_0), \mathtt{YVal}_2(w_1),\end{aligned}$$

is intended to represent the first block of universally quantified variables ($\forall x_1$). As mentioned, this query is strongly minimal because it is full.
The former query $Q_1$, with head $H(w_0, w_1, x_1, y_1, y_2)$ and body

$$\begin{aligned}&\mathtt{False}(w_0), \mathtt{True}(w_1), \mathtt{XVal}_1(x_1), \mathtt{Res}(w_0), \mathtt{Res}(\omega_2),\\ &\mathtt{YVal}_1(y_1), \mathtt{YVal}_1(\bar{y}_1), \mathtt{YVal}_2(y_2), \mathtt{YVal}_2(\bar{y}_2),\\ &\mathtt{Neg}(w_0, w_1), \mathtt{Neg}(w_1, w_0), \text{ and the remaining atoms from } \mathcal{A}_{sat},\\ &\mathtt{Neg}(x_1, \bar{x}_1), \mathtt{Neg}(y_1, \bar{y}_1), \mathtt{Neg}(y_2, \bar{y}_2), \mathtt{Neg}(z_1, \bar{z}_1),\\ &\mathtt{And}(x_1, y_1, z_1, \alpha_1), \mathtt{And}(\bar{x}_1, y_2, z_1, \alpha_2), \mathtt{Or}(\alpha_1, \alpha_1, \omega_1), \mathtt{Or}(\omega_1, \alpha_2, \omega_2),\end{aligned}$$

is intended to represent the second block of existentially quantified variables ($\exists y_1, y_2$) and also the third block of universally quantified variables ($\forall z_1$). Here, minimality helps to capture *universal* quantification via a *single* mapping by a dual perspective: if there exists a non-satisfying extension, then it requires fewer facts; otherwise all extensions are satisfying, and require the same additional fact.

For these inputs, both $\varphi \notin \Pi_3$-QBF and $(Q_1, Q_2) \notin$ PComp-T(cq) hold.

The former is easy to see. Formula $\varphi$ is *not* in $\Pi_3$-QBF because, regardless how a truth assignment for $x_1$ and $y_1, y_2$ is defined, its extension on the universally quantified variable $z_1$ by $z_1 \mapsto 0$ is *not* satisfying for $\psi$ since both conjunctive clauses contain the literal $z_1$. In particular, this argument works for the truth assignment $\beta_{\boldsymbol{x}} \stackrel{\text{def}}{=} \{x_1 \mapsto 0\}$.

The latter is a consequence of the following observation: there is a minimal valuation $V_2$ for $Q_2$ that is not covered by any minimal valuation $V_1$ for $Q_1$. We define valuation $V_2$ based on the truth assignment $\beta_{\boldsymbol{x}}$ mentioned above, let $V_2 \stackrel{\text{def}}{=} \{w_0 \mapsto 0, w_1 \mapsto 1, x_1 \mapsto 0\}$. This valuation, which is minimal, requires facts

$$\texttt{False}(0), \texttt{True}(1), \texttt{XVal}_1(0), \texttt{Res}(1),$$
$$\texttt{YVal}_1(0), \texttt{YVal}_1(1), \texttt{YVal}_2(0), \texttt{YVal}_2(1)$$

for query $Q_2$. Let us now assume, towards a contradiction, that there exists a minimal valuation $V_1$ for $Q_1$ that covers $V_2$. Then, the first four facts above imply that $V_1(w_0, w_1, x_1, \omega_2) = (0, 1, 0, 1)$ holds. Analogously, the latter four facts imply that $V_1(y_1, \bar{y}_1)$ and $V_1(y_2, \bar{y}_2)$ are either $(0, 1)$ or $(1, 0)$ and thus induce a (valid) truth assignment $\beta_{\boldsymbol{y}}$ on $\boldsymbol{y}$, namely $\beta_{\boldsymbol{y}} \stackrel{\text{def}}{=} \{y_1 \mapsto V_1(y_1), y_2 \mapsto V_1(y_2)\}$. The mapping $V_1(w_0, w_1) = (0, 1)$ guarantees by $V_1(\mathcal{A}_{\text{sat}})$ that $V_1$ requires all `Neg`-, `And`- and `Or`-facts that represent the corresponding logical connectives, the 'logical facts'. The assumed minimality of $V_1$ then only allows $V_1(\omega_2) = 0$, the desired contradiction, as argued next. If $V_1(\omega_2) = 1$, then there is a valuation $W_1$ that agrees with $V_1$ on the previously mentioned variables, maps $(z_1, \bar{z}_1) \mapsto (0, 1)$ and maps the variables $\alpha_1, \alpha_2, \omega_1, \omega_2$ all to 0—according to their intended meaning. This valuation $W_1$ then requires only facts

$$\texttt{False}(0), \texttt{True}(1), \texttt{XVal}_1(0), \texttt{Res}(0),$$
$$\texttt{YVal}_1(0), \texttt{YVal}_1(1), \texttt{YVal}_2(0), \texttt{YVal}_2(1),$$
$$\texttt{Neg}(0, 1), \texttt{Neg}(1, 0)$$

and the remaining 'logical facts', contradicting the minimality of $V_1$.
Therefore, fact $\texttt{Res}(1)$ is required by $V_2$ but not by $V_1$. Hence, the assumed valuation $V_1$ does not exist and thus $(Q_1, Q_2) \notin \text{PCOMP-T}(\texttt{CQ})$ because the queries do not satisfy Condition (PComp-T).

Let us turn back to the reduction property. Not astonishingly, we are interested in valuations with range $\{0, 1\}$ and particularly in such that map $(w_0, w_1)$ to $(0, 1)$ and $(\pi, \bar{\pi})$ to either $(0, 1)$ or $(1, 0)$ for each proposition $\pi$. We call these valuations *boolean*. To facilitate the following arguments, we call a boolean valuation $V_1$ for query $Q_1$ *compatible* if it respects the intended meaning of the query's other variables: variables $\alpha_1, \ldots, \alpha_p$ and $\omega_1, \ldots, \omega_p$ are mapped to the truth values of the represented clauses and partial disjunctions, which is unambiguously possible for boolean valuations. Thus, a boolean valuation $V_1$ for $Q_1$ is compatible if and only if $V_1(\mathcal{A}_\psi) \subseteq V_1(\mathcal{A}_{\text{sat}})$ holds.

We will repeatedly define a boolean valuation $V$ from a truth assignment $\beta$ and vice versa. It is in this sense that we say that the mappings are *induced*. This means that $V(w_0, w_1) = (0, 1)$ holds and also $V(\pi) = \beta(\pi)$ for every proposition $\pi$ in the context—implicitly defined by the mapping or stated explicitly. If valuation $V$ is for the *left-hand* query $Q_1$, it is furthermore required to map $\bar{\pi}$ to the complement of $V(\pi)$ for every $\pi$ under consideration.

With the previous observation, we show in the remainder of this proof that the mapping defined in the beginning satisfies the reduction property, that is, $\varphi \in \Pi_3$-QBF holds if and only if $(Q_1, Q_2) \in$ PComp-T(CQ). As usual, we prove both implications separately.

⋙ If. For a proof by contraposition, we assume that $\varphi \notin \Pi_3$-QBF and show that Condition (PComp-T) is violated for the pair $(Q_1, Q_2)$ of queries for $\varphi$.

By our assumption $\varphi \notin \Pi_3$-QBF, there exists a truth assignment $\beta_{\boldsymbol{x}}$ for $\boldsymbol{x}$ such that, for every truth assignment $\beta_{\boldsymbol{y}}$ for $\boldsymbol{y}$, there is a truth assignment $\beta_{\boldsymbol{z}}$ for $\boldsymbol{z}$ such that the combined assignment $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}} \cup \beta_{\boldsymbol{z}}$ does *not* satisfy subformula $\psi$ of $\varphi$. Using this assignment $\beta_{\boldsymbol{x}}$, we define valuation $V_2$ for $Q_2$ by

$$V_2 \stackrel{\text{def}}{=} \{w_0 \mapsto 0, w_1 \mapsto 1, x_1 \mapsto \beta_{\boldsymbol{x}}(x_1), \ldots, x_r \mapsto \beta_{\boldsymbol{x}}(x_r)\},$$

which is minimal since $Q_2$ is strongly minimal.

There is however no valuation $V_1$ for $Q_1$ that is both minimal and covering for $V_2$. Towards a proof by contradiction, assume that such a valuation $V_1$ exists. The fact-condition then particularly implies $V_2(\mathsf{pos}(Q_2)) \subseteq V_1(\mathsf{pos}(Q_1))$, and hence the following three properties.

(P1) Valuation $V_1$ agrees with $V_2$ on variables $w_0, w_1$ and $x_1, \ldots, x_r$.

(P2) Valuation $V_1$ maps $\omega_p$ to 1.

(P3) Valuation $V_1$ maps $(y_i, \bar{y}_i)$ either to $(0, 1)$ or to $(1, 0)$ for every $i \in \{1, \ldots, s\}$.

Property (P1) is implied because of the atoms in $\mathcal{A}_{\text{fix}}$, which are the only atoms that refer to relations `True`, `False` and $\texttt{XVal}_h$, for $h \in \{1, \ldots, r\}$. Property (P2) is implied because $V_2(\texttt{Res}(w_1))$ can only be covered by $V_1(\texttt{Res}(\omega_p))$, not by the only other `Res`-fact $V_1(\texttt{Res}(w_0)) = \texttt{Res}(0)$. Finally, Property (P3) is implied because, for each $i \in \{1, \ldots, s\}$, facts $\texttt{YVal}_i(0), \texttt{YVal}_i(1)$ are required by $V_2$ due to $\mathcal{A}^{0,1}_{\boldsymbol{y}}$. They can only be required by $V_1$ if variables $y_i, \bar{y}_i$ are mapped to complementary values from $\{0, 1\}$.

Since Property (P3) holds, valuation $V_1$ unambiguously induces a truth assignment $\beta_{\boldsymbol{y}}$ on $\boldsymbol{y}$, where $\beta_{\boldsymbol{y}}(y_1, \ldots, y_s) = V_1(y_1, \ldots, y_s)$. From our starting assumption, it follows that there is an assignment $\beta_{\boldsymbol{z}}$ such that $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}} \cup \beta_{\boldsymbol{z}}$ does *not* satisfy $\psi$. Let $W_1$ be the compatible valuation induced by $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}} \cup \beta_{\boldsymbol{z}}$. We claim that $W_1 <_{Q_1} V_1$, which contradicts the assumed minimality of $V_1$. First, both

valuations obviously agree on the head variables. Second, they require the same facts for $\mathcal{A}_{\text{fix}}$, $\mathcal{A}_{\boldsymbol{y}}$ and $\mathcal{A}_{\text{sat}}$. If $V_1$ is *not* compatible, then $W_1 <_{Q_1} V_1$ is clear. Otherwise, both valuation are compatible and thus require no additional facts for $\mathcal{A}_\psi$. However, $W_1(\omega_p) = 0$ by the choice of $\beta_{\boldsymbol{z}}$, while $V_1(\omega_p) = 1$, and thus $W_1$ requires one $\texttt{Res}$-fact less than $V_1$. Valuation $V_1$ is hence not minimal, the desired contradiction.

Therefore, Condition (PComp-T) is violated and parallel completeness does *not* transfer from query $Q_1$ to query $Q_2$.

⋙ Only if. Assume that $\varphi \in \Pi_3$-QBF. We show that every (minimal) valuation $V_2$ for $Q_2$ has a covering minimal valuation $V_1$ for $Q_1$.

Let $V_2$ be an arbitrary valuation for $Q_2$, which maps the truth variables to some data values $V_2(w_0) = c_0$ and $V_2(w_1) = c_1$. Without loss of generality, we may assume that $c_0 = 0$ and $c_1 \in \{0, 1\}$ because the cover conditions are not affected by the application of a bijection over $\mathsf{dom}$ on both valuations. We distinguish three cases, depending on how $V_2$ maps the head variables $w_0, w_1, x_1, \ldots, x_r$.

» 1. Case ($V_2$ is boolean). Valuation $V_2$ induces a truth assignment $\beta_{\boldsymbol{x}}$ on $\boldsymbol{x}$. Since $\varphi \in \Pi_3$-QBF, there is an assignment $\beta_{\boldsymbol{y}}$ on $\boldsymbol{y}$ such that, for every assignment $\beta_{\boldsymbol{z}}$ on $\boldsymbol{z}$, it holds $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}} \cup \beta_{\boldsymbol{z}} \models \psi$. In particular, this holds for the assignment $\beta_{\boldsymbol{z}}$ that maps every proposition $z_1, \ldots, z_t$ to 0. Let $V_1$ be the compatible valuation induced by these assignments, $\beta_{\boldsymbol{x}}$, $\beta_{\boldsymbol{y}}$ and $\beta_{\boldsymbol{z}}$. We claim that this valuation covers $V_2$ and that it is minimal.

That $V_1$ covers $V_2$ follows from the next two observations. First, both valuations are boolean and thus refer to the same set of data values, namely to $\{0, 1\}$. Second, both valuations agree on variables $w_0, w_1, x_1, \ldots, x_r$ by definition and thus $V_2(\mathcal{A}_{\text{fix}}) = V_1(\mathcal{A}_{\text{fix}})$. Furthermore, $V_2(\mathcal{A}_{\boldsymbol{y}}^{0,1}) = V_1(\mathcal{A}_{\boldsymbol{y}})$ because $V_1$ is boolean. Compatibility of $V_1$ and the choice of $\beta_{\boldsymbol{y}}$ further ensure that $V_1(\omega_p) = 1$. Therefore, the only additional fact $V_2(\texttt{Res}(w_1)) = \texttt{Res}(1)$ required by $V_2$ is also required by $V_1$. Hence, $V_2(\mathsf{pos}(Q_2)) \subseteq V_1(\mathsf{pos}(Q_1))$.

Valuation $V_1$ is also minimal. Towards a contradiction, assume that a valuation $U_1$ with $U_1 <_{Q_1}^{\text{pos}} V_1$ exists. Clearly, both valuations require the same facts for the atoms in $\mathcal{A}_{\text{fix}} \cup \mathcal{A}_{\text{sat}}$. They also require the same facts for $\mathcal{A}_{\boldsymbol{y}}$. This is immediate for head variables $y_1, \ldots, y_s$. For their counterparts $\bar{y}_1, \ldots, \bar{y}_s$, this follows from the observation that $V_1(\mathcal{A}_{\text{sat}} \cup \mathcal{A}_\psi) = V_1(\mathcal{A}_{\text{sat}})$ and the assumption that $U_1$ requires no $\texttt{Neg}$-facts except for $\texttt{Neg}(0, 1)$ and $\texttt{Neg}(1, 0)$. Hence, valuation $U_1$ encodes truth assignments $\beta_{\boldsymbol{x}}$, $\beta_{\boldsymbol{y}}$ and $\beta'_{\boldsymbol{z}}$ on the respective propositions, which particularly agree with the defining ones for $V_2$ on $x_1, \ldots, x_r$ and $y_1, \ldots, y_s$. Strict containment $U_1(\mathsf{pos}(Q_1)) \subsetneq V_1(\mathsf{pos}(Q_1))$ then implies $U_1(\omega_p) = 0$ and compatibility of $U_1$. Compatibility, in turn, witnesses that $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}} \cup \beta'_{\boldsymbol{z}}$ does *not* satisfy $\psi$, a contradiction to the choice of $\beta_{\boldsymbol{y}}$.

Therefore, valuation $V_1$ is minimal and covering for $V_2$.

» 2. Case ($V_2(w_1) = 0$ and $V_2(\{x_1, \ldots, x_r\}) \subseteq \{0, 1\}$). Let $W_1$ be the valuation

for $Q_1$ that agrees with $V_2$ on variables $w_0, w_1$ and $x_1, \ldots, x_r$ and that maps all other variables to 0. Let $V_1$ be a minimal valuation with $V_1 \leq_{Q_1} W_1$. Then, $V_1$ also agrees with $V_2$ on the head variables of $Q_2$, implying $V_2(\mathcal{A}_{\text{fix}}) = V_1(\mathcal{A}_{\text{fix}})$. By the case assumption, valuation $V_2$ requires only facts $\{\texttt{YVal}_1(0), \ldots, \texttt{YVal}_s(0)\}$ for $\mathcal{A}_{\boldsymbol{y}}^{0,1}$, which are also required by $V_1$ for $\mathcal{A}_{\boldsymbol{y}}$ because $V_1$ agrees with $W_1$ on the head variables $y_1, \ldots, y_s$. Finally, fact $V_2(\texttt{Res}(w_1)) = \texttt{Res}(0)$ is also required by $V_1$ because of $\texttt{Res}(w_0) \in \mathsf{pos}(Q_1)$. Furthermore, valuation $V_1$ refers to data value 1 only if valuation $V_2$ does (for some $x_i$). Hence, valuations $V_1$ and $V_2$ refer to the same set of data values, either to $\{0\}$ or to $\{0, 1\}$.

Therefore, valuation $V_1$ is minimal and covering for $V_2$.

» 3. Case ($V_2(\{x_1, \ldots, x_r\}) \not\subseteq \{0, 1\}$). To show that there is a covering valuation for $V_2$, we have to deal carefully with the data values from $V_2$ that are not interpreted as false or true. The following argument works independently of the mapping $V_2(w_1) \in \{c_0, c_1\} \subseteq \{0, 1\}$.

By the case assumption, there is at least one variable $x_h$ that is mapped by $V_2$ to a data value different from 0 and 1. We call each variable $x_h$ with this property *foul*. Similarly, we call a clause *foul* if it contains $x_h$ or $\neg x_h$ for a foul variable $x_h$. Furthermore, we fix one of the data values that a foul variable is mapped to, say $d \stackrel{\text{def}}{=} V_2(x_h)$ for the minimal index $h$ of a foul variable.

As an intermediate step, we define a valuation $W_1$ for $Q_1$, which is not necessarily minimal, in two steps. First, valuation $W_1$ satisfies the following equations on the truth and literal variables

$$W_1(w_0, w_1, x_1, \ldots, x_r) = V_2(w_0, w_1, x_1, \ldots, x_r) \tag{4.1}$$
$$W_1(y_1, \ldots, y_s, z_1, \ldots, z_t) = (c_0, \ldots, c_0) \tag{4.2}$$
$$W_1(\bar{y}_1, \ldots, \bar{y}_s, \bar{z}_1, \ldots, \bar{z}_t) = (c_1, \ldots, c_1) \tag{4.3}$$
$$W_1(\bar{x}_i) = \overline{V_2(x_i)} \text{ for every non-foul variable } x_i \tag{4.4}$$
$$W_1(\bar{x}_i) = d \text{ for every foul variable } x_i, \tag{4.5}$$

where $\overline{c_0} = c_1$ and $\overline{c_1} = c_0$. Second, valuation $W_1$ is successively defined on the remaining variables. For each $j$ from 1 to $p$,

- variable $\alpha_j$ is mapped to
  - 1 if clause $C_j$ is foul or all its literals are mapped to 1, and to
  - 0 otherwise;
- variable $\omega_j$ is mapped to
  - 1 if $W_1(\alpha_i) = 1$ for some $i \in \{1, \ldots, j\}$, and to
  - 0 otherwise.

Although valuation $W_1$ covers valuation $V_2$, it is not necessarily minimal. In the remainder of this proof, we show that each minimal valuation 'below' $W_1$ is also

a cover of $V_2$. To this end, let $V_1$ be a minimal valuation such that $V_1 \leq_{Q_1} W_1$ holds. We claim that $V_1$ covers $V_2$.

First, we consider the domain condition. Because of Equation (4.1), valuation $W_1$ refers to all data values that valuation $V_2$ refers to. These include $c_0, c_1$ and $d$, to which all remaining variables are mapped to by $W_1$. Therefore, both valuations refer to the same set of data values. Since $w_0, w_1, x_1, \ldots, x_r$ are head variables of $Q_1$ and $V_1 \leq_{Q_1} W_1$, this also holds for $V_1$.

Next, we consider the fact condition. Valuation $V_1$ requires all of the facts from $V_2(\mathcal{A}_{\text{fix}} \cup \mathcal{A}_{\boldsymbol{y}}^{0,1})$ for the atoms in $\mathcal{A}_{\text{fix}} \cup \mathcal{A}_{\boldsymbol{y}}$ because of the agreement on variables $w_0, w_1$ and $x_1, \ldots, x_r$ and Equations (4.2) and (4.3). It remains to show that $V_1$ also requires fact $V_2(\texttt{Res}(w_1)) = \texttt{Res}(c_1)$.

By the case assumption, there is an index $j_0 \in \{1, \ldots, p\}$ such that the $j_0$-th clause is foul because of the foul variable $x_h$ with minimal index $h$. Without restriction, we assume that $x_h$ is referred to in position 1. Then, valuation $V_1$ requires a fact of the form $\texttt{And}(d, \cdot, \cdot, \cdot)$. More concretely, this fact is of the form $\texttt{And}(d, \cdot, \cdot, c_1)$ because $V_1 \leq_{Q_1} W_1$ and valuation $W_1$ by definition only requires $\texttt{And}$-facts with value $d$ that have value $c_1$ in the last position. Thus, $V_1(\alpha_{j_0}) = c_1$. Finally, valuation $W_1$ only requires $\texttt{Or}$-facts from $W_1(\mathcal{A}_{\text{sat}})$, where value $c_1$ in one of the first two positions enforces value $c_1$ in the last position. Thus, we can even conclude $V_1(\omega_{j_0}) = c_1$. By the same argument, this holds for every $j \geq j_0$ and thus for $j = p$ in particular. Therefore, valuation $V_1$ requires fact $V_1(\texttt{Res}(\omega_p)) = \texttt{Res}(c_1)$, as claimed.

Hence, valuation $V_1$ is minimal and covering for $V_2$.

This concludes the case distinction and proves that every valuation $V_2$ for $Q_2$ has a minimal covering valuation for $Q_1$. Therefore, parallel completeness transfers from $Q_1$ to $Q_2$.

The mapping provided is thus indeed a polynomial reduction. □

For conjunctive queries, or unions thereof, transfer of parallel completeness thus has a higher complexity than parallel completeness for a single, given policy: in the polynomial hierarchy, we jump from $\Pi_2^p$ to $\Pi_3^p$. However, there are cases where transfer can be decided more efficiently. One such case is formed by strongly minimal queries, which we address next.

### 4.2.2. Strongly minimal queries

The need for a third level of (universally quantified) extra input in Algorithm $\mathbb{A}^{\text{trans}}$ for PCOMP-T($\text{UCQ}^{\neq}$) arises from the minimality test for valuations for $Q_1$. Accordingly, for a strongly minimal query $Q_1$—where every valuation is guaranteed to be minimal—, a decrease of the worst-case complexity can be expected. Somewhat surprisingly, the problem drops not by one but even two levels in the polynomial hierarchy.

**Proposition* 4.2.8.** PComp-T($\mathsf{UCQ}_{[\mathsf{sm}]},\mathsf{UCQ}$) is NP-complete, and NP-hard already if inputs are restricted to queries from CQ.

Two aspects contribute to this decrease in complexity. First, for strongly minimal queries, parallel-completeness transfer is equivalent to a weaker form. Second, the weaker form of parallel-completeness transfer can be characterised by a relatively simple structural relationship between queries. More formally, the weaker form of transfer is defined as follows.

**Definition* 4.2.9 (Weak parallel-completeness transfer).** Let $Q_1$ and $Q_2$ be queries from $\mathsf{UCQ}^{\neq}$. Parallel completeness *transfers weakly from* $Q_1$ *to* $Q_2$ if the following implication holds for every policy $\mathbb{P}$: if $Q_1$ is strongly parallel-complete under $\mathbb{P}$, then $Q_2$ is parallel-complete under $\mathbb{P}$. ◀

Note the restriction to *strong* parallel completeness—which refers to the left-hand side of the implication (only). Weak parallel-completeness transfer, in turn, can be characterised by syntactical containment relationship, as in the next proposition.

**Proposition* 4.2.10.** For all queries $Q_1$ and $Q_2$ from CQ, parallel completeness transfers weakly from $Q_1$ to $Q_2$ if and only if there exists an endomorphism $\varepsilon$ and a substitution $\sigma$ where $\varepsilon(\mathsf{pos}(Q_2)) \subseteq \sigma(\mathsf{pos}(Q_1))$.

The upper bound for UCQs in Proposition 4.2.8 follows by a simple generalisation of the argument for CQs. Basically, for irredundant UCQs, endomorphisms and substitutions have to exist from every disjunct of $Q_1$ to some disjunct of $Q_2$. For redundant UCQs, where the previous condition is sufficient but not necessary, an irredundant equivalent subset of disjuncts from $Q_1$ can be 'guessed' first, via homomorphisms.

However, the characterisation in Proposition 4.2.10 does not extend (directly) to queries with disequalities. For these, the complexity rises again by one level, as we prove below. The $\Pi_2^p$-lower bounds in Proposition 4.2.15 already hold for conjunctive queries (without union). Furthermore, they even hold when the use of disequalities is restricted to only one of the queries, either $Q_1$ or $Q_2$.—At least, after overcoming a technical obstacle, as described next.

Parallel-completeness transfer, as defined above, is a relatively strong condition. This is partially in accordance with the intention: whatever the global instance and the concrete policy are, does complete evaluation of $Q_1$ already guarantee complete evaluation of $Q_2$? There are some borderline cases however where only very restricted policies prevent transfer—policies that can be deemed impractical. These cases arise from an 'imbalanced use' of constants or disequalities between the queries, as demonstrated next.

Let us consider the use of constants first. A necessary condition for the transfer of parallel completeness from a query $Q_1$ to a query $Q_2$ is that $Q_1$ refers only to constants that also occur in $Q_2$. For instance, parallel completeness does *not* transfer

from the boolean query $Q_1 = H() \leftarrow R(c), S(x), T(d)$ to the boolean query $Q_2 = H() \leftarrow R(c), S(x)$. This is witnessed by policy $\mathbb{P} \stackrel{\text{def}}{=} \{(k_1, R(c)), (k_2, S(c)), (k_3, T(c))\}$ with universe $\mathsf{univ}(\mathbb{P}) = \{c\}$.

On the one hand, query $Q_1$ is trivially parallel-complete under $\mathbb{P}$ because there is no globally satisfying valuation over this universe. Query $Q_2$ on the other hand is *not* parallel-complete under $\mathbb{P}$ because the facts $R(c)$ and $S(c)$, required by the (unique) minimal valuation $\{x \mapsto c\}$, do not meet at any node.

The previous argument is easily generalised, and culminates in the following claim.

**Claim 4.2.11.** Let $Q_1$, $Q_2$ be queries from $\mathsf{UCQ}^{\neq}_{\text{dom}}$. If parallel completeness transfers from $Q_1$ to $Q_2$, then every constant in $Q_1$ also occurs in $Q_2$.

For queries with disequalities, the situation is similar. Parallel completeness does *not* transfer from the boolean query $Q_1 = H() \leftarrow R(x), S(x), R(y), x \neq y$ to the boolean query $Q_2 = H() \leftarrow R(x), S(x)$, as witnessed by policy $\mathbb{P} \stackrel{\text{def}}{=} \{(k_1, R(a)), (k_2, S(a))\}$ with universe $\mathsf{univ}(\mathbb{P}) = \{a\}$.

Since every consistent valuation for $Q_1$ refers to two data values, query $Q_1$ is trivially parallel complete under $\mathbb{P}$ because there is no globally satisfying valuation over this universe. Query $Q_2$, however, is *not* parallel-complete under $\mathbb{P}$ because facts $R(a)$ and $S(a)$, required by the minimal valuation $\{x \mapsto a\}$, do not meet at any node. However, for each policy $\mathbb{P}'$ with a universe that contains at least two data values, query $Q_2$ is parallel-complete under $\mathbb{P}'$ if so is query $Q_1$.

Again, the previous argument can be generalised.

**Claim 4.2.12.** Let $Q_1$ and $Q_2$ be queries from $\mathsf{UCQ}^{\neq}$. If parallel completeness transfers from $Q_1$ to $Q_2$, then every minimal valuation for $Q_2$ refers to at least as many data values that some minimal valuation for $Q_1$ refers to.

Note that Claim 4.2.12 refers to constant-free queries only. Since the main contribution of this work on parallel-completeness transfer for strongly minimal queries are the lower bounds stated in Proposition 4.2.15, which already hold for constant-free queries, we restrict our attention to these.

Contrary to the condition in Claim 4.2.11, it is not *a priori* clear how complex it is to determine algorithmically whether the Condition in Claim 4.2.12 is satisfied. We neglect this question here, because the motivation for this detour is to rule out cases where only 'impractical' policies contradict transfer.

In practice, the database is usually orders of magnitude larger than the query to be evaluated. Accordingly, the number of data values is usually greater than the number of variables of a query—the latter being a simple bound for the number of data values a valuation can refer to. This motivates the following definition.

**Definition 4.2.13 (Mild transfer).** Let $Q_1$ and $Q_2$ be queries from $\mathsf{UCQ}^{\neq}$ and let $s$ be the maximum number of variables in a disjunct of $Q_1$. Parallel correctness *transfers*

*mildly from* $Q_1$ *to* $Q_2$ if the following implication holds for every policy $\mathbb{P}$ with at least $s$ data values: if $Q_1$ is parallel-correct under $\mathbb{P}$, then $Q_2$ is parallel-correct under $\mathbb{P}$.
*Mild transfer of parallel soundness* and *parallel completeness* is defined analogously. ◀

Mild transfer of parallel completeness can be characterised in a similar way as the standard transfer. The only difference between Condition (PComp-T) and the condition below is that the valuations do not have to satisfy the domain restriction additionally required by covers (Definition 4.2.2).

**Condition (mPComp-T)**

**Assumptions: Let $Q_1$ and $Q_2$ be queries from $\mathsf{UCQ}^{\neq}$.**
For every minimal valuation $V_2$ for $Q_2$,
there is a minimal valuation $V_1$ for $Q_1$ that is a fact-cover of $V_2$.

This condition is sufficient and necessary for mild transfer of parallel completeness.

**Proposition 4.2.14 (Characterisation of mild transfer).** For queries $Q_1, Q_2 \in \mathsf{UCQ}^{\neq}$, parallel completeness transfers mildly from $Q_1$ to $Q_2$ if and only if they satisfy Condition (mPComp-T).

The proof is a simple adaptation of the proof for unrestricted transfer and thus deferred to the appendix (Proposition A.5). We are now ready to state the hardness results for conjunctive queries *with disequalities* precisely.

**Proposition 4.2.15.** The following two problems are $\Pi_2^p$-complete.

1. mPCOMP-T($\mathsf{CQ}^{\neq}_{[\mathsf{sm}]}$,$\mathsf{CQ}$)
2. PCOMP-T($\mathsf{CQ}_{[\mathsf{sm}]}$,$\mathsf{CQ}^{\neq}$)

*Proof.* The upper bounds follow by a simple adaptation of Algorithm $\mathbb{A}^{\text{trans}}$ (page 77), ignoring the third level of (existentially) quantified extra input and the test related to valuation $W_1$. For mild transfer, another change is required: valuation $V_1$ is tested only of being not a *fact*-cover. The resulting algorithms are correct by Conditions (PComp-T) and (mPComp-T), respectively.

For the lower bounds, we provide polynomial reductions from $\Pi_2$-QBF to each of the variants of the parallel-completeness transfer problem. In both cases, a pair $(Q_1, Q_2)$ of queries is derived from an input formula $\varphi$ for $\Pi_2$-QBF. This formula is assumed to be of the form $\varphi = \forall \boldsymbol{x} \exists \boldsymbol{y} \psi(\boldsymbol{x}, \boldsymbol{y})$, where $\psi$ is a propositional formula in 3-CNF with clauses $C_1, \ldots, C_p$ over propositions $\boldsymbol{x} = (x_1, \ldots, y_r)$ and $\boldsymbol{y} = (y_1, \ldots, y_s)$, where $C_j = (\ell_{j,1} \vee \ell_{j,2} \vee \ell_{j,3})$ for each $j \in \{1, \ldots, p\}$.

Following the pattern of previous reductions, the queries refer to 'truth variables' $w_0, w_1$ and 'literal variables' $\boldsymbol{x} = (x_1, \ldots, x_r)$ and $\boldsymbol{y} = (y_1, \ldots, y_s)$ for positive literals as well as $\bar{\boldsymbol{x}} = (\bar{x}_1, \ldots, \bar{x}_r)$ and $\bar{\boldsymbol{y}} = (\bar{y}_1, \ldots, \bar{y}_s)$ for their negated counterparts.

We start with the reduction to mPComp-T($\mathsf{CQ}^{\neq}_{[\mathsf{sm}],\mathsf{CQ}}$), where disequalities are only allowed in the left-hand query, and later provide a reduction to PComp-T($\mathsf{CQ}_{[\mathsf{sm}],\mathsf{CQ}^{\neq}}$), where disequalities are only allowed in the right-hand query.

⋙ Hardness of mPComp-T($\mathsf{CQ}^{\neq}_{[\mathsf{sm}],\mathsf{CQ}}$). Apart from the already mentioned variables, query $Q_1$ refers to an additional variable $\alpha$. Query $Q_2$, which does not use disequalities, is defined by

$$\begin{aligned}\mathsf{head}(Q_2) &\stackrel{\text{def}}{=} H(w_0, w_1, \boldsymbol{x}, \bar{\boldsymbol{x}}),\\ \mathsf{pos}(Q_2) &\stackrel{\text{def}}{=} \mathcal{A}_{\text{fix}} \cup \mathcal{A}_{\boldsymbol{y}} \cup \mathcal{A}^{0,1}_{\boldsymbol{y}} \cup \mathcal{A}_{\text{sat}} \cup \mathcal{A}_{\psi}\end{aligned}$$

and query $Q_1$ is defined by

$$\begin{aligned}\mathsf{head}(Q_1) &\stackrel{\text{def}}{=} H(w_0, w_1, \boldsymbol{x}, \bar{\boldsymbol{x}}, \boldsymbol{y}, \bar{\boldsymbol{y}}, \alpha),\\ \mathsf{pos}(Q_1) &\stackrel{\text{def}}{=} \mathcal{A}_{\text{fix}} \cup \mathcal{A}_{\boldsymbol{y}} \cup \mathcal{A}^{0,1}_{\boldsymbol{y}} \cup \mathcal{A}_{\text{sat}} \cup \mathcal{A}^{\text{inv}}_{\text{sat}},\\ \mathsf{diseq}(Q_1) &\stackrel{\text{def}}{=} \{\alpha \neq w_0, \alpha \neq w_1\},\end{aligned}$$

using only two disequalities. Query $Q_1$ is strongly minimal because it is full.

The first three sets of atoms are similar to those used in the proof of Proposition 4.2.7, with relations $\mathtt{XVal}_h$ and $\mathtt{YVal}_i$ being binary now however. More concretely, these sets of atoms are defined by

$$\begin{aligned}\mathcal{A}_{\text{fix}} &\stackrel{\text{def}}{=} \{\mathtt{False}(w_0), \mathtt{True}(w_1)\} \cup \{\mathtt{XVal}_h(x_h, \bar{x}_h) \mid h \in \{1, \ldots, r\}\},\\ \mathcal{A}_{\boldsymbol{y}} &\stackrel{\text{def}}{=} \{\mathtt{YVal}_i(y_i, \bar{y}_i), \mathtt{YVal}_i(\bar{y}_i, y_i) \mid i \in \{1, \ldots, s\}\} \text{ and}\\ \mathcal{A}^{0,1}_{\boldsymbol{y}} &\stackrel{\text{def}}{=} \{\mathtt{YVal}_i(w_0, w_1), \mathtt{YVal}_i(w_1, w_0) \mid i \in \{1, \ldots, s\}\}.\end{aligned}$$

The remaining atoms refer to relation symbols $\mathtt{C}_1, \ldots, \mathtt{C}_p$ of arity $5 + 2r$ and represent clauses under a given assignment for truth and literal variables. Set

$$\mathcal{A}_{\text{sat}} \stackrel{\text{def}}{=} \{\mathtt{C}_j(w_0, w_1, \boldsymbol{x}, \bar{\boldsymbol{x}}, \boldsymbol{w}) \mid \boldsymbol{w} \in \mathbb{W}^+, j \in \{1, \ldots, p\}\},$$

encodes satisfied clauses—where $\mathbb{W}^+$ consists of all triples of truth variables, except for $(w_0, w_0, w_0)$—, while set

$$\mathcal{A}_{\psi} \stackrel{\text{def}}{=} \{\mathtt{C}_j(w_0, w_1, \boldsymbol{x}, \bar{\boldsymbol{x}}, \ell_{j,1}, \ell_{j,2}, \ell_{j,3}) \mid j \in \{1, \ldots, p\}\}$$

represents the clauses of $\psi$. The last set $\mathcal{A}^{\text{inv}}_{\text{sat}}$ of atoms however does not resemble any set of atoms used before. It is specifically geared to deal with valuations for $Q_2$ that do *not* represent a truth assignment or do not do so *unambiguously*. This can happen for one out of three reasons. First, truth and false cannot be distinguished if $V_2(w_0) = V_2(w_1)$. Second, literals do not represent truth values if $V_2(x_i)$ or $V_2(\bar{x}_i)$

equals neither $V_2(w_0)$ nor $V_2(w_1)$. Third, complementary literals do not represent complementary values if $V_2(x_h) = V_2(\bar{x}_h)$ for some $h \in \{1, \dots, r\}$.

To facilitate notation, for a sequence $\boldsymbol{v}$ of variables and variables $u, v$, let $\boldsymbol{v}[v \mapsto u]$ denote the sequence where every occurrence of $v$ is replaced by $u$. Then, set

$$\begin{aligned}\mathcal{A}^{\text{inv}}_{\text{sat}} \stackrel{\text{def}}{=} &\{\mathtt{C}_j(w_0, w_0, \boldsymbol{x}, \bar{\boldsymbol{x}}, \ell_{j,1}, \ell_{j,2}, \ell_{j,3}) \mid j \in \{1, \dots, p\}\}\\ \cup &\{\mathtt{C}_j(w_0, w_1, \boldsymbol{x}[x_h \mapsto \alpha], \bar{\boldsymbol{x}}, \ell_{j,1}, \ell_{j,2}, \ell_{j,3}) \mid h \in \{1, \dots, r\}, j \in \{1, \dots, p\}\}\\ \cup &\{\mathtt{C}_j(w_0, w_1, \boldsymbol{x}, \bar{\boldsymbol{x}}[\bar{x}_h \mapsto \alpha], \ell_{j,1}, \ell_{j,2}, \ell_{j,3}) \mid h \in \{1, \dots, r\}, j \in \{1, \dots, p\}\}\\ \cup &\{\mathtt{C}_j(w_0, w_1, \boldsymbol{x}, \bar{\boldsymbol{x}}[\bar{x}_h \mapsto x_h], \ell_{j,1}, \ell_{j,2}, \ell_{j,3}) \mid h \in \{1, \dots, r\}, j \in \{1, \dots, p\}\}\end{aligned}$$

handles all three cases mentioned above.

The mapping from $\varphi$ to $(Q_1, Q_2)$ is clearly total and computable in polynomial time. Next, we show that it also satisfies the reduction property: $\varphi \in \Pi_2$-QBF holds if and only if $(Q_1, Q_2) \in$ mPComp-T($\mathsf{CQ}^{\neq[\mathsf{sm}],\mathsf{CQ}}$).

» If. For a proof by contraposition, assume that $\varphi \notin \Pi_2$-QBF. We show that parallel completeness then does not transfer mildly from $Q_1$ to $Q_2$. By our assumption, there exists a truth assignment $\beta_{\boldsymbol{x}}$ for $\boldsymbol{x}$ such that, for every truth assignment $\beta_{\boldsymbol{y}}$ for $\boldsymbol{y}$, the combined assignment $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$ does not satisfy $\psi$. Fix such an assignment $\beta_{\boldsymbol{x}}$. Then, fix a truth assignment $\beta_{\boldsymbol{y}}$ such that $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$ minimises the number of unsatisfied clauses in $\psi$.

These assignments induce a valuation $V_2$ for $Q_2$ that is minimal but lacking a fact-covering valuation for $Q_1$. Let $V_2$ map $(w_0, w_1)$ to $(0, 1)$ and $\pi$ to $(\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}})(\pi)$ and $\bar{\pi}$ to the corresponding complementary value for every proposition $\pi$.

First, valuation $V_2$ is minimal. Assume, towards a contradiction, that there exists a valuation $U_2$ for $Q_2$ such that $U_2 <_{Q_2} V_2$. Agreement on the head variables immediately implies that both valuations require the same facts for $\mathcal{A}_{\text{fix}} \cup \mathcal{A}^{0,1}_{\boldsymbol{y}} \cup \mathcal{A}_{\text{sat}}$. Furthermore, $U_2 <_{Q_2} V_2$ implies $U_2(\mathcal{A}_{\boldsymbol{y}} \cup \mathcal{A}^{0,1}_{\boldsymbol{y}}) \subseteq V_2(\mathcal{A}_{\boldsymbol{y}} \cup \mathcal{A}^{0,1}_{\boldsymbol{y}})$, which is contained in $V_2(\mathcal{A}^{0,1}_{\boldsymbol{y}})$ by definition of $V_2$. Valuation $U_2$ hence also describes truth assignment $\beta_{\boldsymbol{x}}$ on $\boldsymbol{x}$ and some assignment $\beta'_{\boldsymbol{y}}$ on $\boldsymbol{y}$, but such that $U_2$ requires *fewer* facts than $V_2$ for $\mathcal{A}_{\text{sat}} \cup \mathcal{A}_\psi$. This can only happen if $\beta_{\boldsymbol{x}} \cup \beta'_{\boldsymbol{y}}$ is unsatisfying for *fewer* clauses of $\psi$ than $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$, which contradicts the choice of assignment $\beta_{\boldsymbol{y}}$.

Second, there is no valuation $V_1$ for $Q_1$ that covers $V_2$. Every such valuation agrees with $V_2$ on variables $w_0, w_1$ and $\boldsymbol{x}, \bar{\boldsymbol{x}}$ because of the atoms in $\mathcal{A}_{\text{fix}}$. It also satisfies $V_2(\mathcal{A}_\psi) \subseteq V_1(\mathcal{A}_{\text{sat}} \cup \mathcal{A}^{\text{inv}}_{\text{sat}})$ because there are no other $\mathtt{C}_j$-facts in $Q_1$. Indeed, this implies even $V_2(\mathcal{A}_\psi) \subseteq V_1(\mathcal{A}_{\text{sat}})$ because valuation $V_1$ is boolean and thus no atom from $\mathcal{A}^{\text{inv}}_{\text{sat}}$ can cover an atom from $\mathcal{A}_\psi$. This, however, contradicts the definition of $V_2$, which is induced by the unsatisfying truth assignment $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$ for $\psi$.

Therefore, queries $Q_1$ and $Q_2$ do not satisfy Condition (PComp-T) and hence parallel completeness does not transfer mildly from $Q_1$ to $Q_2$.

» Only if. Assume that $\varphi \in \Pi_2$-QBF. We show that parallel completeness transfers mildly from $Q_1$ to $Q_2$ via Condition (mPComp-T). For that purpose, let $V_2$ be an arbitrary minimal valuation for $Q_2$. We distinguish three cases, depending on the mapping of the head variables, and show that in each of them, a fact-cover exists.

› 1. Case ($V_2(w_0) = V_2(w_1)$ or $V_2(x_h) = V_2(\bar{x}_h)$ for some $h \in \{1, \ldots, r\}$). Let $V_1$ be a valuation for $Q_1$ that agrees with $V_2$ on variables $w_0, w_1, \boldsymbol{x}, \bar{\boldsymbol{x}}, \boldsymbol{y}, \bar{\boldsymbol{y}}$ and that maps variable $\alpha$ to some fresh value. Because of the partial variable agreement, the mappings trivially require the same facts for $\mathcal{A}_{\text{fix}}$, $\mathcal{A}_{\boldsymbol{y}}$, $\mathcal{A}^{0,1}_{\boldsymbol{y}}$ and $\mathcal{A}_{\text{sat}}$. Therefore, to prove that $V_1$ is a fact-cover of $V_2$, it suffices to show that $V_2(\mathcal{A}_\psi) \subseteq V_1(\mathcal{A}_{\text{sat}} \cup \mathcal{A}^{\text{inv}}_{\text{sat}})$. Indeed, even $V_2(\mathcal{A}_\psi) \subseteq V_1(\mathcal{A}^{\text{inv}}_{\text{sat}})$ holds because every $\mathtt{C}_j$-fact in $V_2(\mathcal{A}_\psi)$ has, by the case assumption, value $V_1(w_0)$ in its first two positions or value $V_1(x_h)$ in the positions of $x_h$ and $\bar{x}_h$. In both cases, the fact is required by $V_1$ for $\mathcal{A}^{\text{inv}}_{\text{sat}}$ too: in the former case because of atom $\mathtt{C}_j(w_0, w_0, \boldsymbol{x}, \bar{\boldsymbol{x}}, \ell_{j,1}, \ell_{j,2}, \ell_{j,3})$, in the latter case because of atom $\mathtt{C}_j(w_0, w_1, \boldsymbol{x}, \bar{\boldsymbol{x}}[\bar{x}_h \mapsto x_h], \ell_{j,1}, \ell_{j,2}, \ell_{j,3})$.

For the remaining two cases, we implicitly assume $V_2(w_0) \neq V_2(w_1)$ and $V_2(x_h) \neq V_2(\bar{x}_h)$ for every $h \in \{1, \ldots, r\}$. More particularly, we assume that valuation $V_2$ maps variables $w_0, w_1$ to values $0, 1$, respectively (for other data values, the argument is analogous).

› 2. Case ($V_2(x_h, \bar{x}_h) \notin \{(0,1),(1,0)\}$ for some $h \in \{1, \ldots, r\}$). Fix an index $h$ that witnesses the case assumption. Then $x_h$ or $\bar{x}_h$ is mapped to a value different from 0 and 1 because $V_2(x_h, \bar{x}_h) \in \{(0,0),(1,1)\}$ is already handled in the first case. Let us assume the former—for the latter, we can argue analogously. We define valuation $V_1$ to agree with $V_2$ on all variables but $\alpha$, and to map $\alpha$ to $V_2(x_h)$. Note that this yields a consistent valuation since $V_1(\alpha) \neq 0 = V_1(w_0)$ and $V_1(\alpha) \neq 1 = V_1(w_1)$.

Furthermore, valuation $V_1$ is a fact-cover for $V_2$. Once more, the partial variable agreement readily implies that both valuations require the same facts for atoms in $\mathcal{A}_{\text{fix}}$, $\mathcal{A}_{\boldsymbol{y}}$, $\mathcal{A}^{0,1}_{\boldsymbol{y}}$ and $\mathcal{A}_{\text{sat}}$. Hence, it suffices to argue $V_2(\mathcal{A}_\psi) \subseteq V_1(\mathcal{A}^{\text{inv}}_{\text{sat}})$. This containment relationship clearly holds because every fact required for atom $\mathtt{C}_j(w_0, w_1, \boldsymbol{x}, \bar{\boldsymbol{x}}, \ell_{j,1}, \ell_{j,2}, \ell_{j,3})$ in $\mathcal{A}_\psi$ is also required for an atom in $\mathcal{A}^{\text{inv}}_{\text{sat}}$, namely for $\mathtt{C}_j(w_0, w_1, \boldsymbol{x}[x_h \mapsto \alpha], \bar{\boldsymbol{x}}, \ell_{j,1}, \ell_{j,2}, \ell_{j,3})$.

› 3. Case (otherwise). In this case, valuation $V_2$ induces a truth assignment $\beta_{\boldsymbol{x}}$ for $\boldsymbol{x}$, and it does so unambiguously. Since $\varphi \in \Pi_2$-QBF, there is an assignment $\beta_{\boldsymbol{y}}$ for $\boldsymbol{y}$ such that $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$ satisfies subformula $\psi$. Let $V_1$ be the valuation induced by $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$ such that it maps $\alpha$ to 2. We claim that $V_1$ is minimal and that it is a fact-cover for $V_2$. First, valuation $V_1$ is minimal because it requires only facts $V_1(\mathsf{pos}(Q)) \subseteq V_1(\mathcal{A}_{\text{fix}} \cup \mathcal{A}^{0,1}_{\boldsymbol{y}} \cup \mathcal{A}_{\text{sat}} \cup \mathcal{A}^{\text{inv}}_{\text{sat}})$ where $V_1(\mathcal{A}_{\text{fix}} \cup \mathcal{A}^{0,1}_{\boldsymbol{y}} \cup \mathcal{A}_{\text{sat}})$ are required by every valuation $W_1$ for $Q_1$ that

agrees with $V_1$ on the head variables and $W_1$ obviously cannot require fewer facts than $V_1$ for $\mathcal{A}^{\text{inv}}_{\text{sat}}$ because of variable $\alpha$. Second, valuation $V_1$ is a fact-cover for $V_2$ because $V_2(\mathsf{pos}(Q_2)) \subseteq V_2(\mathcal{A}_{\text{fix}} \cup \mathcal{A}^{0,1}_{\boldsymbol{y}} \cup \mathcal{A}_{\text{sat}})$ holds, as argued next, and the right set is equivalent to $V_1(\mathcal{A}_{\text{fix}} \cup \mathcal{A}^{0,1}_{\boldsymbol{y}} \cup \mathcal{A}_{\text{sat}})$ by agreement of $V_1$ and $V_2$ on the head variables of $Q_2$. The containment relationship $V_2(\mathsf{pos}(Q_2)) \subseteq V_2(\mathcal{A}_{\text{fix}} \cup \mathcal{A}^{0,1}_{\boldsymbol{y}} \cup \mathcal{A}_{\text{sat}})$ follows since otherwise $V_1$, interpreted as a valuation for $Q_2$, contradicts the assumed minimality of $V_2$.

This completes the case distinction. In every case, Condition (mPComp-T) is satisfied and, thus, parallel completeness transfers mildly from $Q_1$ to $Q_2$.

Therefore, the specified mapping is indeed a polynomial reduction.

⋙ Hardness of PComp-T($\mathsf{CQ}_{[\mathsf{sm}],\mathsf{CQ}^{\neq}}$). The mapping defined next maps an input $\varphi$ for $\Pi_2$-QBF to a pair $(Q_1, Q_2)$ of queries and works similar to the reduction above. Instead of a single additional variable $\alpha$, however, both queries refer now to two sequences $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r)$ and $\bar{\boldsymbol{\alpha}} = (\bar{\alpha}_1, \ldots, \bar{\alpha}_r)$ of new variables whose lengths equals the number $r$ of $x_i$-propositions. The intention of these variables is to capture mappings of $\boldsymbol{x}$ and $\bar{\boldsymbol{x}}$ that cannot be interpreted meaningfully as a truth assignment. Accordingly, additional unary relation symbols $\texttt{XInv}_h$ and $\overline{\texttt{XInv}}_h$ are used for every $h \in \{1, \ldots, r\}$.

Query $Q_2$ is defined as

$$\begin{aligned}
\mathsf{head}(Q_2) &\stackrel{\text{def}}{=} H(w_0, w_1, \boldsymbol{x}, \bar{\boldsymbol{x}}) \\
\mathsf{pos}(Q_2) &\stackrel{\text{def}}{=} \mathcal{A}_{\text{fix}} \cup \mathcal{A}^{\text{inv}}_{\boldsymbol{x}} \cup \mathcal{A}^{\text{inv}}_{\boldsymbol{\alpha}} \cup \mathcal{A}_{\boldsymbol{y}} \cup \mathcal{A}^{0,1}_{\boldsymbol{y}} \cup \mathcal{A}_{\text{sat}} \cup \mathcal{A}_{\psi} \\
\mathsf{diseq}(Q_2) &\stackrel{\text{def}}{=} \{\alpha_h \neq w_0, \bar{\alpha}_h \neq w_0, \alpha_h \neq w_1, \bar{\alpha}_h \neq w_1 \mid h \in \{1, \ldots, r\}\}
\end{aligned}$$

and query $Q_1$, which does not use disequalities, is defined as

$$\begin{aligned}
\mathsf{head}(Q_1) &\stackrel{\text{def}}{=} H(w_0, w_1, \boldsymbol{x}, \bar{\boldsymbol{x}}, \boldsymbol{\alpha}, \bar{\boldsymbol{\alpha}}), \\
\mathsf{pos}(Q_1) &\stackrel{\text{def}}{=} \mathcal{A}_{\text{fix}} \cup \mathcal{A}^{\text{inv}}_{0,1} \cup \mathcal{A}^{\text{inv}}_{\boldsymbol{\alpha}} \cup \mathcal{A}_{\boldsymbol{y}} \cup \mathcal{A}^{0,1}_{\boldsymbol{y}} \cup \mathcal{A}_{\text{sat}} \cup \mathcal{A}^{\text{inv}}_{\text{sat}}
\end{aligned}$$

with the following sets of atoms. Sets

$$\begin{aligned}
\mathcal{A}^{\text{inv}}_{\boldsymbol{x}} &\stackrel{\text{def}}{=} \{\texttt{XInv}_h(x_h), \overline{\texttt{XInv}}_h(\bar{x}_h) \mid h \in \{1, \ldots, r\}\}, \\
\mathcal{A}^{\text{inv}}_{\boldsymbol{\alpha}} &\stackrel{\text{def}}{=} \{\texttt{XInv}_h(\alpha_h), \overline{\texttt{XInv}}_h(\bar{\alpha}_h) \mid h \in \{1, \ldots, r\}\}, \text{ and} \\
\mathcal{A}^{\text{inv}}_{0,1} &\stackrel{\text{def}}{=} \{\texttt{XInv}_h(w_0), \overline{\texttt{XInv}}_h(w_0), \texttt{XInv}_h(w_1), \overline{\texttt{XInv}}_h(w_1) \mid h \in \{1, \ldots, r\}\}
\end{aligned}$$

are used with the following intentions. The first two sets in combination with the disequalities, and set $\mathcal{A}^{\text{inv}}_{\text{sat}}$ of atoms defined below, enforce that a minimal valuation $V_2$ for $Q_2$ maps variable $\alpha_h$ to the same value as $x_h$ *if the latter does not encode a truth value*, $V_2(x_h) \notin \{V_2(w_0), V_2(w_1)\}$. In this case, only a mapping that maps $x_h$ and $\alpha_h$ identically does not require any additional fact, compared with the facts required for $\mathcal{A}^{\text{inv}}_{0,1}$ and $\mathcal{A}_{\text{sat}}$, while a different mapping does require at least one additional fact.

For instance, for a mapping $V_2$ with $V_2(w_0, w_1, x_1, \alpha_1) = (0, 1, 2, 3)$, four $\texttt{XVal}_1$-facts are required. The argument for $\bar{\alpha}_h$ and $\bar{x}_h$ is analogous.

All other sets of atoms are defined as in the previous reduction, with the exception of $\mathcal{A}^{\text{inv}}_{\text{sat}}$, which is altered such that each literal variable $x_h$ and $\bar{x}_h$ is replaced by a distinct variable $\alpha_h$ and $\bar{\alpha}_h$, respectively.

$$\begin{aligned}\mathcal{A}^{\text{inv}}_{\text{sat}} \stackrel{\text{def}}{=} &\{\mathtt{C}_j(w_0, w_0, \boldsymbol{x}, \bar{\boldsymbol{x}}, \ell_{j,1}, \ell_{j,2}, \ell_{j,3}) \mid j \in \{1, \dots, p\}\}\\ \cup &\{\mathtt{C}_j(w_0, w_1, \boldsymbol{x}[x_h \mapsto \alpha_h], \bar{\boldsymbol{x}}, \ell_{j,1}, \ell_{j,2}, \ell_{j,3}) \mid j \in \{1, \dots, p\}, h \in \{1, \dots, r\}\}\\ \cup &\{\mathtt{C}_j(w_0, w_1, \boldsymbol{x}, \bar{\boldsymbol{x}}[\bar{x}_h \mapsto \bar{\alpha}_h], \ell_{j,1}, \ell_{j,2}, \ell_{j,3}) \mid j \in \{1, \dots, p\}, h \in \{1, \dots, r\}\}\\ \cup &\{\mathtt{C}_j(w_0, w_1, \boldsymbol{x}, \bar{\boldsymbol{x}}[\bar{x}_h \mapsto x_h], \ell_{j,1}, \ell_{j,2}, \ell_{j,3}) \mid j \in \{1, \dots, p\}, h \in \{1, \dots, r\}\}\end{aligned}$$

Query $Q_1$ is strongly minimal because it is full. Furthermore, both queries have identical sets of variables and thus every valuation for one of them is also a valuation for the other.

Again it is clear that this mapping is total and computable in polynomial time. In the remainder of this proof, we hence show that it satisfies the reduction property: $\varphi \in \Pi_2$-QBF holds if and only if $(Q_1, Q_2) \in \text{PCOMP-T}(\mathsf{CQ}_{[\mathsf{sm}],\mathsf{CQ}^{\neq}})$.

» If. Assume $\varphi \notin \Pi_2$-QBF towards a proof by contraposition. Then, there is a truth assignment $\beta_{\boldsymbol{x}}$ for $\boldsymbol{x}$ such that no assignment $\beta_{\boldsymbol{y}}$ for $\boldsymbol{y}$ satisfies subformula $\psi$ in combination with $\beta_{\boldsymbol{x}}$. Let $\beta_{\boldsymbol{y}}$ be a truth assignment such that $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$ minimises the number of unsatisfied clauses of $\psi$.

Let $V_2$ be the valuation for $Q_2$ induced by this combined assignment $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$ and the equalities $V_2(\alpha_h) = V_2(\bar{\alpha}_h) = 2$ for every $h \in \{1, \dots, r\}$. Valuation $V_2$ is minimal but has no cover $V_1$ for $Q_1$, as argued next. Therefore, Condition (PComp-T) is violated and parallel completeness does *not* transfer from $Q_1$ to $Q_2$.

First, valuation $V_2$ is minimal. Assume, towards a contradiction, that there is a valuation $U_2$ for $Q_2$ such that $U_2 <_{Q_2} V_2$. Then, agreement of the head variables ensures that both valuations require the same facts for the atoms in $\mathcal{A}_{\text{fix}} \cup \mathcal{A}^{\text{inv}}_{\boldsymbol{x}} \cup \mathcal{A}^{0,1}_{\boldsymbol{y}} \cup \mathcal{A}_{\text{sat}}$. They also require the same facts for $\mathcal{A}^{\text{inv}}_{\boldsymbol{\alpha}}$ because $V_2$ maps all variables in $\boldsymbol{\alpha}$ and $\bar{\boldsymbol{\alpha}}$ to the same value 2 and $V_1$ also has to map them to a value different from 0 and 1. Furthermore $U_2(\mathcal{A}_{\boldsymbol{y}}) \subseteq V_2(\mathcal{A}_{\boldsymbol{y}} \cup \mathcal{A}^{0,1}_{\boldsymbol{y}})$ ensures that $U_2$ encodes a truth assignment $\beta'_{\boldsymbol{x},\boldsymbol{y}}$ on $\boldsymbol{x}$ and $\boldsymbol{y}$ that agrees with $\beta_{\boldsymbol{x}}$ on $\boldsymbol{x}$. Hence, we can infer $U_2(\mathcal{A}_{\text{sat}} \cup \mathcal{A}_{\psi}) \subsetneq V_2(\mathcal{A}_{\text{sat}} \cup \mathcal{A}_{\psi})$ and that that $\beta'_{\boldsymbol{x},\boldsymbol{y}}$ is unsatisfying for a strict subset of the clauses unsatisfied by $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$, which contradicts the choice of $\beta_{\boldsymbol{x}}$ and $\beta_{\boldsymbol{y}}$.

Second, valuation $V_2$ has no fact-covering valuation $V_1$ for $Q_1$. Assume, towards a contradiction, that such a valuation $V_1$ exists. Then, $V_2(\mathsf{pos}(Q_2)) \subseteq V_1(\mathsf{pos}(Q_1))$ implies $V_2(\mathcal{A}_{\text{fix}}) \subseteq V_1(\mathcal{A}_{\text{fix}})$ such that $V_1$ encodes the same truth assignment on $\boldsymbol{x}$. Furthermore, $V_2(\mathcal{A}^{\text{inv}}_{\boldsymbol{x}} \cup \mathcal{A}^{\text{inv}}_{\boldsymbol{\alpha}}) \subseteq V_1(\mathcal{A}^{\text{inv}}_{0,1} \cup \mathcal{A}^{\text{inv}}_{\boldsymbol{\alpha}})$ holds because there are no other sets of atoms with $\texttt{XInv}_h$- and $\overline{\texttt{XInv}}_h$-facts. This, in turn,

guarantees that $V_1$ also maps all variables in $\boldsymbol{\alpha}$ and $\bar{\boldsymbol{\alpha}}$ to 2. Then, $V_1(\mathcal{A}^{\text{inv}}_{\text{sat}})$ is disjoint from $V_2(\mathcal{A}_\psi)$ and the latter has to be covered by $V_1(\mathcal{A}_{\text{sat}})$, which proves $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$ to be satisfying for $\psi$ and thus contradicts the choice of $\beta_{\boldsymbol{x}}$.

Hence, there is no minimal fact-covering valuation $V_1$ for $Q_1$ for valuation $V_2$, which is minimal for $Q_2$, and Condition (PComp-T) is violated.

» Only if. For a direct proof, assume now $\varphi \in \Pi_2$-QBF. We show that each minimal valuation for $Q_2$ has a cover. Indeed, we show that every $V_2$ for $Q_2$ covers *itself*, seen as a valuation $V_1 = V_2$ for $Q_1$. These valuations clearly obey the domain condition since both queries refer to the same variables. Furthermore, valuation $V_1$ is minimal since query $Q_1$ is strongly minimal. In the remainder of this proof, we hence argue that $V_1$ is a fact-cover for $V_2$. For this, we can immediately neglect the facts required by valuation $V_2$ for the sets $\mathcal{A}_{\text{fix}}$, $\mathcal{A}^{\text{inv}}_{\boldsymbol{\alpha}}$, $\mathcal{A}_{\boldsymbol{y}}$, $\mathcal{A}^{0,1}_{\boldsymbol{y}}$ and $\mathcal{A}_{\text{sat}}$ of atoms common to both queries. Accordingly, it suffices to prove $V_2(\mathcal{A}^{\text{inv}}_{\boldsymbol{x}} \cup \mathcal{A}_\psi) \subseteq V_1(\mathsf{pos}(Q_1))$.

Let $V_2$ be an arbitrary minimal valuation for $Q_2$. For ease of description, let $V_1$ denote the *same* mapping, associated with $Q_1$. We distinguish three cases.

› 1. Case ($V_2(w_0) = V_2(w_1)$ or $V_2(x_h) = V_2(\bar{x}_h)$ for some $h \in \{1, \dots, r\}$). First, we can conclude that $V_2(\mathcal{A}^{\text{inv}}_{\boldsymbol{x}}) \subseteq V_1(\mathcal{A}^{\text{inv}}_{0,1} \cup \mathcal{A}^{\text{inv}}_{\boldsymbol{\alpha}})$ because, as mentioned above, every minimal valuation for $Q_2$ maps variable $\alpha_h$ identical to variable $x_h$ if $V_2(x_h)$ is different from $V_2(w_0)$ and $V_2(w_1)$. Thus, every fact $V_2(\mathtt{XInv}_h(x_h))$ required for $\mathcal{A}^{\text{inv}}_{\boldsymbol{x}}$ equals one of the facts $V_1(\mathtt{XInv}_h(w_0))$, $V_1(\mathtt{XInv}_h(w_1))$ or $V_1(\mathtt{XInv}_h(\alpha_h))$ required for $\mathcal{A}^{\text{inv}}_{0,1} \cup \mathcal{A}^{\text{inv}}_{\boldsymbol{\alpha}}$. The argument for fact $V_2(\overline{\mathtt{XInv}}_h(\bar{x}_h))$ is analogous. Second, $V_2(\mathcal{A}_\psi) \subseteq V_1(\mathcal{A}^{\text{inv}}_{\text{sat}})$ holds by an argument analogous to that in the first case for the previous reduction, using variable $\alpha_h$ instead of $\alpha$.

Once more, we implicitly assume $V_2(w_0) = 0 \neq 1 = V_2(w_1)$ and $V_2(x_h) \neq V_2(\bar{x}_h)$ for every $h \in \{1, \dots, r\}$ in the remaining cases.

› 2. Case ($V_2(x_h, \bar{x}_h) \notin \{(0,1),(1,0)\}$). By the case assumption, valuation $V_2$ maps $x_h$ or $\bar{x}_h$ to a value different from 0 and 1. Without loss of generality, we assume $V_2(x_h) \notin \{0,1\}$. Minimality of $V_2$ then implies $V_2(x_h) = V_2(\alpha_h)$. Containment $V_2(\mathcal{A}^{\text{inv}}_{\boldsymbol{x}}) \subseteq V_1(\mathcal{A}^{\text{inv}}_{0,1} \cup \mathcal{A}^{\text{inv}}_{\boldsymbol{\alpha}})$ can be argued as above. Furthermore, $V_2(\mathcal{A}_\psi) \subseteq V_1(\mathcal{A}^{\text{inv}}_{\text{sat}})$ holds because of atoms of the form $\mathtt{C}_j(w_0, w_1, \boldsymbol{x}[x_h \mapsto \alpha_h], \bar{\boldsymbol{x}}, \ell_{j,1}, \ell_{j,2}, \ell_{j,3})$ in $\mathcal{A}^{\text{inv}}_{\text{sat}}$.

› 3. Case (otherwise). Valuation $V_2$ now induces unambiguously a truth assignment $\beta_{\boldsymbol{x}}$ on $\boldsymbol{x}$. For this, an assignment $\beta_{\boldsymbol{y}}$ on $\boldsymbol{y}$ exists such that $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$ satisfies subformula $\psi$ because $\varphi \in \Pi_2$-QBF. The case assumption directly implies that $V_2(x_h, \bar{x}_h)$ equals either $(0,1)$ or $(1,0)$ and this, in turn, ensures $V_2(\mathcal{A}^{\text{inv}}_{\boldsymbol{x}}) \subseteq V_1(\mathcal{A}^{\text{inv}}_{0,1})$. Furthermore, since $V_2$ is minimal for $Q_2$, it satisfies $V_2(\mathcal{A}_\psi) \subseteq V_1(\mathcal{A}_{\text{sat}})$, as shown by the following contradiction. If containment

does not hold, then a valuation $W_2$ that agrees with $V_2$ on all variables but $y_1, \ldots, y_s, \bar{y}_1, \ldots, \bar{y}_s$ and represents $\beta_{\boldsymbol{y}}$ as above, requires strictly fewer $\mathtt{C}_j$-facts, namely only those from $W_2(\mathcal{A}_{\text{sat}})$. Therefore, valuation $V_1$ satisfies also $V_2(\mathcal{A}_\psi) = V_1(\mathcal{A}_{\text{sat}}) \subseteq V_1(\mathsf{pos}(Q_1))$ and is a fact-cover for $V_2$ indeed.

This completes the case distinction and shows that Condition (PComp-T) is satisfied. Therefore, parallel completeness transfers from $Q_1$ to $Q_2$.

As claimed, the given mapping is a polynomial reduction. □

This concludes our investigation of parallel-completeness (and parallel-correctness) transfer for monotonic queries from $\mathsf{UCQ}^{\neq}$. In the next section, we address non-monotonic queries and, as a consequence, also parallel-soundness transfer.

## 4.3. Unions of conjunctive queries with negation

So far, basically all published results on the transfer of parallel-correctness deal with monotonic query classes—and thus neglect the transfer of parallel-soundness in particular. The work on parallel-correctness transfer under bag semantics [KNV18] can be partly seen as an exception, as discussed in Section 4.4.

In this section, we use the characterisation of parallel-soundness for *polarised* queries via Condition (PSound-pol), presented in Section 3.3.3, to establish a characterisation of parallel-soundness *transfer* between such queries.[1]

Parallel-completeness transfer for monotonic queries from $\mathsf{UCQ}^{\neq}$ has been characterised by Condition (PComp-T). This condition demands that, for every minimal valuation $V_2$ for $Q_2$, there is a minimal valuation $V_1$ for $Q_1$ that *covers* $V_2$.

Parallel-soundness transfer for the larger class of polarised queries from $\mathsf{UCQ}^{\neg,\neq}$, can be characterised based on the similar notion of *guarding* valuations. This notion differs from that of covering valuations (Definition 4.2.2) in two aspects. First, this notion refers not only to positive but also to negated atoms. Second, a valuation can require *multiple* guards.

**Definition 4.3.1 (Guarding valuations).** Let $Q$ and $Q'$ be queries from $\mathsf{UCQ}^{\neg,\neq}$. A valuation $V$ for $Q$ is *guarded* by valuations $V'_1, \ldots, V'_r$ for $Q'$ if the following conditions are satisfied.

1. *Requirement condition:* $V'_1, \ldots, V'_r$ require only facts required by $V$, that is, $V'_1(\mathsf{pos}(Q')), \ldots, V'_r(\mathsf{pos}(Q')) \subseteq V(\mathsf{pos}(Q))$.
2. *Prohibition condition:* $V'_1, \ldots, V'_r$ together prohibit all facts prohibited by $V$, that is, $V(\mathsf{neg}(Q)) \subseteq V'_1(\mathsf{neg}(Q')) \cup \ldots \cup V'_r(\mathsf{neg}(Q'))$. ◀

[1] For parallel-completeness transfer between polarised queries and for all variants of transfer between queries with negation in general, it remains open to determine the complexity.

We remark that—different from covers—there is no condition on the domain for guards.

**Example 4.3.2 (Guarding valuations).** Consider the following two polarised queries.

$$Q_1 = H(x, y) \leftarrow R(x), R(y), R(u), \neg S(u)$$
$$Q_2 = H(x, y) \leftarrow R(x), R(y), \neg S(x), \neg S(y)$$

Then, valuation $V_2 \stackrel{\text{def}}{=} \{x \mapsto 1, y \mapsto 2\}$ for $Q_2$ is guarded by *two* valuations for $Q_1$, namely $V_{1,1} \stackrel{\text{def}}{=} \{x \mapsto 1, y \mapsto 2, u \mapsto 1\}$ and $V_{1,2} \stackrel{\text{def}}{=} \{x \mapsto 1, y \mapsto 2, u \mapsto 2\}$ because the requirement condition is satisfied,

$$V_{1,1}(\mathsf{pos}(Q_1)) = V_{1,2}(\mathsf{pos}(Q_1)) = \{R(1), R(2)\} \subseteq \{R(1), R(2)\} = V_2(\mathsf{pos}(Q_2)),$$

as is the prohibition condition,

$$V_2(\mathsf{neg}(Q_2)) = \{S(1), S(2)\} \subseteq \{S(1)\} \cup \{S(2)\} = V_{1,1}(\mathsf{neg}(Q_1)) \cup V_{1,2}(\mathsf{neg}(Q_1)).$$

However, valuation $V_2$ is not guarded by a *single* valuation for $Q_1$ because it prohibits two facts, while every valuation for $Q_1$ prohibits a single fact only. Note also that all three valuations are minimal for the respective queries.

The next proposition shows that parallel soundness transfers from $Q_1$ to $Q_2$. ■

Exchanging the notion of a 'cover' for that of a 'guard' in Condition (PComp-T) yields the following condition.

**Condition (PSound-T-pol)**

**Assumptions: Let $Q_1$ and $Q_2$ be polarised queries from $\mathsf{UCQ}^{\neg,\neq}$.**
For every minimal valuation $V_2$ for $Q_2$,
there are minimal valuations $V_{1,1}, \dots V_{1,r}$ for $Q_1$ that guard $V_2$.

This condition is necessary and sufficient for transfer of parallel-soundness between polarised queries, as stated next.

**Proposition 4.3.3 (Characterisation of parallel soundness-transfer).** Let $Q_1$ and $Q_2$ be arbitrary polarised queries from $\mathsf{UCQ}^{\neg,\neq}$. Parallel soundness transfers from query $Q_1$ to query $Q_2$ if and only if Condition (PSound-T-pol) is satisfied for $Q_1$ and $Q_2$.

*Proof.* As usual, we prove both implications of the equivalence separately.

⋙ If. Let $Q_1$ and $Q_2$ be polarised queries that satisfy Condition (PSound-T-pol). We show that parallel soundness transfers from $Q_1$ to $Q_2$.

To this end, let $\mathbb{P}$ be an arbitrary policy such that query $Q_1$ is parallel-sound under $\mathbb{P}$. Furthermore, let $V_2$ be a minimal valuation for $Q_2$ and let $k \in \mathsf{net}(\mathbb{P})$ be an arbitrary node that is responsible for all facts required by $V_2$. If there is no such node, then this valuation is conform with Condition (PSound-pol). Otherwise, it suffices to show that this node is also responsible for the facts prohibited by $V_2$.

By Condition (PSound-T-pol), there are minimal valuations $V_{1,1}, \ldots, V_{1,r}$ that guard valuation $V_2$. This implies that $V_{1,i}(\mathsf{pos}(Q_1)) \subseteq V_2(\mathsf{pos}(Q_2)) \subseteq \mathbb{P}(k)$ for each $i \in \{1, \ldots, r\}$. Since $Q_1$ is parallel-sound under $\mathbb{P}$, Condition (PSound-pol) ensures that node $k$ is responsible for all facts prohibited by valuations $V_{1,1}, \ldots, V_{1,r}$. Thus, guardedness also implies $V_2(\mathsf{neg}(Q_2)) \subseteq V_{1,1}(\mathsf{neg}(Q_1)), \ldots, V_{1,r}(\mathsf{neg}(Q_1)) \subseteq \mathbb{P}(k)$. Therefore, query $Q_2$ and policy $\mathbb{P}$ satisfy Condition (PSound-pol). In particular, query $Q_2$ is parallel-sound under $\mathbb{P}$ and parallel soundness transfers from $Q_1$ to $Q_2$.

⋙ Only if. For a proof by contraposition, assume that arbitrary queries $Q_1$ and $Q_2$ do not satisfy Condition (PSound-T-pol). We show that then parallel soundness does not transfer from $Q_1$ to $Q_2$.

Let $V_2$ be a minimal valuation for query $Q_2$ that has no guards for query $Q_1$. Then, if $V_{1,1}, \ldots, V_{1,r}$ are all minimal valuations for $Q_1$ that only require facts required by $V_2$ for $Q_2$, there is a fact $f \in V_2(\mathsf{neg}(Q_2))$ that is not contained in $V_{1,1}(\mathsf{neg}(Q_1)) \cup \ldots \cup V_{1,r}(\mathsf{neg}(Q_1))$ because valuation $V_2$ has no minimal guards. Note that $r = 0$ is possible and $r \in \mathbb{N}_0$ can be assumed by our restriction to safe negation in queries (at some point, additional valuations do not prohibit additional facts). Based on this fact $f$, we define a single-node policy

$$\mathbb{P} \stackrel{\text{def}}{=} \{k\} \times \Big(V_2(\mathsf{pos}(Q_2)) \cup (\mathsf{facts}(\mathcal{S}^-, U) - \{f\})\Big),$$

where $U$ is the set of data values referred to by $V_2$. Policy $\mathbb{P}$ is designed such that query $Q_1$ is parallel-sound under $\mathbb{P}$ but query $Q_2$ is not.

First, by assumption, valuations $V_{1,1}, \ldots, V_{1,r}$ are the only minimal valuations for $Q_1$ such that node $k$ is responsible for the facts they require. Furthermore, by definition of policy $\mathbb{P}$, the single node $k$ is responsible for the facts prohibited by the minimal valuations among them, because none of them prohibits fact $f$. Hence, by Condition (PSound-pol), query $Q_1$ is parallel-sound under $\mathbb{P}$.

Second, query $Q_2$ is not parallel-sound under $\mathbb{P}$. This is witnessed by the global instance $\mathcal{G} \stackrel{\text{def}}{=} \mathcal{G}^+ \cup \mathcal{G}^-$, comprising $\mathcal{G}^+ \stackrel{\text{def}}{=} V_2(\mathsf{pos}(Q_2))$ and

$$\mathcal{G}^- \stackrel{\text{def}}{=} (\mathsf{facts}(\mathcal{S}^-, U) - V_2(\mathsf{neg}(Q_2))) \cup \{f\},$$

which contains all facts over the negative schema that are not prohibited by $V_2$, with the exception of fact $f$. Obviously, valuation $V_2$ satisfies query $Q_2$ on the local instance of node $k$ because this node is not responsible for the only prohibited fact $f$ that is also contained in $\mathcal{G}$. However, there is no valuation $W_2$ that derives the same

fact as $V_2$ on the global instance because this implies $W_2(\mathsf{pos}(Q_2)) \subseteq V_2(\mathsf{pos}(Q_2))$ and $W_2(\mathsf{neg}(Q_2)) \subseteq V_2(\mathsf{neg}(Q_2)) - \{f\} \subsetneq V_2(\mathsf{neg}(Q_2))$ by polarisation and thus contradicts minimality of $V_2$.

Therefore, there is a policy $\mathbb{P}$ such that query $Q_1$ is parallel-sound under $\mathbb{P}$ but query $Q_2$ is not. In particular, parallel soundness does not transfer from $Q_1$ to $Q_2$.

Hence, parallel soundness transfers from $Q_1$ to $Q_2$ if and only if the queries satisfy Condition (PSound-T-pol). □

Note that Condition (PSound-T-pol) does *not* require both queries to adhere to the same bipartition of the underlying schema. If parallel-soundness transfers from $Q_1$ to $Q_2$, then this implies $\mathcal{S}^+(Q_1) \subseteq \mathcal{S}^+(Q_2)$ and $\mathcal{S}^-(Q_2) \subseteq \mathcal{S}^-(Q_1)$. This is also satisfied for queries $Q_1 \stackrel{\text{def}}{=} H(x) \leftarrow R(x), \neg S(x), \neg T(x)$ and $Q_2 \stackrel{\text{def}}{=} H(x) \leftarrow R(x), S(x), \neg T(x)$, where relation $S$ belongs to $\mathcal{S}^-(Q_1)$ and to $\mathcal{S}^+(Q_2)$, respectively, and parallel soundness transfers from $Q_1$ to $Q_2$.

We conclude by determining the complexity of the associated decision problem.

**Theorem 4.3.4.** Decision problem PSound-T($\mathcal{Q}$) is $\Pi_3^p$-complete for every query class $\mathcal{Q} \in \{\mathsf{CQ}^\neg[\mathsf{pol}], \mathsf{CQ}^{\neg,\neq}[\mathsf{pol}], \mathsf{UCQ}^\neg[\mathsf{pol}], \mathsf{UCQ}^{\neg,\neq}[\mathsf{pol}]\}$.

*Proof.* The upper bound follows by a simple adaptation of Algorithm $\mathbb{A}^{\text{trans}}$ where the test for the *cover* property is replaced by the test for *guardedness*. The correctness of this algorithm can be argued by a modified version of Lemma 4.2.5.

The lower bound follows from Proposition A.6 in the appendix, which is similar to the $\Pi_3^p$-hardness result for the transfer of parallel *completeness* (Proposition 4.2.7). □

## 4.4. Related work and bibliographical remarks

Just like for parallel correctness, it is possible to consider the *transfer* of parallel correctness in different settings by variation of the query language, the database model, the evaluation semantics and other parameters (cf. Section 3.4). So far, the focus of the literature seems to be on parallel correctness not on its transfer. However, to the best of the author's knowledge, two variants of transfer have been studied—with results as summarised next.

**Hypercube policies [AGK+17a].** In Section 4.1, we argued that transfer of parallel-completeness can be viewed as a generalisation of parallel completeness. More precisely, parallel completeness transfers from a query $Q_1$ to a query $Q_2$ if and only if query $Q_2$ is parallel-complete under the family $\mathcal{P}_{\text{comp}}(Q_1)$ of all policies under which $Q_1$ is parallel-complete. Hence, if query $Q_2$ is parallel-complete under a

subfamily $\mathcal{P} \subseteq \mathcal{P}_{\text{comp}}(Q_1)$, this can be considered as a *restricted* form of parallel-completeness transfer.

The hypercube algorithm, introduced by Afrati and Ullman [AU10], is a worst-case optimal algorithm for the distributed one-round evaluation of multi-way joins on skew-free relational databases in the MPC model [BKS17, BKS14]. Moreover, this algorithm serves as an important tool in more recent algorithms where the restrictions to one round and to skew-free databases are dropped while keeping worst-case load-optimality for cycle and chain queries or full binary queries, respectively [KBS16, KS17].

For a brief introduction of the hypercube principle, we refer to Section 5.1.3. Here, it suffices to say that this algorithm determines, for a given query $Q \in \mathsf{CQ}$, a distribution policy $\mathbb{H}_Q$ that describes a partition of all valuations for the query over some domain. The exact policy depends on the size of the network and a choice of (hash) mappings. Variation of these parameters therefore induces a family $\mathcal{H}(Q)$ of policies. Each of these policies guarantees *strong* parallel completeness of $Q$, that is, $\mathcal{H}(Q) \subseteq \mathcal{P}_{\text{comp}}(Q)$ holds.

Due to their highly regular distribution pattern, transfer of parallel completeness *restricted to hypercube policies* is easier than for conjunctive queries in general: the problem is NP-complete [AGK+17a].

**Bag semantics [KNV18].** The initial results from Ketsman, Neven and Vandevoort on parallel correctness under bag semantics, as described in Section 3.4, are accompanied by the following results on the transfer of parallel correctness [KNV18]. It may be helpful to remember our remark in in Section 3.4 that parallel correctness under bag semantics does not only ask for an overapproximation as parallel correctness under set semantics but for an *exact* evaluation.

We already discussed that parallel correctness under bag semantics implies parallel correctness under set semantics but not the other way around. Thus, it may seem surprising on the first glance, that for *transfer* of parallel correctness, there is no implication in any direction—both notions are incomparable. The authors exemplify this by some queries from $\mathsf{CQ}^{\neq}$ [KNV18, Figure 1].

**Transfer under set semantics is incomparable to transfer under bag semantics.** Indeed, this can be argued already with simpler queries, even without disequalities. On the one hand, transfer under set semantics does not imply transfer under bag semantics. Query $Q = H() \leftarrow R(u)$ is parallel-complete (and thus parallel-correct) under set semantics for policy $\mathbb{P} = \{(k, R(a)), (\ell, R(a))\}$, while—the same query—$Q$ is not parallel-correct under bag semantics for $\mathbb{P}$ because the policy replicates the fact required by valuation $V = \{u \mapsto a\}$.
On the other hand, transfer under bag semantics does not imply transfer under set semantics. Under bag semantics, for a query like $Q_1 = H() \leftarrow R(u), R(v)$,

which is non-parallelisable, parallel-correctness transfers to query $Q_2 = H(x, y, z) \leftarrow R(x), R(y), R(z)$ because all $R$-facts have to be located on the same node. Under set semantics, however, parallel-correctness does not transfer from $Q_1$ to $Q_2$, as witnessed by policy $\mathbb{P} = \{(k, R(a)), (\ell, R(b))\}$, under which $Q_1$ is parallel-complete but $Q_2$ is not.

An exception is formed by the case where both queries $Q_1$ and $Q_2$ are strongly minimal and only non-replicating policies are considered—then, transfer under bag semantics and transfer under set semantics coincide.

As the non-parallelisability of queries like $Q_1$ under bag semantics indicates, the responsibility of some node $k$ for the facts required by a valuation $V$ for query $Q$ can imply the responsibility of this node for additional facts, dependent on the query. Let $\mathcal{F}_{Q,V}$ denote these 'implied' facts. Ketsman et al. characterise transfer of parallel correctness from $Q_1$ to $Q_2$ under bag semantics by the following condition [KNV18, Lemma 4.5].

**Condition (PC-T-bag)**

**Assumptions: Let $Q_1$ and $Q_2$ be queries from $\mathbf{CQ}^{\neq}$.**
For every valuation $V_2$ for $Q_2$, there is a valuation $V_1$ for $Q_1$
such that $V_2$ requires all facts required by $V_1$ but only such from $\mathcal{F}_{Q_1,V_1}$.

The first part ($V_2$ requires all facts required by $V_1$) ensures that the facts required for $V_2$ are not replicated and the second part ($V_2$ requires only facts from $\mathcal{F}_{Q,V}$) corresponds with the fact-cover property in Condition (PComp-T). Using this characterisation, an EXPTIME-upper bound has been provided (without matching lower bound), whose complexity is mainly determined by the computation of the implied facts.[2]

Motivated by the non-parallelisability of some queries under the naïve evaluation strategy, a more sophisticated strategy for the distributed evaluation is studied additionally. Here, the network is ordered and at most one node (the 'smallest') derives a fact for each valuation. In this setting, parallel correctness is characterised by Condition (PComp-naïve) and transfer is characterised by a similar variation of Condition (PComp-T), which extends the scope from minimal to arbitrary valuations.

**Bibliographical remarks.** *Transfer* of parallel correctness (Definition* 4.1.1) has been introduced by Ameloot et al. [AGK+15, AGK+17a] for conjunctive queries. The presentation here, which incorporates unions and disequalities, is a straightforward generalisation.

[2] In Section 5.1.3, we show that this upper bound can also be derived via a reduction to the implication problem for distribution dependencies.

While the characterisation of parallel-completeness transfer (Theorem* 4.2.3) has appeared earlier [AGK+15], the key notion of a cover (Definition* 4.2.2), including the—subtle, but important—domain condition, has been provided a little later [AGK+17a].

The $\Pi_3^p$-complexity bounds (Theorem 4.2.4) for CQs and UCQs in general are a result of the author. They have been published together with initial results (Propositions* 4.2.8 and 4.2.10) by his co-authors on the transfer from strongly minimal queries [AGK+15]. Later, the complexity analysis has been extended to strongly minimal queries *with disequalities*. The $\Pi_2^p$-completeness result (Proposition 4.2.15) is due to the author and has been published before. The original statement in the publication [AGK+17a, Proposition 5.21], which refers to *(standard)* transfer instead of *mild* transfer (Definition 4.2.13) is technically not correct for the case $(\mathsf{CQ}^{\neq}[\mathsf{sm}], \mathsf{CQ})$, as explained in Section 4.2.2 (the proof is basically the same however).

Section 4.3 exclusively contains previously unpublished results by the author.

# 5. Distribution dependencies and their implication problem

With parallel correctness and parallel-correctness transfer, we have studied static analysis problems that refer to the evaluation of queries over distributed relational data in Chapters 3 and 4. In the current chapter, we follow a more general approach to reason about the placement of distributed relational facts via constraints. To this end, we adapt the well-known formalisms of tuple- and equality-generating dependencies to account explicitly for *distributed* databases.

Our study focuses on the complexity of a core problem in the context of constraints: the *implication* problem. This problem asks, for a given set $\Sigma$ of constraints and a single constraint $\tau$,

$$\text{does every data set that satisfies } \Sigma \text{ also satisfy } \tau?$$

Although interesting in its own right, the implication problem gains importance also as a natural subproblem for other problems. This is particularly true for the 'chase' procedure that is commonly used to obtain upper bounds for the implication problem, which is also useful for tasks like query rewriting under views, the computation of solutions in data exchange or of certain answers in data integration settings.

**Structure of this chapter.** We start with a recapitulation of classical dependencies and their adaptation to *distribution* dependencies in Section 5.1, where we also discuss applications like the description of common partitioning schemes or the modelling of properties like parallel completeness. In Section 5.2, we study the implication problem for several variants of *data-full* distribution dependencies. Lastly, we provide an overview of related results in Section 5.3.

Two syntactic restrictions are the key to the complexity results in Chapter 5.2. Both restrictions are based on a separation of two types of variables, *node* variables and *data* variables—and the positions where they may occur. The first restriction, data-fullness, allows existential quantification only on node variables, which yields a fragment where the implication problem is decidable in exponential time. The second restriction bases on bounds for the arity of relations and the *context sizes* of node variables. Intuitively, the context size bounds the amount of data that can be referred to on a specific node. The number and position of occurrences of node

variables with unbounded context in distribution dependencies affects the complexity of the implication problem. The most restricted fragment is shown to be NP-complete in Section 5.2.2. Consecutive relaxations then lead to a PSPACE-complete fragment, discussed in Section 5.2.3, and several fragments that are EXPTIME-complete, in Section 5.2.4.

## 5.1. Definition

First, we recapitulate the necessary preliminaries for (classical) tuple- and equality-generating dependencies in Section 5.1.1. Then, in Section 5.1.2, we define variants of these dependencies that are specifically geared to the modeling of *distributed* relational data and, in Section 5.1.3, show how they can be used to capture common distribution patterns.

### 5.1.1. Tuple- and equality-generating dependencies

Tuple-generating and equality-generating dependencies are well-known formalisms for constraints on relational data. A *tuple-generating dependency (tgd)* $\sigma = \mathcal{A} \to \mathcal{A}'$ comprises two finite sets of atoms, its *body* and its *head*, denoted $\mathsf{body}(\sigma) \stackrel{\text{def}}{=} \mathcal{A}$ and $\mathsf{head}(\sigma) \stackrel{\text{def}}{=} \mathcal{A}'$, respectively. An instance $\mathcal{I}$ *satisfies* a tgd $\mathcal{A} \to \mathcal{A}'$ if, for every valuation $V$ for $\mathcal{A}$ where $V(\mathcal{A}) \subseteq \mathcal{I}$, there is an extension $V'$ to the variables in $\mathcal{A}'$ such that $V'(\mathcal{A}') \subseteq \mathcal{I}$. An *equality-generating dependency (egd)* $\sigma = \mathcal{A} \to (\xi = \eta)$ comprises a finite set of atoms and an equality term over variables in these atoms, its *body* and its *head*, denoted $\mathsf{body}(\sigma) \stackrel{\text{def}}{=} \mathcal{A}$ and $\mathsf{head}(\sigma) \stackrel{\text{def}}{=} (\xi = \eta)$, respectively. An instance $\mathcal{I}$ *satisfies* an egd $\mathcal{A} \to (\xi = \eta)$ if, for every valuation $V$ for $\mathcal{A}$ where $V(\mathcal{A}) \subseteq \mathcal{I}$, equality $V(\xi) = V(\eta)$ holds.

An instance $\mathcal{I}$ satisfies a set $\Sigma$ of dependencies if it satisfies every dependency in $\Sigma$.

The rule-based descriptions of tgds and egds represent first-order logical formulas of the form $\forall \boldsymbol{x}, \boldsymbol{y}(\varphi(\boldsymbol{x}, \boldsymbol{y}) \to \exists \boldsymbol{z} \psi(\boldsymbol{x}, \boldsymbol{z}))$, where $\varphi$ is a conjunction of (positive) relation atoms and $\psi$ is either, for tgds, also a conjunction of (positive) relation atoms or, for egds, an equality atom. The existentially quantified variables (often called *existential variables* for short) in a tgd $\mathcal{A} \to \mathcal{A}'$ are exactly those variables that occur in $\mathcal{A}'$ but not in $\mathcal{A}$. For example, the tgd $R(x, y), S(x) \to T(x, z_1), T(z_2, x)$ can be regarded as the formula $\forall x, y\big[(R(x, y) \wedge S(x)) \to \exists z_1, z_2 (T(x, z_1) \wedge T(z_2, x))\big]$. Following this correspondance, the usual notions of implication and equivalence from formulas carry over to depenencies. For two sets $\Sigma$ and $\Sigma'$ of dependencies, $\Sigma$ *implies* $\Sigma'$, denoted $\Sigma \models \Sigma'$, if every instance that satisfies $\Sigma$ also satisfies $\Sigma'$. For singleton sets $\Sigma' = \{\tau\}$, we also simply write $\Sigma \models \tau$. Furthermore, the sets are *equivalent*, denoted $\Sigma \equiv \Sigma'$, if they mutually imply each other.

**Remark 5.1.1.** In particular, a tgd $\mathcal{A} \to \mathcal{A}'$ with multiple atoms in its head $\mathcal{A}' = \{A_1, \ldots, A_n\}$ can be transformed into an equivalent set $\{\mathcal{A} \to A_1, \ldots, \mathcal{A} \to A_n\}$ of tgds—given that the head atoms refer to pairwise disjoint sets of existential variables. For example, the tgd $R(x, y), S(x) \to T(x, z_1), T(z_2, x)$ from above is equivalent to the set of tgds that consists of $R(x, y), S(x) \to T(x, z_1)$ and $R(x, y), S(x) \to T(z_2, x)$.

We can now define distribution dependencies as a syntactic extension of tuple- and equality-generating dependencies.

### 5.1.2. Distribution tuple- and equality-generating dependencies

Tuple- and equality-generating dependencies can be used to model constraints based on the equality of certain data values of facts. They allow, for instance, to model constraints like the following.

1. For each order, there exists an address record of the customer.
2. The address record is unique for each customer.

The intention behind distribution dependencies is to allow additionally to refer to the *location* of facts, as in the next examples.

1'. For each order, there exists an address record of the customer *on some node where the order is located too.*
2'. For each customer, all orders are located *on the same node.*

Our approach to cover constraints as these two is simple: relational atoms in tgds and egds can be annotated by node variables (inspired by the notation in Webdamlog[1]). Remember that, in Section 2.1, we have defined the set $\mathsf{var}$ of variables as the *disjoint* union of two sets $\mathsf{xvar}$ and $\mathsf{nvar}$ of data and node variables. This distinction is crucial in the following as it leads to 'semi-typed' dependencies: node variables cannot be compared or unified with data variables. Furthermore, each atom can be annotated by *at most one* node variable.

Eventually, the purpose of atoms (annotated or not) is to refer to facts under a given valuation. Next, we introduce a representation of distributed relational data by annotated facts and then turn to the annotation of atoms.

Let $\mathbb{D} = (\mathcal{G}, \mathbb{L})$ be a distributed database with global instance $\mathcal{G}$ and a distribution $\mathbb{L} \subseteq N \times \mathcal{G}$ over some network $N$. A *distributed fact* $f@k$ consists of a fact $f$ and a node $k$. We write $f@k \in \mathbb{D}$ if fact $f$ is located on node $k$, that is, if $f \in \mathbb{L}(k)$ holds. Similarly, for a fact $f$ (without annotation), we write $f \in \mathbb{D}$ if $f \in \mathcal{G}$. More generally, for a set $\mathcal{F} = \{f_1@k_1, \ldots, f_r@k_r, g_1, \ldots, g_s\}$, we write $\mathcal{F} \subseteq \mathbb{D}$ if $f_i@k_i \in \mathbb{D}$ and $g_j \in \mathbb{D}$ for every $i \in \{1, \ldots, r\}$ and every $j \in \{1, \ldots, s\}$.

[1] See Section 5.3 for more details on the relationship to Webdamlog.

Analogously, a *distributed atom* $A@\kappa$ consists of an atom $A$ (over data variables) and a term $\kappa$, which is either a node variable or a data value. The notion of valuations is extended canonically: a mapping is a valuation for a distributed atom or a set of atoms if it it preserves data values and if it maps all (data and node) variables to data values. Then, a valuation $V$ for a distributed atom $A@\kappa$ induces the distributed fact $V(A@\kappa) \stackrel{\text{def}}{=} V(A)@V(\kappa)$.

Node variables are usually denoted by Greek letters like $\kappa$, $\lambda$ and $\mu$ and nodes by Latin letters like $k$, $\ell$ and $m$. For brevity, $\mathcal{A}@\kappa$ denotes the set $\{A_1@\kappa, \ldots, A_r@\kappa\}$ of distributed atoms for a set $\mathcal{A} = \{A_1, \ldots, A_r\}$ of atoms and, analogously, $\mathcal{F}@k$ the set $\{f_1@k, \ldots, f_r@k\}$ of distributed facts

Distribution tgds and egds are basically defined like their classical counterparts, but allowed to contain also distributed atoms.

**Definition 5.1.2 (Distribution tgd).** A *distribution tuple-generating dependency (dtgd)* $\sigma = \mathcal{A} \to \mathcal{A}'$ comprises two sets $\mathcal{A}$ and $\mathcal{A}'$ of distributed or non-distributed atoms.

A distributed database $\mathbb{D}$ *satisfies* $\sigma$ if, for every valuation $V$ for $\mathcal{A}$ where $V(\mathcal{A}) \subseteq \mathbb{D}$, there is an extension $V'$ of $V$ to the variables in $\mathcal{A}'$ such that $V'(\mathcal{A}') \subseteq \mathbb{D}$. Then, $V'$ *witnesses* the satisfaction of $\sigma$ *relative to* $V$. If there is no such extension, then $V$ *witnesses* the *violation* of $\sigma$ in $\mathbb{D}$. ◀

The definition of degds follows the same lines. The equality atom may refer to data or to node variables, but only in a restricted fashion.

**Definition 5.1.3 (Distribution egd).** A *distribution equality-generating dependency (degd)* $\sigma = \mathcal{A} \to (\xi = \eta)$ comprises a set $\mathcal{A}$ of distributed or non-distributed atoms and variables $\xi, \eta$ in $\mathcal{A}$ that are either both data variables or both node variables. In the former case, the degd is *data-identifying*, in the latter case, it is *node-identifying*.

A distributed database $\mathbb{D}$ *satisfies* $\sigma$ if, for every valuation $V$ for $\mathcal{A}$ where $V(\mathcal{A}) \subseteq \mathbb{D}$, equality $V(\xi) = V(\eta)$ holds. Otherwise, $V$ *witnesses* the *violation* of $\sigma$ in $\mathbb{D}$. ◀

Syntactically, distribution tgds and egds form an extension of standard tgds and egds. However, the annotation by node variables is syntactic sugar only and distribution tgds and egds conversely induce a (strict) fragment of standard tgds and egds. This relationship is discussed in more detail in Section 5.3.

Before we turn to the complexity theoretical investigation of the implication problem, we illustrate the usefulness of distribution dependencies by a few examples.

**Example 5.1.4.** The constraints 1' and 2' mentioned above can be modeled by distribution tgds and distribution egds.

- Distribution tgd $\sigma = \texttt{Order}(i, c) \to \texttt{Order}(i, c)@\kappa, \texttt{Addr}(c, a)@\kappa$ models the constraint 'For each order, there exists an address record of the customer *on some node where the order is located too*.'

- Distribution egd $\tau = \texttt{Order}(i_1, c)@\kappa, \texttt{Order}(i_2, c)@\lambda \to (\kappa = \lambda)$ models the constraint 'For each customer, all orders are located *on the same node*.'

Consider a complete distribution $\mathbb{L}$ over two nodes with the following local instances.

$$\mathbb{L}(k_1) = \{\texttt{Order('Printer', 'John')}, \texttt{Addr('John'}, a_{\text{John}})\}$$
$$\mathbb{L}(k_2) = \{\texttt{Order('Paper', 'John')}, \texttt{Addr('John'}, a_{\text{John}})\}$$

The corresponding distributed database $\mathbb{D}$ satisfies dtgd $\sigma$ because, for both `Order`-facts, there exists a node where the order and the address of the customer is located. On the contrary, the database violates degd $\tau$ because orders for customer `'John'` are located on *different* nodes. ■

We conclude this section with a closer look on possible applications of distribution dependencies.

### 5.1.3. Applications of distribution tgds and egds

Distribution dependencies allow to model various common distribution patterns. They allow to model simple range and hash partitionings [ÖV11] but also more advanced patterns, like co-partitionings [DGS+90, FKT86], hierarchical partitionings [SVS+13, SCH+18], predicate-based reference partitionings [ZBS15] and partitionings that base on the hypercube principle [ABGA11, BKS17].

Below, we exemplify this for some cases. We neglect other cases, like range partitionings, because they are more naturally modelled using constants and comparison atoms like $\leq$, which have been studied too [GNS20] but which we do *not* consider in this thesis. Before, however, we explain how parallel correctness relates to distribution dependencies.

#### Parallel correctness

Each conjunctive query can be viewed as a *full* tuple-generating dependency—without existential variables in the head (which would lead to 'invented' values in derived facts). Therefore, it is not very surprising that distribution dependencies can model the derivation process, globally as well as locally.

We consider two approaches to capture parallel correctness via distribution dependencies and illustrate them for the same example query $Q = H(x, y) \leftarrow R(x, u), S(u, y)$ from CQ. Note that, as argued in Section 3.2, parallel correctness is equivalent to parallel completeness for monotonic queries, which we consider here.

In the first approach, we derive a single dtgd $\tau_Q$ from query $Q$ such that $Q$ is parallel-complete under a distributed database $\mathbb{D}$ if and only if $\mathbb{D}$ *satisfies* $\tau_Q$.

Generally, such a dependency is of the form $\tau_Q = \mathsf{body}(Q) \to h(\mathsf{body}(Q))@\kappa$ for some injective homomorphism that maps the existential variables of $Q$ to fresh

variables (and preserves the head variables). For our example query, this may yield dependency

$$\tau_Q = R(x,u), S(u,y) \to R(x,u')@\kappa, S(u',y)@\kappa,$$

via homomorphism $h = [u/u']$.

Indeed, database $\mathbb{D} = (\mathcal{G}, \mathbb{L})$ satisfies $\tau_Q$ if $Q$ is parallel-complete under $\mathbb{D}$, as the following argument shows (the converse direction is similarly simple). If there is a valuation $V$ where $V(\mathsf{body}(\tau_Q)) \subseteq \mathcal{G}$, then this valuation is, in particular, a satisfying valuation for $Q$ on $\mathcal{G}$ because $\mathsf{body}(Q) = \mathsf{body}(\tau_Q)$. Thus, parallel-completeness ensures that there exists a valuation $V'$ that derives the same fact on $\mathbb{L}(k)$ for some node $k$. Derivation of the same fact implies agreement of $V'$ and $V$ on the head variables $x$ and $y$ of $Q$ and local satisfaction implies $V'(\mathsf{body}(Q)) \subseteq \mathbb{L}(k)$. Finally, from $V'$ we can derive the extension $V'' \stackrel{\text{def}}{=} V[u' \mapsto V'(u), \kappa \mapsto k]$ of $V$ such that

$$V''(\mathsf{head}(\tau_Q)) = (V' \circ h^{-1} \circ h)(\mathsf{body}(Q))@k \subseteq \mathbb{D},$$

which proves that $\mathbb{D}$ satisfies $\tau_Q$.

We remark that the use of homomorphism $h$ is necessary in general—in particular, but not only, to account for the case where the globally satisfying valuation $V$ is not minimal. For full queries, however, it is obviously needless. In these cases, we can use $\tau_Q = \mathsf{body}(Q) \to \mathsf{body}(Q)@\kappa$, which is even a *data-full* dtgd (see Definition 5.2.2, below). For non-full CQs, the latter dependency asks only for *strong* parallel completeness instead (Definition 3.2.6).

Unfortunately, this first approach cannot be extended to *unions of* conjunctive queries. As Example 3.2.3 shows, parallel completeness of a UCQ cannot cannot be characterised by parallel completeness of its disjuncts: satisfaction of $\tau_{Q_1}$ and $\tau_{Q_2}$ under $\mathbb{D}$ implies parallel completeness of $Q = Q_1 \cup Q_2$ under $\mathbb{D}$ but it is *not necessary*.

The second approach derives a set $\Sigma_\mathbb{D} \cup \Sigma_Q$ and a dependency $\tau_Q$ from $\mathbb{D}$ and $Q$ such that $Q$ is parallel-complete under $\mathbb{D}$ if and only if $\Sigma_\mathbb{D} \cup \Sigma_Q$ *implies* $\tau_Q$.

The idea is to encode the database $\mathbb{D}$ in the set $\Sigma_\mathbb{D}$ and, furthermore, the global and the local derivations of facts for $Q$ in $\Sigma_Q$. For ease of description, we keep the transformation as direct as possible by allowing constants in dependencies.[2] Then, the database corresponds with a set of dtgds with empty body that state the unconditioned presence of facts. For instance, if a fact $f$ is in the local instance $\mathbb{L}(k)$ of some node $k$, then this yields a dtgd $\sigma_f$ with $\mathsf{body}(\sigma_f) = \emptyset$ and $\mathsf{head}(\sigma_f) = f@k$. Set $\Sigma_\mathbb{D}$ contains one such dependency for each global and each distributed fact in $\mathbb{D}$. Additionally, set $\Sigma_\mathbb{D}$ contains a dependency of the form $R(\boldsymbol{x})@\kappa \to R(\boldsymbol{x})$ for each relation $R$, which models the presence of all local facts in the global instance (in accordance with Definition 2.2.1).

[2] It is possible to avoid the use of constants by the use of additional variables.

Set $\Sigma_Q$ contains two dtgds, $\mathsf{body}(Q) \to \mathsf{head}(Q)$ and $\mathsf{body}(Q)@\kappa \to \mathsf{head}(Q)@\kappa$, which describe the global and local derivation of facts, respectively. We claim that $\Sigma_{\mathbb{D}} \cup \Sigma_Q$ implies $\tau = \mathsf{head}(Q) \to \mathsf{head}(Q)@\kappa$ if and only if query $Q$ is parallel-complete under $\mathbb{D}$.

More concretely, for our example query, set $\Sigma_Q$ contains $R(x,u), S(u,y) \to H(x,y)$ and $R(x,u)@\kappa, S(u,y)@\kappa \to H(x,y)@\kappa$ and we ask for implication of $H(x,y) \to H(x,y)@\kappa$, that is, whether each globally derived fact appears in some local instance too.

Notably, this approach also works for arbitrary *unions of* conjunctive queries, where two dependencies are added to $\Sigma_Q$ per disjunct.

Furthermore, it is possible to extend this approach from single distributed databases to rule-based *policies*. In that case, set $\Sigma_{\mathbb{D}}$ is replaced by a set $\Sigma_{\mathbb{P}}$. If $\mathbb{P}$ contains a rule of the form $R(\tau_1, \ldots, \tau_r) \to k$, then $\Sigma_{\mathbb{P}}$ contains a dependency of the form $R(\tau_1, \ldots, \tau_r) \to R(\tau_1, \ldots, \tau_r)@k$. Also, to restrict to valuations over the policy's universe, set $\Sigma_Q$ contains a dtgd with an empty premise and head $\mathtt{Dom}(a_1), \ldots, \mathtt{Dom}(a_s)$ for $\mathsf{univ}(\mathbb{P}) = \{a_1, \ldots, a_s\}$. Similarly, the bodies of the dependencies in $\Sigma_Q$ that describe the global and local derivation, respectively, are amended by an atom $\mathtt{Dom}(x)$ for each data variable $x$.

### Parallel correctness under bag semantics

In Sections 3.4 and 4.4, we discussed the work of Ketsman, Neven and Vandevoort on (transfer of) parallel correctness under bag semantics [KNV18]. When a query $Q$ is evaluated naïvely, parallel correctness demands that, for each valuation $V$ for $Q$, there is exactly one node responsible for the required facts, $V(\mathsf{pos}(Q))$. This condition is easily captured by two dependencies,

- a dtgd $\sigma' = \mathsf{body}(Q) \to \mathsf{body}(Q)@\kappa$,
  which demands that there is at least one responsible node; and
- a degd $\sigma'' = \mathsf{body}(Q)@\kappa, \mathsf{body}(Q)@\lambda \to (\kappa = \lambda)$,
  which demands that there is at most one such node.

Evidently, a conjunctive query $Q$ is parallel-correct on a distributed database that *satisfies* both $\sigma'$ and $\sigma''$, derived from $Q$. The same holds for rule-based policies.

Moreover, *implication* captures parallel-correctness *transfer*. If $\sigma'_1, \sigma''_1$ are derived from $Q_1$, as described above, and $\sigma'_2, \sigma''_2$ are analogously derived from $Q_2$, then parallel-correctness transfers from $Q_1$ to $Q_2$ (under bag semantics) if and only if $\{\sigma'_1, \sigma''_1\} \models \{\sigma'_2, \sigma''_2\}$ holds. In combination with Theorem 5.2.3, we can thus reconstruct the EXPTIME-upper bound provided originally [KNV18], at least for queries without disequalities.[3]

[3] For queries with disequalities (and other comparison atoms), this is also true, as the extended study on distribution dependencies shows [GNS20].

As we mentioned in Section 3.4, our example query is non-parallelisable: all facts required by some valuation have to be located on one node. This question can also be translated into an instance for the implication problem. We illustrate this for example query $Q = H(x, y) \leftarrow R(x, u), R(u, y)$. This query is non-parallelisable because set $\{\sigma', \sigma''\}$ implies degd

$$R(x_1, u_1)@\kappa_1, R(u_1, y_1)@\kappa_1, R(x_2, u_2)@\kappa_2, R(u_2, y_2)@\kappa_2 \to (\kappa_1 = \kappa_2),$$

which states that all pairs $(V_1, V_2)$ of valuations for $Q$ are located on the same node. Hence, we get an EXPTIME-upper bound via Theorem 5.2.3 for the 'parallelisability' problem too.

### Hierarchical partitionings

Attempting to provide performance *and* consistency under updates, some modern database systems, like Google's F1 [SCH+18, SVS+13], use hierarchical partitioning schemes. These schemes are a variant of *co-partitioning* schemes [SKN18], which have been introduced under the name *predicate-based reference partitioning* [ZBS15].

The idea is to hash partition a relation $S$ after the distribution of another relation $R$ according to the following two conditions.

- Every $S$-fact is located at some node.
- If an $S$-fact matches an $R$-fact on a predefined set of attributes, then this $S$-fact is located at every node with matching $R$-facts.

Distribution dependencies allow to model these conditions. For instance, the 'AdWords' example for F1 [SVS+13], which refers to customers, advertising campaigns and adword groups, can be modelled by the next two dependencies.

$$\sigma_1 = \texttt{Cust}(\text{cust}, \boldsymbol{x})@\kappa, \texttt{Camp}(\text{cust}, \text{ca}, \boldsymbol{y}) \to \texttt{Camp}(\text{cust}, \text{ca}, \boldsymbol{y})@\kappa,$$
$$\sigma_2 = \texttt{Camp}(\text{cust}, \text{ca}, \boldsymbol{y})@\kappa, \texttt{AdGrp}(\text{cust}, \text{ca}, \boldsymbol{z}) \to \texttt{AdGrp}(\text{cust}, \text{ca}, \boldsymbol{z})@\kappa$$

Moreover, distribution dependencies can be used to specify more elaborate co-hashing strategies since they allow multiple atoms (and, in particular, multiple relations) in their bodies. As an example, the distribution of `Camp`-facts, described by $\sigma_1$, could be conditioned on the existence of a supplier whose 'nation' key equals that of the customer.

$$\texttt{Cust}(\text{cust}, \text{nat}, \boldsymbol{x})@\kappa, \texttt{Camp}(\text{cust}, \text{ca}, \boldsymbol{y}), \texttt{Supp}(\text{s}, \text{nat}, \boldsymbol{z}) \to \texttt{Camp}(\text{cust}, \text{ca}, \boldsymbol{y})@\kappa$$

Analogously, distribution tgds do not require the head atom to occur in the body. Thus, a dependency like

$$\texttt{Cust}(\text{cust}, \text{nat}, \boldsymbol{x})@\kappa, \texttt{Camp}(\text{cust}, \text{ca}, \boldsymbol{y}) \to \texttt{Camp}^*(\text{cust}, \text{ca})@\kappa,$$

referring to a new relation $\texttt{Camp}^*$, can model the co-hashing of *derived facts*.

### Hypercube partitionings

We already mentioned the hypercube principle as an important building block for the distributed evaluation of multi-way joins [AU10, BKS14, KS17]. Now, we describe this principle by an example and show how essential properties can be modelled via distribution dependencies. Note that hypercube partitionings are an advanced type of hash-based partitionings. Accordingly, the modelling of simpler hash partitionings via distribution dependencies is also possible.

The 'triangle' query $Q \stackrel{\text{def}}{=} H(x,y,z) \leftarrow R(x,y), S(y,z), T(z,x)$ is evaluated by the Hypercube algorithm over a three-dimensional network $\{1,\ldots,p_1\} \times \{1,\ldots,p_2\} \times \{1,\ldots,p_3\}$ with $p_1 \cdot p_2 \cdot p_3$ servers. The responsibility of nodes for facts from the global instance is defined by hash functions $h_1, h_2, h_3$ for the dimensions. Each hash function $h_i : \mathsf{dom} \to \{1,\ldots,p_i\}$ maps a single data value $a$ to a slot $h_i(a)$ in the range $\{1,\ldots,p_i\}$ associated with the respective dimension.

The behaviour of a hypercube policy can be modeled by a distribution policy. We illustrate this for query $Q$ and a network where $p_1 = 6$, $p_2 = 4$ and $p_3 = 3$. Policy $\mathbb{H}_Q$ behaves as follows (assuming some fixed universe), see Figure 5.1 for a visualisation.

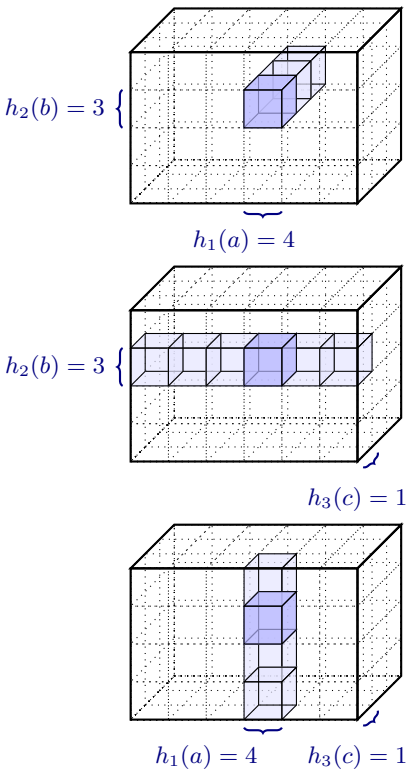


Figure 5.1.:
Hypercube policies arrange nodes conceptually in a multi-dimensional grid. Each dimension corresponds to a variable of the query to be computed, that is, to a join over the corresponding attributes.
If at all, facts are replicated in a structurally restricted fashion: along a line, a plane or a hyperplane. This figure illustrates the replication of facts $R(a,b)$, $S(b,c)$, $T(c,a)$, as required by a valuation for the triangle query, for some made-up choice of hash functions. Fact $R(a,b)$ is replicated along the third axis (top), fact $S(b,c)$ along the first axis (middle) and fact $T(c,a)$ along the second axis (bottom).
Note that all facts meet at the node with coordinate $(h_1(a), h_2(b), h_3(c)) = (4,3,1)$. Therefore the fact $H(a,b,c)$ can be derived locally, as desired.

- $\mathbb{H}_Q(R(a,b)) = \{h_1(a)\} \times \{h_2(b)\} \times \{1,2,3\}$ for all data values $a, b$
- $\mathbb{H}_Q(S(b,c)) = \{1,\ldots,6\} \times \{h_2(b)\} \times \{h_3(c)\}$ for all data values $b, c$
- $\mathbb{H}_Q(T(c,a)) = \{h_1(a)\} \times \{1,\ldots,4\} \times \{h_3(c)\}$ for all data values $a, c$

Regardless of the specific choice of hash functions, the hypercube principle ensures that query $Q$ is strongly parallel-complete under policy $\mathbb{H}_Q$, that is, for every—not necessarily minimal—valuation $V$ for $Q$, there is a node that is responsible for all facts required by $V$. In other words, the policy satisfies the following constraint.

$$\sigma_Q = R(x,y), S(y,z), T(z,x) \to R(x,y)@\kappa, S(y,z)@\kappa, T(z,x)@\kappa$$

Indeed, this dependency covers only *a part of* the hypercube properties; others are missing. For example, by Lemma 3.2.16 and Proposition 4.2.10, parallel completeness transfers from $Q$ to query $Q' \stackrel{\text{def}}{=} H'(x,y,z) \leftarrow R(x,y), S(y,z)$. Hence, policy $\mathbb{H}_Q$ also satisfies the next constraint.

$$\sigma_{Q'} \stackrel{\text{def}}{=} R(x,y), S(y,z) \to R(x,y)@\kappa, S(y,z)@\kappa$$

However, dependency $\sigma_{Q'}$ is not implied by dependency $\sigma_Q$ (because of possibly missing $T$-facts). Even worse, in general there can be additional queries that are parallel-complete under $\mathbb{H}_Q$ even though parallel-completeness does not transfer from $Q$ to them [GNS20].

A natural question that arises from these observations is, whether the properties of the hypercube principle can be modelled directly via distribution dependencies. The answer is positive if we abstract from the choice of hash functions. Technically, every choice of mappings $h_1, \ldots, h_d$ with signatures as above yields a hypercube with $d$ dimensions. Since these hash functions are often chosen randomly, it may seem appropriate to neglect the 'incidental' meeting of facts that is purely caused by collisions for a (specific) choice of hash functions, but rather consider only those meetings that are 'structural'. This idealisation leads to the equivalence $a_1 = a_2 \iff h_i(a_1) = h_i(a_2)$ for all data values $a_1, a_2$, while practically only the implication from left to right holds. This approach can be viewed as an 'abstract' infinite hypercube with a $d$-dimensional network $N = \mathsf{dom}^d$, where each hash function $h_i$ is the identity mapping.

In distribution dependencies, the hash values can then be represented as tuples in a $d$-ary relation $H$ such that a distributed fact $H(a_1, \ldots, a_d)@k$ marks node $k$ as responsible. An additional unary atom $\mathtt{Dom}$ is used to handle replication and to enforce completeness of the policy, that is, to prevent skipping of facts.

Given a conjunctive query $Q$, the following set $\Sigma_Q$ of dependencies describes the 'abstract' hypercube. First, for every relation $R$ in $Q$ with arity $n$, set $\Sigma_Q$ contains global dependencies $R(x_1, \ldots, x_n) \to \mathtt{Dom}(x_i)$ for every $i \in \{1, \ldots, n\}$ to

capture the (active) domain. Second, $\Sigma_Q$ contains a dtgd $\texttt{Dom}(x_1), \ldots, \texttt{Dom}(x_d) \to H(x_1, \ldots, x_d)@\kappa$ that ensures that there is at least one responsible node for each combination of data (or hash) values. Finally, there are dependencies that model the placement and replication of facts. We exemplify this for query $Q$ from above, where $\Sigma_Q$ contains three additional dependencies—one for each atom of the query.

$$R(x, y), \texttt{Dom}(z), H(x, y, z)@\kappa \to R(x, y)@\kappa$$
$$S(y, z), \texttt{Dom}(x), H(x, y, z)@\kappa \to S(y, z)@\kappa$$
$$T(z, x), \texttt{Dom}(y), H(x, y, z)@\kappa \to T(z, x)@\kappa$$

This set $\Sigma_Q$ captures the essence of hypercubes for query $Q$: it implies exactly those dependencies that are satisfied in *every* actual hypercube, independent of the size of the network and the choice of hash functions. For instance, using our example queries from above, set $\Sigma_Q$ implies dependency $\sigma_{Q'}$, whereas $\sigma_Q \not\models \sigma_{Q'}$.

Now that we have discussed some exemplary uses of distribution dependencies, we are ready to investigate the complexity of the implication problem for some fragments of them.

## 5.2. Complexity of implication

The implication problem is a fundamental problem in mathematical logic and theoretical computer science, since many problems can be reduced to it, and thus has attracted a lot of attention. In this section, we study the worst-case complexity of the implication problem for different fragments of distribution dependencies.[4]

**IMP($\mathcal{D}$)**

| | |
|---|---|
| **Parameters:** | dependency class $\mathcal{D}$ |
| **Input:** | finite set $\Sigma \subseteq \mathcal{D}$ of dependencies,<br>dtgd or degd $\tau$ |
| **Question:** | Does $\Sigma$ imply $\tau$? |

Note that dependency $\tau$ is *not* required to belong to the class $\mathcal{D}$ but may be an arbitrary distribution dependency, since this does not affect the following upper bounds. Indeed, we mostly consider a variant, $\text{IMP}_\alpha(\mathcal{D})$, of the implication problem, where the arity of the relations referred to by dependencies $\Sigma \cup \{\tau\}$ is at most $\alpha$, for a parameter $\alpha \in \mathbb{N}_0$.

[4] Technically, two variants can be distinguished: finite and unrestricted implication. Here, we study the *finite* implication problem because instances (databases) are finite.

Unfortunately, the implication problem for first-order formulas is undecidable and this is also true for the restricted form of implicational formulas underlying tgds and egds.

**Theorem* 5.2.1 ([BV81, CLM81]).** The implication problem is undecidable for untyped and typed embedded dependencies.

Although formulated for the more general 'embedded' dependencies, these results hold for (untyped and typed) tuple-generating dependencies too [FV84].

This negative result motivates the study of fragments of dependencies in an attempt to explore the boundaries of the implication problem's decidability and complexity.

It has been noted early that the use of existentially quantified variables has a significant influence on the decidablity and—in the positive case—the complexity of the implication problem. Therefore, various restrictions have been considered in the literature.

An obvious and rather strong restriction is to renounce the use of existential quantification completely. A tgd $\sigma = \mathcal{A} \to \mathcal{A}'$ without existentially quantified variables, that is, a tgd where $\mathsf{var}(\mathcal{A}') \subseteq \mathsf{var}(\mathcal{A})$ holds, is called *full*. Slightly less restrictive is the prohibition of existential quantification of data variables only. Notably, many examples of dtgds in Section 5.1.3 adhere to this restriction.

**Definition 5.2.2 (data-full dtgds).** A dtgd $\sigma = \mathcal{A} \to \mathcal{A}'$ is *data-full* if $\mathsf{xvar}(\mathcal{A}') \subseteq \mathsf{xvar}(\mathcal{A})$. The class of all data-full dtgds is denoted $\mathsf{DTGD}[\mathsf{df}]$. ◀

In the following, we study data-full dtgds and (arbitrary) degds. Since only node variables are quantified existentially, without loss of generality, we assume from now on that the head of each dtgd either contains no node variable, only one universal node variable or only one existential node variable, using a generalisation of Remark 5.1.1. This leads to the distinction of three types of data-full dtgds.

1. A dtgd is *global* if it has no node variable in the head.
2. A dtgd is *data-collecting* if it has a universal node variable in the head.
3. A dtgd is *node-generating* if it has an existential node variable in the head.

The first type comprises dtgds with distributed atoms, like $R(x), S_1(x)@\kappa, S_2(x)@\kappa \to T(x)$, and *purely global* dtgds without distributed atoms, like $R(x), S(x) \to T(x)$. The latter correspond to full classical tgds. For the first two types, using Remark 5.1.1 again, we may assume without restriction that the head consists of a single atom.

The restriction to data-full dtgds has the benefit of a decidable implication problem and thus allows the automatic reasoning.

**Theorem 5.2.3.** $\textsc{Imp}(\mathsf{DTGD}[\mathsf{df}] \cup \mathsf{DEGD})$ is EXPTIME-complete.

We defer the proof of this theorem to Section 5.2.4, where we show, in particular, that the lower bound already holds for schemas with maximally binary relations.

Before we turn to the corresponding lower bound, however, we focus on *upper* bounds. To this end, we describe how the chase procedure can be used on distribution dependencies to obtain algorithms for the implication problem.

### 5.2.1. The chase for distribution dependencies

The chase procedure is *the* algorithmic tool for deciding implication $\Sigma \models \tau$ on tgds and egds [BV84], with several other applications like query rewriting under views or the computation of solutions for data exchange scenarios.

Often, the chase is applied to instances with two types of data values, named 'constants' and 'labelled nulls'. However, for the implication problem, which we consider here, it is more straightforwardly applied to sets of *atoms*. To this end, we generalise our basic notions canonically.

- *Distributed databases (Definition 2.2.1).* A database $\mathbb{D} = (\mathcal{G}, \mathbb{L})$ consists of a finite set $\mathcal{G}$ of atoms and a finite relation $\mathbb{L}$ over nodes and atoms.
  In this sense, the original definition can be seen as *ground* databases.
- *Distributed dependencies (Definitions 5.1.2 and 5.1.3).* Satisfaction of dtgds and degds over sets of atoms is formulated with respect to arbitrary homomorphisms instead of valuations—as is their violation.

To decide implication of $\tau$ under $\Sigma$, the chase is roughly used in the following way. In a first step, we start with a database $\mathbb{D}$, induced by $\tau$, and check repeatedly whether a dependency in $\Sigma$ is violated. If no dependency is violated, then $\mathbb{D}$ clearly satisfies all of them; the procedure stops. Otherwise, the violation is considered more closely. If it is possible to repair $\mathbb{D}$ with respect to the violation, then the repair is applied and the procedure continues. If it is impossible to repair $\mathbb{D}$, the procedure fails. If the chase terminates, then it terminates either failing or it succeeding. If the chase succeeds, then $\mathbb{D}$ satisfies $\Sigma$ after (finitely many) repairs. In a second step, two cases are distinguished.

- If the chase fails, then dependency $\tau$ is trivially implied by $\Sigma$ because there is no database that contains facts corresponding to the initial atoms *and* satisfies $\Sigma$.
- If the chase succeeds, then dependency $\tau$ is implied by $\Sigma$ if and only if $\tau$ is satisfied on the repaired database $\mathbb{D}$.

Indeed, for the algorithms below, the process is a little simpler, as explained later. Nevertheless, the basic concepts remain the same and thus are defined next. We start with the formalisation of a 'repair', first for degds, then for dtgds.

**Definition* 5.2.4 (Application of degds).** Let $\mathbb{D}$ be a distributed database. A degd $\sigma = \mathcal{A} \to (\xi = \eta)$ is *applicable to* $\mathbb{D}$ *with homomorphism* $h$ if $h$ witnesses the violation of $\sigma$ and at least one of the terms $h(\xi)$ and $h(\eta)$ is a variable. The *application of* $(\sigma, h)$ yields a distributed database $\mathbb{D}'$ that results from $\mathbb{D}$ by replacing every occurrence of $h(\eta)$ by $h(\xi)$, if $h(\eta)$ is a variable, or by replacing every occurrence of $h(\xi)$ by $h(\eta)$, otherwise. ◀

**Example 5.2.5.** The degd $\sigma = R(x, y)@\kappa, R(x, z)@\lambda \to (y = z)$ states that the first attribute of relation $R$ is a key—irrespective of the location of these facts: If there are two $R$-facts with the same value in the first attribute, then their values in the second attribute have to be identical too. Dependency $\sigma$ is applicable, for instance, to the distributed database $\mathbb{D} = (\mathcal{G}, \mathbb{L})$ with a global instance $\mathcal{G} = \{R(1, y), R(1, 2)\}$ and distribution $\mathbb{L} = \{(k, R(1, y)), (\ell, R(1, 2))\}$. This is witnessed by homomorphism $h = \{x \mapsto 1, y \mapsto y, z \mapsto 2, \kappa \mapsto k, \lambda \mapsto \ell\}$ because $h(y) = y$ is a variable, while $h(z) = 2$ is not. Application of $(\sigma, h)$ on $\mathbb{D}$ yields the distributed database $\mathbb{D}' = (\mathcal{G}', \mathbb{L}')$ with $\mathcal{G} = \{R(1, 2)\}$ and $\mathbb{L} = \{(\kappa, R(1, 2)), (\lambda, R(1, 2))\}$. ■

For *tuple*-generating dependencies, we distinguish two types of applications, 'oblivious' and 'restricted' applications.

**Definition* 5.2.6 (Application of dtgds).** Let $\mathbb{D}$ be a distributed database. A dtgd $\sigma = \mathcal{A} \to \mathcal{A}'$ is *obliviously applicable to* $\mathbb{D}$ *with homomorphism* $h$ if $h(\mathcal{A}) \subseteq \mathbb{D}$ and $h$ maps existential variables in $\sigma$ to fresh variables not occurring in $\mathbb{D}$. It is *restrictedly applicable to* $\mathbb{D}$ *with* $h$ if, additionally, $h$ witnesses the violation of $\sigma$ on $\mathbb{D}$. The *application of* $(\sigma, h)$ yields a distributed database $\mathbb{D}'$ that results from $\mathbb{D}$ by addition of the facts $h(\mathcal{A}')$ to $\mathbb{D}$, that is, only to $\mathcal{G}$, if $\sigma$ is global, and otherwise to some local instance $\mathbb{L}(k)$ of the node $k$ targeted by $(\sigma, h)$. In the latter case, the fact is also added to $\mathcal{G}$ if not already present. ◀

Note that a restricted application $(\sigma, h)$ requires that there is no other homomorphism $h'$ that agrees with $h$ on the body variables of $\sigma$ and satisfies $h'(\mathsf{head}(\sigma)) \subseteq \mathbb{D}$. This has an influence on the complexity of algorithms (at least for non-full dtgds) that we discuss further below. Let us illustrate the application of a full dtgd first.

**Example 5.2.7.** Consider the distributed database $\mathbb{D} = (\mathcal{G}, \mathbb{L})$ with a global instance $\mathcal{G} = \{R(x, y), S(x), S(y)\}$ comprising three atoms with two of them distributed over local instances: $\mathbb{L} = \{(\kappa, S(x)), (\lambda, S(y))\}$.

There, the data-collecting dtgd $\sigma = R(u, v), S(v)@\mu \to R(u, v)@\mu$ is applicable to $\mathbb{D}$ via homomorphism $h = \{u \mapsto x, v \mapsto y, \mu \mapsto \lambda\}$. Application $(\sigma, h)$ yields the distributed database $\mathbb{D}' = (\mathcal{G}, \mathbb{L}')$ where $\mathbb{L}' = \{(\kappa, S(x)), (\lambda, S(y)), (\lambda, R(x, y))\}$ contains an additional atom $R(x, y)$ in the local instance of $\lambda$. In particular, application $(\sigma, h)$ is restricted. ■

Applicability corresponds with the need for a repair. The application of a repair is 'local' in the sense that it only refers to *one* example of a violated dependency—other repairs may be necessary. Moreover, the application of a repair may entail the need for further repairs. Therefore, we consider *sequences* of applications.

**Definition* 5.2.8 (Application sequence).** Let $\mathbb{D}_0$ be a set of distributed atoms and $\Sigma$ be a set of dtgds and degds. An *application sequence for* $\mathbb{D}_0$ *under* $\Sigma$ is a sequence $\boldsymbol{a} = (a_1, a_2, a_3, \dots)$ of successive applications: each $a_i = (\sigma_i, h_i)$ is an application for some $\sigma_i \in \Sigma$ to $\mathbb{D}_{i-1}$ that yields $\mathbb{D}_i$ for every index $i$. The sequence is *restricted* if it only contains restricted applications of dtgds. ◀

Within an application sequence $\boldsymbol{a}$, the combination of an application $a_i$ together with the previous and the resulting database, $\mathbb{D}_{i-1}$ and $\mathbb{D}_i$, is commonly called a *chase step*. In the following, we usually keep these intermediate databases implicit and refer to application sequences also as *chase sequences*—ignoring the subtle distinction.

**Definition* 5.2.9 (Chase result).** Let $\boldsymbol{a}$ be a finite application sequence $(a_1, \dots, a_n)$. The *chase result under* $\boldsymbol{a}$ is defined as $\mathsf{chase}_{\boldsymbol{a}}(\mathbb{D}_0) \stackrel{\text{def}}{=} \mathbb{D}_n$, the database resulting after the last chase step.

Sequence $\boldsymbol{a}$ is *successful* if the chase result satisfies $\Sigma$, that is, if no dtgd is restrictedly applicable and no degd is applicable. It is *failing* if the chase result violates a degd $\sigma = \mathcal{A} \to (\xi = \eta)$, witnessed by $h$, and $(\sigma, h)$ is not applicable to it—because both $h(\xi)$ and $h(\eta)$ are data values. A chase sequence that is either successful or failing is called *saturated*. Otherwise, it is *unsaturated*, meaning that further restricted applications are possible. ◀

The following example illustrates chase sequences.

**Example 5.2.10.** Let $\Sigma = \{\sigma_1, \sigma_2, \sigma_3\}$ be a set of distribution dependencies with

- $\sigma_1 = R(x, y) \to (x = y)$, a data-identifying degd;
- $\sigma_2 = R(x, x) \to S(x)$, a global dtgd; and
- $\sigma_2 = R(x, x), S(x) \to R(x, x)@\kappa, S(x)@\kappa$, a node-generating dtgd.

Consider the following chase sequence for a starting database $\mathbb{D}_0 = (\mathcal{G}_0, \mathbb{L}_0)$ with $\mathcal{G}_0 = \{R(x, y)\}$ and $\mathbb{L}_0 = \emptyset$.

1. Dependency $\sigma_1$ is applicable to $\mathbb{D}_0$ as witnessed by the identity homormorphism $\mathsf{id}$. Application $a_1 = (\sigma_1, \mathsf{id})$ results in the unification of $x$ and $y$ and thus in the distributed database $\mathbb{D}_1 = (\mathcal{G}_1, \mathbb{L}_1)$ with $\mathcal{G}_1 = \{R(x, x)\}$ and $\mathbb{L}_1 = \emptyset$.

2. Now, dependency $\sigma_2$ is applicable to $\mathbb{D}_1$, witnessed again by $\mathsf{id}$. Application $a_2 = (\sigma_2, \mathsf{id})$ results in the addition of the global atom $S(x)$ and thus in the distributed database $\mathbb{D}_2 = (\mathcal{G}_2, \mathbb{L}_2)$ with $\mathcal{G}_2 = \{R(x, x), S(x)\}$ and $\mathbb{L}_2 = \emptyset$.

3. Finally, dependency $\sigma_3$ is applicable to $\mathbb{D}_2$ via $\mathsf{id}$, where $\mathsf{id}(\kappa) = \kappa$ is fresh since it does not appear in $\mathbb{D}_2$. Application $a_3 = (\sigma_3, \mathsf{id})$ results in a new local instance, yielding $\mathbb{D}_3 = (\mathcal{G}_3, \mathbb{L}_3)$ with $\mathcal{G}_3 = \{R(x,x), S(x)\}$ and $\mathbb{L}_3 = \{(\kappa, R(x,x)), (\kappa, S(x))\}$.

No dependency in $\Sigma$ is restrictedly applicable to $\mathbb{D}_3$ and no degd is violated. Hence, the finite chase sequence $\boldsymbol{a} = (a_1, a_2, a_3)$ is successful. ■

In general, chase sequences may be infinite due to the introduction of fresh variables. Furthermore, since there is no order on the dependencies or homomorphisms to choose for applications or on the choice of fresh variables, different chase sequences may result from the same starting database under the same set of dependencies. This does not matter though for our purpose, basically because of the following theorem for *restricted* chase sequences on classical dependencies, which is easily extended to distribution dependencies (by the translation discussed in Section 5.3).

**Proposition* 5.2.11 (Properties of the restricted chase [FKMP05]).** Let $\mathbb{D}_0$ be a distributed database and let $\Sigma$ be a set of dtgds and degds.

1. If some finite chase sequence for $\Sigma$ starting from $\mathbb{D}_0$ is failing, then every saturated finite chase sequence for $\Sigma$ starting from $\mathbb{D}_0$ is failing.
2. The chase results $\mathbb{D} = \mathsf{chase}_{\boldsymbol{a}}(\mathbb{D}_0)$ and $\mathbb{D}' = \mathsf{chase}_{\boldsymbol{a}'}(\mathbb{D}_0)$ for arbitrary finite successful chase sequences $\boldsymbol{a}, \boldsymbol{a}' \in \mathsf{steps}_{\Sigma}(\mathbb{D}_0)$ are homomorphically equivalent.
3. Let $\mathbb{D}_n = \mathsf{chase}_{\boldsymbol{a}}(\mathbb{D}_0)$ be the chase result for a successful chase sequence $\boldsymbol{a}$ and let $\mathbb{D}$ be a database that satisfies $\Sigma$. If there exists a homomorphism $h_0$ from $\mathbb{D}_0$ to $\mathbb{D}$, then there exists a homomorphism $h_n$ from $\mathbb{D}_n$ to $\mathbb{D}$.

These properties guarantee that *each* saturated chase sequence faithfully represents *all* saturated chase sequences: either all of them are failing or all of them are successful. If they are successful, then they yield essentially the same results (with respect to their structure), a property that is called 'universality'.

Eventually, we want to use chase sequences to reason about the implication of a dependency $\tau$ under a set $\Sigma$ of dependencies. We start with an illustration of the approach.

**Example 5.2.12.** Consider again the set $\Sigma$ from Example 5.2.10 and the corresponding chase sequence $\boldsymbol{a} = (a_1, a_2, a_3)$. This sequence proves that $\Sigma$ implies the node-generating dtgd $\tau = R(x,y) \to R(x,y)@\kappa, S(y)@\kappa$, as we argue next.

The procedure starts with the *canonical database* associated with $\tau$, that is, $\mathbb{D}_0 = \mathsf{body}(\tau)$. The first dependency applied is the degd $\sigma_1$, which unifies variables $x$ and $y$. This unification is captured by the equality-generating homomorphism $\varepsilon$ that maps $y$ to $x$ and is the identity on all other variables in $\mathsf{body}(\tau)$.

In particular, degd $\sigma_1$ demands that $V(x) = V(y)$ holds for every valuation $V$ with $V(\mathsf{body}(\tau)) \subseteq \mathbb{D}$ on a database $\mathbb{D}$ *that satisfies* $\Sigma$ because $V(\mathsf{body}(\sigma_1)) \subseteq$

$V(\mathsf{body}(\tau))$. Thus, for all such databases, $\tau$ is satisfied if and only if $\tau' = R(x,x) \to R(x,x)@\kappa, S(x)@\kappa$ is satisfied. Notably, dependency $\tau'$ equals dependency $\varepsilon(\tau) = \varepsilon(\mathsf{body}(\tau)) \to \varepsilon(\mathsf{head}(\tau))$.

After the other applications, dependency $\tau'$ is 'certified' by $\mathsf{chase}_{\boldsymbol{a}}(\mathbb{D}_0) = \mathbb{D}_3$ because there is an extension $\varepsilon'$ of $\varepsilon$, which maps the existentially quantified variable $\kappa$ to itself, such that $\varepsilon'(\mathsf{head}(\tau')) \subseteq \mathbb{D}_3$. ■

In general, a chase sequence may induce several equality-generating homomorphisms, which affect the certification of $\tau$ combinedly. Given a finite chase sequence $\boldsymbol{a} = (a_1, \ldots, a_n)$ starting from a database $\mathbb{D}_0$ and yielding a database $\mathbb{D}_n$, consider exactly those applications $a_{i_1}, \ldots, a_{i_r}$ of degds that replace occurrences of (node or data) variables in $\mathbb{D}_0$. Let $\varepsilon_1, \ldots, \varepsilon_r$ be the homomorphisms underlying these replacements. They induce a homomorphism $\varepsilon(\boldsymbol{a}) \stackrel{\text{def}}{=} \varepsilon_r \circ \cdots \circ \varepsilon_1$ that describes the overall unification of the variables in $\mathbb{D}_0$.

**Definition 5.2.13 (Certifying sequence).** A finite application sequence $\boldsymbol{a}$ *certifies* $\tau$ if it starts from $\mathsf{body}(\tau)$ and yields a chase result $\mathbb{D}$ such that

- if $\tau$ is a degd $\mathcal{A} \to (\xi = \eta)$, then $\varepsilon(\xi) = \varepsilon(\eta)$, and
- if $\tau$ is a dtgd $\mathcal{A} \to \mathcal{A}'$, then there is an extension $\varepsilon'$ of $\varepsilon$ with $\varepsilon'(\mathcal{A}') \subseteq \mathbb{D}$,

for the homomorphism $\varepsilon = \varepsilon(\boldsymbol{a})$. ◀

We are now ready to relate chase sequences to the question of implied dependencies.

**Proposition 5.2.14 (Characterisation of implication).** For every set $\Sigma \cup \{\tau\}$ of data-full distribution dependencies, implication $\Sigma \models \tau$ holds if and only if there is a certifying application sequence for $\tau$ under $\Sigma$.

The proof is deferred to the appendix (Proposition A.7). Note that the characterisation requires the chase sequence neither to be saturated nor to be restricted. Both aspects are crucial for some of the upper bounds provided below.[5]

More particularly, we study different fragments of data-full distribution dependencies.

**Remark 5.2.15 (Constant-free dependencies).** In the following, we study implication only for distribution dependencies that do *not* contain constants.

The fragments are defined by syntactic criteria that are based on the following notion.

[5] The restricted chase was introduced by Fagin et al. [FKMP05] and is also known as the 'standard' chase [One12]. Other variants such as the 'oblivious' [CGK13] or 'semi-oblivious' chase [Mar09] have been studied to eliminate some of its weaknesses.

One weakness of the restricted chase is that the condition for the applicability of a dtgd $\sigma$ with existential variables with a valuation $V$ demands that there is no other valuation $V'$ that agrees with $V$ on $\mathsf{body}(\sigma)$, which is computationally more involved.

**Definition 5.2.16 (Context).** Let $\kappa$ be a node variable. For a set $\mathcal{A}$ of distributed atoms, the *$\kappa$-context* $\mathsf{cont}_\kappa(\mathcal{A})$ is the set of data variables that occur in a distributed atom $A@\kappa \in \mathcal{A}$. The *$\kappa$-context* of a dependency is the context of its relational atoms:

- for a dtgd $\sigma$, it is $\mathsf{cont}_\kappa(\sigma) \stackrel{\text{def}}{=} \mathsf{cont}_\kappa(\mathsf{body}(\sigma) \cup \mathsf{head}(\sigma))$; and
- for a degd $\sigma$, it is $\mathsf{cont}_\kappa(\sigma) \stackrel{\text{def}}{=} \mathsf{cont}_\kappa(\mathsf{body}(\sigma))$.

For $\beta \in \mathbb{N}_0$, a node variable $\kappa$ has *$\beta$-bounded (body) context* in a distributed dependency $\sigma$ if the context of (the body) of $\sigma$ contains at most $\beta$ variables. ◀

The motivation for this definition is to bound the amount of data per 'record', a subset that is distinguishable in some way in a set of atoms. In the simplest case, records are formed by atoms. Then, restricting the arity of relations means restricting the amount of data. However, this does not suffice anymore when fresh variables can be introduced—as with the application of node-generating dtgds. In this case, the fresh variable can serve as an identifier of a record spanning several atoms (and thus exceed possible arity bounds).

**Example 5.2.17.** Variable $\kappa$ has 2-bounded *body context* and 3-bounded *context* in dtgd $\sigma_1 = R(x, y)@\kappa, S(z)@\lambda \to T(x, z)@\kappa$ while neither $\kappa$ nor $\lambda$ have $\beta$-bounded context in dtgd $\sigma_2 = R_1(x_1)@\kappa, \ldots, R_{\beta+1}(x_{\beta+1})@\kappa \to S(x_1, \ldots, x_{\beta+1})@\lambda$ for any $\beta \in \mathbb{N}_0$. ■

The fragments of distribution dependencies studied in the next sections are defined by restricting the number of node variables with unbounded context in the body or the head, depending on whether the dependencies are node-generating or data-collecting dtgds, or degds. Table 5.1 provides an overview of the types of dtgds and degds that are considered more closely.

The upper bounds for our algorithms for the implication problem rely on the following bounds for chase sequences.

**Claim 5.2.18 (Bounds on chase sequences).** Let $\Sigma$ be a set of data-full dtgds and arbitrary degds and let $\mathbb{D}_0$ be an arbitrary distributed database. Every restricted chase sequence for $\mathbb{D}_0$ under $\Sigma$ is finite. More precisely,

1. the global set $\mathcal{G}$ of atoms and each local set of atoms in the intermediate chase results contains at most $(\|\Sigma\| + \|\mathbb{D}_0\|) \cdot \|\mathbb{D}_0\|^\alpha$ atoms; and
2. at most $\|\Sigma\| \cdot \|\mathbb{D}_0\|^{\|\Sigma\|}$ node variables are generated; and
3. the chase sequence has length at most $\mathcal{O}\left(\|\Sigma\|^2 \cdot \|\mathbb{D}_0\|^{\|\Sigma\|+\alpha}\right)$

if $\alpha$ is the maximum arity of relations in $\Sigma$.

| | | NP | PSPACE | EXPTIME | | | |
|---|---|---|---|---|---|---|---|
| (G1) | $\mu_1 \cdots \mu_r \to \kappa$ | ✓ | ✓ | | | | |
| (G2) | $\mu_1 \cdots \mu_r \to \kappa$ | ✓ | ✓ | ✓ | | ✓ | ✓ |
| (G3) | $\mu_1 \cdots \mu_r\ \lambda \to \kappa$ | | ✓ | | | | ✓ |
| (G4) | $\mu\ \lambda \to \kappa$ | | | | ✓ | | |
| (C1) | $\kappa\ \mu_1 \cdots \mu_r \to \kappa$ | ✓ | ✓ | | | | |
| (C2) | $\kappa\ \mu_1 \cdots \mu_r \to \kappa$ | ✓ | ✓ | | | | |
| (C3) | $\kappa\ \mu \to \kappa$ | | | ✓ | | | |
| (E1) | $\kappa\ \lambda\ \mu_1 \cdots \mu_r \to (\kappa = \lambda)$ | ✓ | ✓ | | | | |
| (E2) | $\kappa\ \lambda\ \mu_1 \cdots \mu_r \to (\kappa = \lambda)$ | ✓ | ✓ | | | | |
| (E3) | $\kappa\ \lambda \to (\kappa = \lambda)$ | | | | | | ✓ |
| (E4) | $\kappa\ \lambda\ \mu \to (\kappa = \lambda)$ | | | | | ✓ | |
| | Theorem | 5.2.20 | 5.2.23 | 5.2.25 | | | |

Table 5.1.: Overview of the fragments of data-full distribution dependencies studied in this section. For NP and PSPACE, each column indicates the allowed types of dtgds and degds and the corresponding complexity. For EXPTIME, on the contrary, each column indicates the types that are sufficient to yield hardness; allowing all types remains in EXPTIME.
Node variables that have to be bounded are shaded, others may be unbounded.

*Proof.* The application of data-full dtgds does not introduce new data variables. Thus, for each (intermediate) chase result, the number of data variables is bounded by the number $d$ of data variables in $\mathbb{D}_0$. For the following bounds, let $r$ denote the number of relation symbols that occur in $\Sigma$ or $\mathbb{D}_0$.

Property (1) holds because the global set $\mathcal{G}$ of atoms contains at most $rd^\alpha$ atoms over data variables from $\mathbb{D}_0$, and likewise for each local set of atoms. Obviously, both $r$ and $d$ are bounded by $\|\Sigma\|$ and $\|\mathbb{D}_0\|$, respectively, which yields the bound.

Property (2) follows from the next observation. In a restricted chase sequence, a new node is generated with an application $(\sigma, h)$ of a dtgd $\sigma = \mathcal{A} \to \mathcal{A}'@\kappa$ only if $h(\mathcal{A}')$ is not already present in the (intermediate) chase result. Since there are at most $\|\sigma\| \leq \|\Sigma\|$ variables in $\mathcal{A}'$ and each of these variables has to be mapped by $h$ to a data variable from $\mathbb{D}_0$, there are at most $d^{\|\Sigma\|}$ possible homomorphisms $h$ for each dependency in $\Sigma$. Therefore, the total number of such homomorphisms is bounded by $|\Sigma| \cdot d^{\|\Sigma\|} \leq \|\Sigma\| \cdot \|\mathbb{D}_0\|^{\|\Sigma\|}$.

Property (3) results from the former two bounds. By Property (2), a restricted chase sequence comprises at most $\|\Sigma\| \cdot d^{\|\sigma\|}$ applications of node-generating dtgds. For each of these nodes, at most $\|\Sigma\| \cdot \|\mathbb{D}_0\|^{\alpha}$ data-collecting dtgds can be applied by Property (1). The same bound holds for the number of applications of global dtgds. Furthermore, the number of applications (node- or data-identifying) degds doubles the number of applications at most. □

Additionally, every saturated chase is successful—it cannot be failing because there are no data values (that would have to be unified) because we assume dependencies to be constant-free, see Remark 5.2.15.

In Section 5.2.2, we show that the implication problem can be decided in non-deterministic polynomial time even for a fragment where saturated chase sequences may necessarily comprise an exponential number of nodes. This fragment is subsequently extended by an additional type of dtgds, for which we provide an algorithm that requires polynomial space in Section 5.2.3. Finally, we show, in Section 5.2.4, that basically all extensions by other types of dependencies make the implication problem EXPTIME-hard, even for a fixed arity bound.

### 5.2.2. NP-Fragment

We start with a relatively simple observation. When only node-generating dtgds with a $\beta$-*bounded* head node variable—Type (G1)—are present in a set $\Sigma$ of dependencies, among arbitrary data-collecting dtgds, global dtgds and degds, then the arity-bounded implication problem is easily seen to be in NP, for every fixed arity bound $\alpha$. First, the canonical database $\mathsf{body}(\tau)$ induced by $\tau$ provides at most $\|\tau\|$ nodes and data values. Second, the number of restricted applications of each node-generating of Type (G1) is bounded by $d^{\beta}$, where $d \leq \|\tau\|$ is the number of data values. Thus, the number of nodes increases only polynomially. Furthermore, other tuple-generating dependencies can be applied at most $rd^{\alpha}$ times to the global instance or to each local instance, respectively, where $r \leq \|\Sigma\|$ is the number of relation symbols. Along the lines of the proof of Claim 5.2.18, this gives a polynomial bound on each instance and, even, on the number of generated nodes variables.

Interestingly, the arity-bounded implication problem can also be decided by non-deterministic polynomial time algorithms if node-generating dtgds with *unbounded* head node variables are admitted. In that case, the body node variables have to be bounded and not all types of degds and data-collecting dtgds may be allowed. The following definition provides the exact conditions.

**Definition 5.2.19 (Bounded context).** Let $\beta \in \mathbb{N}_0$. The following types of dtgds and degds have $\beta$-*bounded context*.

- Node-generating dtgds where
  - (G1) the head node variable has $\beta$-bounded context or
  - (G2) all body node variables have $\beta$-bounded context.
- Data-collecting dtgds where
  - (C1) the head node variable has $\beta$-bounded body context or
  - (C2) all other node variables have $\beta$-bounded context.
- Data-identifying degds.
- Node-identifying degds of the form $\mathcal{A} \to (\kappa = \lambda)$ where
  - (E1) both $\kappa$ and $\lambda$ have $\beta$-bounded context or
  - (E2) all node variables except for $\kappa$ have $\beta$-bounded context.

Let $\mathsf{DTGD}_{[\mathsf{bc}(\beta)]}$ and $\mathsf{DEGD}_{[\mathsf{bc}(\beta)]}$ denote the class of all dtgds and degds with $\beta$-bounded context, respectively. ◀

**Theorem 5.2.20.** $\textsc{Imp}_{\alpha}\big(\mathsf{DTGD}_{[\mathsf{bc}(\beta)]} \cup \mathsf{DEGD}_{[\mathsf{bc}(\beta)]}\big)$ is in NP for fixed $\alpha, \beta \geq 0$. It is already NP-hard for $\alpha = 2$ and $\beta = 0$.

*Proof.* The lower bound follows by a simple reduction from the containment problem for conjunctive queries. For a pair $(Q_1, Q_2)$ of such queries, $Q_1 \sqsubseteq Q_2$ holds if and only if $\{\sigma_2\} \models \sigma_1$, where dependencies $\sigma_1$ and $\sigma_2$ are dtgds such that

$$\mathsf{body}(\sigma_i) \stackrel{\text{def}}{=} \mathsf{body}(Q_i) \text{ and } \mathsf{head}(\sigma_i) \stackrel{\text{def}}{=} \mathsf{head}(Q_i)$$

for both $i \in \{1, 2\}$. For instance, $Q_1 = H(x) \leftarrow R(x), S(x)$ and $Q_2 = H(x) \leftarrow R(x)$ satisfy $Q_1 \sqsubseteq Q_2$ and $\{R(x) \to H(x)\} \models R(x), S(x) \to H(x)$. The homomorphism $\mathsf{id}$ that witnesses containment—because $\mathsf{id}(\mathsf{body}(Q_2)) \subseteq \mathsf{body}(Q_1)$ and $\mathsf{id}(\mathsf{head}(Q_2)) = \mathsf{head}(Q_1) = H(x)$ holds—can serve in an application of $\sigma_2$ on the canonincal database $\mathsf{body}(Q_1)$, leading to the addition of the required atom $H(x)$ after one chase step.

The hardness proof for $\textsc{Contain}(\mathsf{CQ}, \mathsf{CQ})$ by Chandra and Merlin [CM77] is by reduction from the graph coloring problem and requires only a binary edge relation and no node variables, which gives $\alpha = 2$ and $\beta = 0$.

Now, we establish the matching upper bound. Fix an arity bound $\alpha$ and a context bound $\beta$ with $\alpha, \beta \geq 0$. Let $(\Sigma, \tau)$ be an input for $\textsc{Imp}_{\alpha}\big(\mathsf{DTGD}_{[\mathsf{bc}(\beta)]} \cup \mathsf{DEGD}_{[\mathsf{bc}(\beta)]}\big)$.

As explained above, although the restricted chase sequences for $\Sigma$ are guaranteed to be finite, they may be exponentially long. For an NP-algorithm, it suffices to show that if there is a chase sequence $\boldsymbol{a}$ that certifies $\tau$, then there is a chase sequence $\boldsymbol{a}''$ of polynomial length that certifies $\tau$.

To this end, we proceed in two steps. First, we normalise $\boldsymbol{a}$ and yield an equivalent chase sequence $\boldsymbol{a}'$. This sequence does not generate fewer nodes but 'simplifies' the

interdependencies between applications. Second, we select a subsequence $\boldsymbol{a}''$ from $\boldsymbol{a}'$ that generates only a polynomial number of node variables.

⋙ Normalisation. Let $\boldsymbol{a} = (a_1, \ldots, a_n)$ be a finite chase sequence. The normalisation step attempts to reduce the number of *witness nodes*, that is, node variables that appear in $V_i(\mathsf{body}(\sigma_i))$ for some application $(\sigma_i, V_i)$. By successive inspection, we define a chase sequence $\boldsymbol{a}' = (a'_1, \ldots, a'_n)$ that yields the same intermediate chase results but has a small number of witness nodes.

Let $a_i = (\sigma_i, V_i)$ be the $i$-th application in chase sequence $\boldsymbol{a}$, then the $i$-th application in chase sequence $\boldsymbol{a}'$ is defined as $a'_i = (\sigma_i, V'_i)$, where valuation $V'_i$ is determined dependent on the type of dependency $\sigma_i$.

- *If $\sigma_i$ is a global dtgd or a dtgd of Type* (G1) *or a data-identifying degd*, then $V'_i$ equals $V_i$. In this case, both applications are identical, $a'_i = a_i$.
- *If $\sigma_i$ is a dtgd of Type* (G2) *or* (C2) *or a degd $\mathcal{A} \to (\kappa = \lambda)$ of Type* (E2), then all node variables except for the head node variable $\kappa$ have $\beta$-bounded context.

  For every node variable $\mu$ of $\sigma_i$ different from $\kappa$, if there is application $a_h$ with $h < i$ such that $\sigma_i = \sigma_h$ and $V_i(x) = V_h(x)$ for every data variable $x \in \mathsf{cont}_\mu(\sigma_i)$, then we define $V'_i(\mu) = V_h(\mu)$ for the minimal such $h$. Furthermore, let $V'_i$ agree with $V_i$ on all other variables.
- *If $\sigma_i$ is a dtgd of Type* (C1) *with head node variable $\kappa$*, then there may be further node variables, with or without $\beta$-bounded body context.

  If there is an application $a_h$ with $h < i$ such that $\sigma_i = \sigma_h$ and $V_i(x) = V_h(x)$ for every data variable $x \in \mathsf{cont}_\kappa(\sigma_i)$, then we define $V'_i(\mu) = V_h(\mu)$ for every node variable $\mu$ different from $\kappa$ and the minimal such $h$. Furthermore, let $V'_i$ agree with $V_h$ on all remaining (node and data) variables.
- *If $\sigma_i$ is a degd $\mathcal{A} \to (\kappa = \lambda)$ of Type* (E1) *but not* (E2), then there may be non-head node variables with $\beta$-bounded context.

  If there is an application $a_h$ with $h < i$ such that $\sigma_i = \sigma_h$ and $V_i(x) = V_h(x)$ for every data variable $x \in \mathsf{cont}_\kappa(\sigma_i) \cup \mathsf{cont}_\lambda(\sigma_i)$, then we define $V'_i(\mu) = V_h(\mu)$. Furthermore, let $V'_i$ agree with $V_i$ on all remaining (node and data) variables.

Note that valuation $V'_i$ is well-defined and agrees with $V_i$ on the head variables of $\sigma_i$. The latter ensures—as claimed—that $\boldsymbol{a}'$ induces the same intermediate chase results $(\mathbb{D}_0, \ldots, \mathbb{D}_n)$ as $\boldsymbol{a}$. In particular, sequence $\boldsymbol{a}'$ generates as many nodes as $\boldsymbol{a}$. However, not all of these nodes are necessary to certify $\tau$.

⋙ Subsequencing. If the number of nodes referenced by chase sequence $\boldsymbol{a}'$ is exponential, most of the nodes (and their local instances) are irrelevant for the certification of $\tau$. In the following, we show that a certifying subsequence with fewer nodes exists.

Let $\mathcal{K}'' \stackrel{\text{def}}{=} \mathcal{K}''_1 \cup \mathcal{K}''_2$, where $\mathcal{K}''_1$ is the subset of node variables from $\mathbb{D}_n$ that are witnesses for $\boldsymbol{a}'$ and $\mathcal{K}''_2$ is the set of node variables that are in $\mathsf{body}(\tau)$ or certify $\tau$.

We define $\boldsymbol{a}''$ as a subsequence of $\boldsymbol{a}'$ by removing each application $(\sigma_i, V_i)$ from $\boldsymbol{a}'$ that references at least one node that is *not* in $\mathcal{K}''$, while keeping all others (in the induced order). This subsequence is indeed a chase sequence because, by definition, it is contiguous and ordered: every application $a_i''$ refers only to nodes that exist in $\mathbb{D}''_{i-1}$ with the required facts. Moreover, subsequence $\boldsymbol{a}''$ still certifies $\tau$. Hence, it remains to show that $\boldsymbol{a}''$ has polynomial length and that each distributed database in the corresponding chase sequence has polynomial size.

By Claim 5.2.18, the number of facts in the global instance and in each local instance is bounded by $(\|\Sigma\| + \|\tau\|)\|\tau\|^{\alpha}$, which is polynomial in $\|(\Sigma, \tau)\|$.

Furthermore, the number of nodes in $\mathcal{K}''$ is polynomial too. To prove this, we consider all applications of each dependency $\sigma$ in $\boldsymbol{a}''$ by the type of the dependency. Let $d$ denote the number of data variables in $\tau$.

- *If $\sigma$ is a data-identifying degd*, then there are at most $d$ applications of $\sigma$. Thus, they have at most $\|\sigma\| \cdot d$ witness nodes.
- *If $\sigma$ is a global dtgd*, then there are at most $d^{\alpha}$ applications of $\sigma$. Thus, they have at most $\|\sigma\| \cdot d^{\alpha}$ witness nodes.
- *If $\sigma$ is a dtgd of Type* (G1), then there are at most $d^{\beta}$ applications of $\sigma$ because the head node variable is bounded. Thus, they have at most $\|\sigma\| \cdot d^{\beta}$ witness nodes.
- *If $\sigma$ is dtgd of Type* (G2) *or* (C2) *or a degd of Type* (E2), then there are at most $d^{\beta}$ valuations for each non-head node variable because these are bounded. Thus, all applications have at most $\|\sigma\| \cdot d^{\beta}$ witness nodes.
- *If $\sigma$ is a dtgd of Type* (C1) *with head node variable $\kappa$*, there are at most $d^{\beta}$ valuations for the data variables in $\mathsf{cont}_{\kappa}(\mathsf{body}(\sigma))$ and at most $d^{\alpha}$ valuations for the data variables in $\mathsf{cont}_{\kappa}(\mathsf{head}(\sigma))$. Thus, all applications have at most $\|\sigma\| \cdot d^{\beta+\alpha}$ witness nodes.
- *If $\sigma$ is a node-identifying degd of Type* (E1), then there are at most $d^{\beta}$ valuations for the data variables of each head node variable. Thus, all applications have at most $\|\sigma\| \cdot d^{2\beta}$ witness nodes.

The length of each dependency is trivially bounded by the encoding size of all dependencies, $\|\sigma\| \leq \|\Sigma\|$, and thus the bounds above can be weakened to the bound $\|\Sigma\| \cdot \|\tau\|^{2\beta+\alpha+1}$. Furthermore, the number of dependencies in $\Sigma$ is at most $\|\Sigma\|$ and $d \leq \|\tau\|$ holds. In summary, this yields a bound of $\|\Sigma\|^2 \cdot \|\tau\|^{2\beta+\alpha+1}$ on the number of witness nodes.

Therefore, a polynomial-size chase sequence $\boldsymbol{a}''$ exists. A nondeterministic algorithm can obviously 'guess' this sequence (at once), keep track of homomorphism $\varepsilon(\boldsymbol{a}'')$, check the correctness of the sequence and 'guess' a certificate, all in polynomial time. □

Our previous investigation has shown that, even if a set of dependencies implies the existence of an exponential number of nodes, it might be that most nodes are irrelevant for the certification of a dependency. The reason for this is the *bounded context of node variables*, which restricts the amount of data that can be accessed from a specific node. In a sense, Definition 5.2.19 allows unbounded context only on one side (body or head) of each dependency and thereby restricts the 'necessary' interdependencies between nodes in chase sequences. Next, we depart from this restriction.

### 5.2.3. PSPACE-Fragment

In the previous section, we have argued that interdependencies between nodes in chase sequences are, in principle, rather simple for distribution dependencies with bounded context. The intuitive reason for this is that only a very limited (context-bounded) amount of data *per node* can be referred in the body or in the head of a dependency.

Now, we show that this does not hold anymore if (different) node variables with unbounded context are allowed both in the body and in the head. Accordingly, the complexity of the implication problem rises. However, if only *one* node variable with unbounded context is allowed in the body, the complexity rises 'moderately' from NP to PSPACE. Later, we show that other extensions lead to EXPTIME.

We start with a formalisation of the mentioned additional type of dependencies.

**Definition 5.2.21 (Weakly bounded context).** Let $\beta \in \mathbb{N}_0$. A distribution dependency has *weakly $\beta$-bounded context* if it has $\beta$-bounded context or it is a node-generating dtgd such that

(G3) exactly one of its body node variables has *not* $\beta$-bounded context.

Let $\mathsf{DTGD}[\mathsf{wbc}(\beta)]$ denote the class of all dtgds with weakly $\beta$-bounded context. ◀

Different from the dependencies in Section 5.2.2, chase sequences for dependencies of Type (G3) may lead to the generation of exponentially many nodes such that all of them are indeed required. We illustrate this next.

**Example 5.2.22.** Let $n$ be a positive integer. We define a dtgd $\tau$ and a set $\Sigma \cup \{\sigma\}$ of weakly-bounded dtgds such that $\Sigma \cup \{\sigma\}$ implies $\tau$ but every certifying chase sequence has exponential length in $n$.

First, let $\Sigma = \{\sigma_0, \ldots, \sigma_{n-1}\}$ be a set where, for every $i \in \{0, \ldots, n-1\}$, dependency $\sigma_i$ is a dtgd of Type (G3), defined by

$$\begin{aligned}\mathsf{body}(\sigma_i) &\stackrel{\text{def}}{=} \{\mathtt{Bool}(b_n), \ldots, \mathtt{Bool}(b_{i+1}), \mathtt{False}(w_0), \mathtt{True}(w_1)\} \\ &\quad \cup \{R_n(b_n), \ldots, R_{i+1}(b_{i+1}), R_i(w_0), R_{i-1}(w_1), \ldots, R_0(w_1)\}@\kappa, \\ \mathsf{head}(\sigma_i) &\stackrel{\text{def}}{=} \{R_n(b_n), \ldots, R_{i+1}(b_{i+1}), R_i(w_1), R_{i-1}(w_0), \ldots, R_0(w_0)\}@\lambda.\end{aligned}$$

This set emulates the stepwise incrementation of a binary counter $b_n, \dots, b_0$, represented by the $R_i$-facts on some node $k$.

Second, dependency $\tau$ is a dtgd of Type (G3) where

$$\begin{aligned}\mathsf{body}(\tau) &\stackrel{\text{def}}{=} \{\mathtt{Bool}(w_0), \mathtt{Bool}(w_1), \mathtt{False}(w_0), \mathtt{True}(w_1)\} \\ &\quad \cup \{R_n(w_0), \dots, R_0(w_0)\}@\kappa, \\ \mathsf{head}(\tau) &\stackrel{\text{def}}{=} \{S(w_1)\}\end{aligned}$$

that provides the necessary atoms to initiate the incrementation process.

Lastly, the following dtgd $\sigma$ with 2-bounded context,

$$\begin{aligned}\mathsf{body}(\sigma) &\stackrel{\text{def}}{=} \{\mathtt{True}(w_1)\} \cup \{R_n(w_1), R_{n-1}(w_1)\}@\kappa, \\ \mathsf{head}(\sigma) &\stackrel{\text{def}}{=} \{S(w_1)\},\end{aligned}$$

which links the presence of a binary representation of the form $11\cdots$ to the existence of fact $S(1)$.

Evidently, a chase sequence from $\mathsf{body}(\tau)$, in the beginning, allows the restricted application of dependencies from $\Sigma$ only: starting with the representation $0\dots0$ on a node, the intermediate chase result eventually contains $110\dots0$ on some other node. This is the first time that dependency $\sigma$ can be applied, which then certifies $\tau$. Overall, a certifying chase sequence requires at least $2^n + 2^{n-1} + 1$ applications and leads to the generation of $2^n + 2^{n-1} - 1$ nodes. ■

Nevertheless, most nodes are important only for a 'short while' and can be ignored afterwards. This idea is made more precise in the proof of the following theorem.

**Theorem 5.2.23.** $\textsc{Imp}_\alpha\big(\mathsf{DTGD}_{[\mathsf{wbc}(\beta)]} \cup \mathsf{DEGD}_{[\mathsf{bc}(\beta)]}\big)$ is in PSPACE for fixed $\alpha, \beta \geq 0$.

*Proof.* Fix an arity bound $\alpha$ and a context bound $\beta$ with $\alpha, \beta \geq 0$. Let $(\Sigma, \tau)$ be an input for $\textsc{Imp}_\alpha\big(\mathsf{DTGD}_{[\mathsf{wbc}(\beta)]} \cup \mathsf{DEGD}_{[\mathsf{bc}(\beta)]}\big)$.

Since PSPACE = NPSPACE, it suffices to provide a nondeterminstic algorithm with polynomially bounded space that tests for the existence of a chase sequence that certifies $\tau$ under $\Sigma$. As illustrated above, even the shortest chase sequence may have exponential length. Therefore, such an algorithm *cannot* 'guess' and verify the sequence *at once*. A canonical attempt to bound the space is to guess successive elements of the sequence and 'forget' those that are not needed anymore. However, to verify the coherence of the sequence, the algorithm has to keep track of all facts that are used in later chase steps—and the nodes where they are located. Unfortunately, the previous example demonstrates that also the number of such nodes can be exponential. Nevertheless, we show in the following that the 'linear' form of dtgds of Type (G3) allows us to bound the number of nodes that are relevant for each step by a polynomial, despite their possible interaction with dependencies with $\beta$-bounded context. Intuitively, we bound the 'width' of the chase, not its 'length'.

We show that a well-behaved certifying chase sequence exists if some certifying chase sequence exists. To this end, we rely on *normalised* chase sequences and an observation on *minimal* such sequences.

⋙ Normalisation. The normalisation defined in the proof of Theorem 5.2.20 can be extended in the following way—considering, as before, all applications $a_i = (\sigma_i, V_i)$ successively. Assume that $\sigma_i$ does not have $\beta$-bounded context but is of Type (G3), then the following modification is applied, yielding $a_i' = (\sigma_i, V_i')$.

- If $\sigma_i$ has head node variable $\kappa$, then, for every node variable $\mu$ different from $\kappa$, if there is an application $a_h$ with $h < i$ such that $\sigma_i = \sigma_h$ and $V_i(x) = V_h(x)$ for every data variable $x \in \mathsf{cont}_\mu(\sigma_i)$, then we define $V_i'(\mu) = V_h(\mu)$ for the minimal such $h$. Furthermore, let $V_i'$ agree with $V_i$ on all remaining (node and data) variables.

In the remainder of this proof, we show that if there is a chase sequence $\boldsymbol{a}$ that certifies $\tau$, then there is also a chase sequence $\boldsymbol{a}'$ that certifies $\tau$ such that $\boldsymbol{a}'$ can be guessed and verified in a linear fashion, requiring only polynomial space. To this end, let $\boldsymbol{a}$ be a chase sequence that certifies $\tau$ and that satisfies the following two assumptions.

First, we assume, without loss of generality, that $\boldsymbol{a}$ is a *restricted* chase sequence. Then, as argued in Claim 5.2.18, its length is bounded exponentially due to the arity bound $\alpha$, which is less than $\|\Sigma\| + \|\mathbb{D}_0\|$. This bound $B = B(\Sigma, \tau)$ is obviously computable (even in polynomial time) and can be used by the verification algorithm to guarantee termination: with each newly guessed application, a counter is incremented by 1. If the counter exceeds $B$ and $\tau$ has not been certified, the algorithm rejects. Clearly, this counter requires only polynomial space.

Second, we assume that $\boldsymbol{a}$ is a *normalised* chase sequence that certifies $\tau$ of *minimal* length. Then, the sequence contains only applications that are needed for the certification of $\tau$. In particular, it contains only applications of node-generating dtgds such that the generated node participates in the certification of $\tau$ or is referenced by a later application.

Let us consider the set of nodes referenced in $\boldsymbol{a}$ as a bipartition $\mathcal{K} \uplus \mathcal{L}$, where $\mathcal{K}$ contains each node that satisfies at least one of the following properties:

1. it is a witness node as in the proof of Theorem 5.2.20; or
2. it is a node that is referenced via node variables *with $\beta$-bounded context* in applications of dtgds of Type (G3); or
3. is is a node that participates in the certification of $\tau$.

Then,—following the argument in the proof of Theorem 5.2.20 again—the size of $\mathcal{K}$ is still in $\mathcal{O}(\|\Sigma\|^2 \cdot \|\tau\|^{\alpha+2\beta+1})$ because the number of witness nodes for dtgds of Type (G3) is at most $\|\Sigma\| \cdot \|\tau\|^\beta$, thanks to the normalisation.

Note that the set $\mathcal{K}$ is closed under predecessors in the previous proof while this is *not* necessarily the case now. We further remark that set $\mathcal{K}$ may contain nodes that are introduced by the application of dtgds of Type (G3). Set $\mathcal{L}$, however, contains exclusively nodes that are generated via dtgds of Type (G3) due to the assumed normalisation and minimality of chase sequence $\boldsymbol{a}$.

⋙ Properties of minimal chase sequences. Although we cannot bound the size of $\mathcal{L}$ polynomially, we can bound the sizes of subsets $\mathcal{L}_0, \ldots, \mathcal{L}_n \subseteq \mathcal{L}$ accordingly such that, for every application $a_i$ in chase sequence $\boldsymbol{a} = (a_1, \ldots, a_n)$,

(P1) application $a_i$ requires only node variables in $\mathcal{K} \cup \mathcal{L}_{i-1}$; and

(P2) for every node variable $\lambda \in \mathcal{L}$, if $\lambda \in \mathcal{L}_{i-1}$ but $\lambda \notin \mathcal{L}_i$, then $\lambda \notin \mathcal{L}_i \cup \ldots \cup \mathcal{L}_n$.

Sets $\mathcal{L}_1, \ldots, \mathcal{L}_n$ are defined inductively, from $i = n$ down to 1, as follows. For formal reasons, fix $\mathcal{L}_n \stackrel{\text{def}}{=} \emptyset$. We distinguish two cases, based on the dependency in application $a_i = (\sigma_i, h_i)$.

1. *If $\sigma_i$ has $\beta$-bounded context*, then $h_i(\mathsf{body}(\sigma_i))$ refers to node variables in $\mathcal{K}$ only since $\boldsymbol{a}$ is normalised. In this case, let $\mathcal{L}_{i-1} \stackrel{\text{def}}{=} \mathcal{L}_i$.
2. *Otherwise*, $\sigma_i$ is a dtgd of Type (G3), which references exactly two node variables $\lambda, \lambda'$ with unbounded context. Let $\lambda'$ be the newly generated node variable.
   a) *If* $\lambda' \in \mathcal{K}$, let $\mathcal{L}_{i-1} \stackrel{\text{def}}{=} \mathcal{L}_i \cup \{\lambda\}$.
   b) *Otherwise*, let $\mathcal{L}_{i-1} \stackrel{\text{def}}{=} (\mathcal{L}_i - \{\lambda'\}) \cup \{\lambda\}$.

Property (P1) holds because, in Case 1), chase step $a_i$ requires only node variables in $\mathcal{K} \subseteq \mathcal{K} \cup \mathcal{L}_{i-1}$ and, in Case 2), chase step $a_i$ requires only node variables in $\mathcal{K} \cup \{\lambda\} \subseteq \mathcal{K} \cup \mathcal{L}_{i-1}$, by definition of $\mathcal{L}_{i-1}$.

Property (P2) holds as the following argument shows. Towards a contradiction, assume that there is a node variable $\lambda$ such that $\lambda \in \mathcal{L}_{i_1}$ and $\lambda \in \mathcal{L}_{i_3}$ but $\lambda \notin \mathcal{L}_{i_2}$ for indices $i_1 < i_2 < i_3$. Without loss of generality, we can assume that $i_3 = i_2 + 1$ and that $i_1$ is maximal with that property. Then, in the construction of $\mathcal{L}_{i_2}$, node variable $\lambda$ has been removed from $\mathcal{L}_{i_3}$ in Case 2b), that is, in the role of node variable $\lambda'$, which has been *generated* by application $a_{i_3}$. However, $\lambda \in \mathcal{L}_{i_1}$ implies that the application of $a_{i_1+1}$ requires $\lambda$. This is a contradiction, since $\boldsymbol{a}$ is a (valid) chase sequence and $\lambda$ cannot have been generated before (more than once).

Furthermore, subsets $\mathcal{L}_0, \ldots, \mathcal{L}_n$ are sufficiently small. In Case 1), the size of $\mathcal{L}_i$ obviously remains the same, compared to $\mathcal{L}_{i+1}$. This is is also true for Case 2b) because, by minimality, the generated node variable $\lambda'$ is required by a later application—if it would be needed only to certify $\tau$, then it would be in $\mathcal{K}$. Finally, in Case 2a), the size of $\mathcal{L}_i$ increases by at most 1—if $\lambda'$ has not already been in $\mathcal{L}_{i+1}$. Notably, the latter case occurs at most $|\mathcal{K}|$ times since every node is generated only once. Hence, starting with $\mathcal{L}_n = \emptyset$, we have $|\mathcal{L}_i| \leq |\mathcal{K}|$ for every $i \in \{0, \ldots, n\}$, a polynomial bound.

To conclude the proof, we sketch the algorithm in more detail.

⋙ Algorithm. An algorithm for $\textsc{Imp}_\alpha(\mathsf{DTGD}_{[\mathsf{wbc}(\beta)]} \cup \mathsf{DEGD}_{[\mathsf{bc}(\beta)]})$ can, if a certifying sequence $\boldsymbol{a}$ for $\tau$ exists, nondeterministically construct—step by step—a sequence $\boldsymbol{a}'$ that certifies $\tau$ and that *resembles* the restricted minimal sequence certifying $\tau$, which exists by the previous argument. In the $i$-th step, the algorithm has to keep track of all node variables in $\mathcal{K} \cup \mathcal{L}_{i-1}$ and $\mathcal{K} \cup \mathcal{L}_i$ and their local instances, respectively. For each generated node, the algorithm nondeterministically determines whether it belongs to $\mathcal{K}$ or $\mathcal{L}_i$. As argued before, both sets have polynomial cardinality. Furthermore, the sizes of the corresponding local instances are also polynomially bounded since $\alpha$ is fixed. As usual, the algorithm also keeps track of the homomorphism $\varepsilon = \varepsilon(\boldsymbol{a}')$, which refers to nodes from $\mathsf{body}(\tau)$ only and thus also requires polynomial space. After each step, certification can be checked—by guessing an extension $\varepsilon'$—and the algorithm proceeds at most $B$ steps. Clearly, if no certifying sequence exists, then the algorithm does not find any.

We emphasise that the algorithm *neither* verifies that sequence $\boldsymbol{a}'$ is *restricted nor* that it is *minimal*. An oblivious non-minimal sequence is fine as well—as long as it certifies $\tau$ and obeys the length bound $B$.

Therefore, $\textsc{Imp}_\alpha(\mathsf{DTGD}_{[\mathsf{wbc}(\beta)]} \cup \mathsf{DEGD}_{[\mathsf{bc}(\beta)]})$ is in PSPACE, as claimed. □

Indeed, the implication problem for dtgds with weakly bounded context is complete for PSPACE. This has been proven by a reduction from linear bounded automata [GNS20], similar to that for the implication problem for inclusion dependencies, which is also PSPACE-complete [AHV95].

**Proposition* 5.2.24.** $\textsc{Imp}_\alpha(\mathsf{DTGD}_{[\mathsf{wbc}(\beta)]})$ is PSPACE-hard for $\alpha \geq 1$ and $\beta \geq 0$.

The last results indicate that allowing node variables with unbounded context in both body and head of dependencies does indeed increase the worst-case complexity of the implication problem, even though dtgds of Type (G3) are rather restricted. In the last part of this section, we show for most of the other possible relaxations, that the increase is even higher.

### 5.2.4. EXPTIME-Fragments

In our study of the implication problem for data-full distribution dependencies above, we have ignored certain types of dependencies like the following, illustrated in Table 5.1:

- dtgds where the head node variable and at least *two* body node variables have unbounded context, like
  - node-generating dtgds of Type (G4) or

  - data-collecting dtgds of Type (C3); and
- degds where at least *both* head node variables have unbounded context, like
  - degds of Type (E3) or
  - degds of Type (E4).

In the following we show that the extension of the previous fragments by one of these types makes the interdependencies between nodes in chase sequence in the worst case so complicated that it allows to model arbitrary problems that are solvable in exponential time. Indeed, this already holds for smaller fragments, as shown further below.

First, however, we prove the corresponding upper bound for arbitrary data-full distribution dependencies.

*Proof of Theorem 5.2.3.* That $\textsc{Imp}(\mathsf{DTGD}_{[\mathsf{df}]} \cup \mathsf{DEGD})$ is in EXPTIME, follows quite directly from the properties of restricted chase sequences for these dependencies provided by Claim 5.2.18. Since the arity of relations is trivially bounded by the encoding size of the dependencies, the length of such a chase sequence is in $\mathcal{O}\left(\|\Sigma\|^2 \cdot \|\mathbb{D}_0\|^{2\|\Sigma\|}\right)$, and hence exponential in the input. Clearly, an algorithm can determine valid restricted chase steps (and keep track of the intermediate chase results) in exponential time.

Hardness, in turn, is a consequence of Theorem 5.2.25. □

**Theorem 5.2.25.** $\textsc{Imp}(\mathsf{DTGD}_{[\mathsf{df}]} \cup \mathsf{DEGD})$ is EXPTIME-hard. This lower bound already holds if the input is restricted to one of the following combinations, even over schemas with a maximal arity of 2, that is, for $\textsc{Imp}_2(\mathsf{DTGD}_{[\mathsf{df}]} \cup \mathsf{DEGD})$.

a) Node-generating dtgds of Type (G2) and data-collecting dtgds of Type (C3);
b) Node-generating dtgds of Types (G2) and (G4);
c) Node-generating dtgds of Type (G2) and degds of Type (E4);
d) Node-generating dtgds of Types (G2) and (G3) and degds of Type (E3).

*Proof.* Without additional restrictions, the EXPTIME-hardness of $\textsc{Imp}(\mathsf{DTGD}_{[\mathsf{df}]})$ follows from an early EXPTIME-completeness result for full (classical) tgds [CLM81]. This result however relies on schemas of unbounded arity—for a fixed arity bound, the problem is easily seen to be in NP. The arguments below show that *in the presence of node variables*, hardness already follows for schemas with only nullary, unary and binary relations.

For each of the four fragments, the argument follows the same line of reasoning. Fix an EXPTIME-complete problem $L$ and an alternating Turing machine $\mathcal{M}$ with linearly

bounded space that decides $L$. We provide polynomial reductions from $L$ based on the general idea described next.[6]

Given a word $w$, an instance $(\Sigma, \tau)$ for $\textsc{Imp}_\alpha$ is computed such that the chase simulates the computation of Turing machine $\mathcal{M}$ on input $w$. Basically, every node $k$ in the chase result represents a single configuration $C(k)$ of the Turing machine.

The dependencies in $\Sigma$ are responsible for, first, the generation of the configuration representations, and, second, to mark them accepting—if appropriate. As usual, for an alternating Turing machine, a configuration $C$ referring to state $q$ is accepting if one of the following conditions holds:

- state $q$ is accepting; or
- state $q$ is universal and both its successor configurations are accepting; or
- state $q$ is existential and at least one of its successor configurations is accepting.

Finally, Turing machine $\mathcal{M}$ accepts input $w$ if the initial configuration is accepting.

Let us consider this approach in more detail. We assume that the Turing machine is of the form $\mathcal{M} = (Q, A, (\delta_1, \delta_2), q_0, F)$ with a finite bipartite set $Q = Q_\exists \uplus Q_\forall$ of *states* that are either *existential* or *universal*. State $q_0 \in Q$ is the *initial state* and the states in $F \subseteq Q$ are *accepting*. Furthermore, for each state $q \in Q$ and each symbol $a \in A$ of the *alphabet* $A = \{a_1, \ldots, a_t\}$, two transitions $\delta_1(q, a)$ and $\delta_2(q, a)$ are defined. Each transition function uniquely determines a successor configuration $C'$ for each configuration $C$. This relation is denoted by $C \vdash_j C'$ for the $j$-th transition function $\delta_j$.

Without loss of generality, we assume that the initial state is *not* accepting and that, for each input $w = w_1 \ldots w_n$, the machine uses only $n + 2$ cells of its tape. Furthermore, the first and the last cell (with positions 0 and $n + 1$) contain marker symbols $a_1 = \triangleright$ and $a_t = \triangleleft$, which are never changed by the transition functions. Also, the transition functions do not move left or right, respectively, when reading these symbols. A configuration $C$ of $\mathcal{M}$ is represented as a triple $(q, i, u)$, consisting of a state $q \in Q$, a position $i \in \{0, \ldots, n + 1\}$ and a word $u \in A^n$ that represents the tape content $\triangleright u \triangleleft$.

We want to refer to $n + 2$ positions of the tape and its contents over alphabet $A$. To this end, a binary relation $\mathtt{Succ}$ establishes an order among $n + 2$ variables via the set

$$\mathcal{A}_{\text{succ}} \stackrel{\text{def}}{=} \{\mathtt{Succ}(\pi_0, \pi_1), \ldots, \mathtt{Succ}(\pi_n, \pi_{n+1})\}$$

of atoms and unary relations $\mathtt{Alph}_1, \ldots, \mathtt{Alph}_t$ and $\mathtt{Alph}$ are used either to distinguish different symbols or select an arbitrary one via the set

[6] The reductions are inspired by a reduction for weakly guarded tgds by Calì et al. [CGK13]. Technically, however, they differ significantly from the original reduction because in distribution tgds, each atom can refer to *at most one* node variable.

$$\mathcal{A}_{\text{alph}} \stackrel{\text{def}}{=} \{\texttt{Alph}_1(\alpha_1), \ldots, \texttt{Alph}_t(\alpha_t)\} \cup \{\texttt{Alph}(\alpha_1), \ldots, \texttt{Alph}(\alpha_t)\}$$

of atoms. Furthermore, to represent configurations, a nullary relation $\texttt{State}_q$ for each state $q \in Q$, a single unary relation `Head` for the position on the tape and a binary relation `Sym` for the cell contents of the tape are used. More concretely, with each state $q$, each position $i \in \{0, \ldots, n+1\}$ and each variable sequence $\boldsymbol{u} = (u_1, \ldots, u_n)$ of length $n$, we associate the set

$$\begin{aligned}\mathcal{C}_{q,i,\boldsymbol{u}} \stackrel{\text{def}}{=} &\{\texttt{State}_q, \texttt{Head}(\pi_i), \} \\ &\cup \{\texttt{Sym}(\pi_0, \alpha_1), \texttt{Sym}(\pi_1, u_1), \ldots, \texttt{Sym}(\pi_n, u_n), \texttt{Sym}(\pi_{n+1}, \alpha_t)\}\end{aligned}$$

of atoms, intended to represent a configuration $(q, i, u)$ for some word $u$ encoded by $\boldsymbol{u}$. Sometimes, a variable sequence $\boldsymbol{u}$ is intended to represent an arbitrary word, sometimes, a specific word $v = v_1 \ldots v_n \in A^n$ is intended instead. For the latter case, let $\alpha$ be defined by $\alpha(v_1 \ldots v_n) = (\alpha_{j_1} \ldots \alpha_{j_n})$, where $j_i \in \{1, \ldots, t\}$ is the unique index such that $v_i = a_{j_i}$ for each $i \in \{1, \ldots, n\}$, mapping words from $A^n$ to variable sequences of length $n$.

Additional nullary relations `Acc` and $\texttt{Acc}_1, \texttt{Acc}_2$ document acceptance. If `Acc` is present on a node $k$, it marks the configuration $C(k)$ *itself* accepting. Slightly different, the presence of $\texttt{Acc}_j$ marks the *$j$-th successor* configuration as accepting *on node $k$*.

The simulation of the Turing machine by the chase procedure follows different patterns, dependent on the types of dependencies allowed in each case. The behaviour of this procedure is described by the following algorithm for the proof of Statement a).

The other proofs follow similar approaches. Common to them are two phases: one for the generation of nodes representing configurations (Lines 2 – 5) and another for the propagation of acceptance flags (Lines 6 – 15). Accordingly, the set $\Sigma$ of dependencies is always defined as the union $\Sigma = \Sigma_1 \cup \Sigma_2$ of two sets that reflect these phases.

In all four reductions, the body of $\tau$ provides a node representing the initial configuration as well as, globally, the base information on the order of positions and the alphabet (Line 1).

As a side note, some of the dependencies defined below are only needed for certain choices of parameters. To keep the description simple, we use the symbol $\top$ to denote a tautological (purely global) dtgd of the form $\texttt{Acc} \to \texttt{Acc}$ for the cases where the dependency is not needed—instead of conditioning the addition of dependencies.

**Algorithm** $\mathbb{A}^{\mathsf{ATM}}_{\mathsf{sim,a}}$

**Input:** word $w$

add node $k_0$ representing $(q_0, 0, w)$
**for each** possible configuration $(q, i, v)$ where $|v| = |w|$ **do**
  add a node $k$ representing $(q, i, v)$
  **if** $q$ is accepting **then**
    add $\texttt{Acc}$ to $k$
**repeat**
  **for each** pair $(k, m)$ of nodes and $j \in \{1, 2\}$ **do**
    **if** $C(k) \vdash_j C(m)$ and $\texttt{Acc}@m$ **then**
      add $\texttt{Acc}_j$ to $k$
  **for each** node $k$ with state $q$ **do**
    **if** $q$ is existential **and** ($\texttt{Acc}_1@k$ **or** $\texttt{Acc}_2@k$) **then**
      add $\texttt{Acc}$ to $k$
    **if** $q$ is universal **and** ($\texttt{Acc}_1@k$ **and** $\texttt{Acc}_2@k$) **then**
      add $\texttt{Acc}$ to $k$
**until** no more changes
**if** $\texttt{Acc}@k_0$ **then**
  **accept**
**else**
  **reject**

⋙ Proof of a). $\textsc{Imp}(\mathsf{DTGD}_{[\mathsf{df}]} \cup \mathsf{DEGD})$ is already EXPTIME-hard if restricted to only node-generating dtgds of Type (G2) and data-collecting dtgds of Type (C3) as input. We start with the basic idea of the reduction from the acceptance problem for $\mathcal{M}$ and provide the details afterwards.

» Idea. Subset $\Sigma_1$ is responsible for the generation of one node for each possible configuration of the Turing machine (whether reachable or not). Configurations with accepting states are directly marked as accepting. Subset $\Sigma_2$ collects acceptance markers for successor configurations. Based on these markers, it also collects markers for the configuration itself.

Dependency $\tau$ requires that fact $\texttt{Acc}$ is present on node $k_0$ for the initial configuration.

» Details. Dependency $\tau$ relates to the input word $w$ and is defined as follows.

$$\mathsf{body}(\tau) \stackrel{\text{def}}{=} (\mathcal{A}_{\text{succ}} \cup \mathcal{A}_{\text{alph}} \cup \mathcal{C}_{q_0,0,\alpha(w)})@\kappa_0$$
$$\mathsf{head}(\tau) \stackrel{\text{def}}{=} \mathtt{Acc}@\kappa_0$$

To represent all possible configurations, $\Sigma_1$ contains a node-generating dtgd $\sigma^{\text{init}}_{q,i}$ of Type (G2) for every state $q \in Q$ and every position $i \in \{0, \ldots, n+1\}$.

$$\mathsf{body}(\sigma^{\text{init}}_{q,i}) \stackrel{\text{def}}{=} \mathcal{A}_{\text{succ}} \cup \{\mathtt{Alph}_1(\alpha_1), \mathtt{Alph}_t(\alpha_t)\} \cup \{\mathtt{Alph}(u_1), \ldots, \mathtt{Alph}(u_n)\}$$
$$\mathsf{head}(\sigma^{\text{init}}_{q,i}) \stackrel{\text{def}}{=} (\mathcal{C}_{q,i,u} \cup \{\mathtt{Acc} \mid q \in F\})@\kappa$$

Note that the acceptance marker $\mathtt{Acc}$ only occurs in those dependencies that refer to an accepting state.

Finally, $\Sigma_2$ contains data-collecting dtgds of Type (C3). First, to propagate acceptance markers for successor configurations, it contains a dtgd $\sigma^{\text{trans}}_{q,h,i,j}$ for every state $q \in Q$, every symbol $a_h \in A$, every position $i \in \{0, \ldots, n+1\}$ and both $j \in \{1, 2\}$. In particular, these dependencies reflect the transition functions. We exemplify the construction of this dependency under the assumption that the transition $\delta_j(q, a_h)$ leads to state $\hat{q}$, overwrites the cell content by a symbol $a_g \in A$ and moves one step to the right.

If the head is already on the last position, $i = n+1$, then $\sigma^{\text{trans}}_{q,h,i,j} \stackrel{\text{def}}{=} \top$. Otherwise, this dtgd is defined by

$$\begin{aligned}\mathsf{body}(\sigma^{\text{trans}}_{q,h,i,j}) &\stackrel{\text{def}}{=} \mathcal{A}_{\text{succ}} \cup \{\mathtt{Alph}_h(u_i), \mathtt{Alph}_g(\hat{u}_i)\} \\ &\quad \cup\, \mathcal{C}_{q,i,\boldsymbol{u}}@\kappa \cup (\mathcal{C}_{\hat{q},i+1,\hat{\boldsymbol{u}}} \cup \{\mathtt{Acc}\})@\mu \\ \mathsf{head}(\sigma^{\text{trans}}_{q,h,i,j}) &\stackrel{\text{def}}{=} \{\mathtt{Acc}_j\}@\kappa,\end{aligned}$$

where $\hat{\boldsymbol{u}} = (u_1, \ldots, u_{i-1}, \hat{u}_i, u_{i+1}, \ldots, u_n)$ represents the new tape content, possibly differing from $\boldsymbol{u}$ in the $i$-th position.

Second, to derive acceptance markers based on the information about the successors' acceptance behaviour, $\Sigma_2$ also contains a dtgd $\sigma^{\text{acc}}_q$ for each universal state $q$ and, analogously, two dtgds $\sigma^{\text{acc}}_{q,1}$ and $\sigma^{\text{acc}}_{q,2}$ for each existential state. Formally, they are defined as follows.

$$\mathsf{body}(\sigma^{\text{acc}}_q) \stackrel{\text{def}}{=} \{\mathtt{State}_q, \mathtt{Acc}_1, \mathtt{Acc}_2\}@\kappa$$
$$\mathsf{body}(\sigma^{\text{acc}}_{q,1}) \stackrel{\text{def}}{=} \{\mathtt{State}_q, \mathtt{Acc}_1\}@\kappa$$
$$\mathsf{body}(\sigma^{\text{acc}}_{q,2}) \stackrel{\text{def}}{=} \{\mathtt{State}_q, \mathtt{Acc}_2\}@\kappa$$
$$\mathsf{head}(\sigma^{\text{acc}}_q) \stackrel{\text{def}}{=} \mathsf{head}(\sigma^{\text{acc}}_{q,1}) \stackrel{\text{def}}{=} \mathsf{head}(\sigma^{\text{acc}}_{q,2}) \stackrel{\text{def}}{=} \{\mathtt{Acc}\}@\kappa$$

Obviously, the mapping described above is total and the output $(\Sigma, \tau)$ can be derived in polynomial time from $\mathcal{M}$ and $w$. The mapping also satisfies the reduction property: Turing machine $\mathcal{M}$ accepts input $w$ if and only if $\Sigma \models \tau$. This can be proven by a straightforward induction showing that each node $k$ contains fact $\mathtt{Acc}$ if and only if configuration $C(k)$ is accepting.

⋙ Proof of b). $\textsc{Imp}(\textsf{DTGD}_{[\textsf{df}]} \cup \textsf{DEGD})$ is already EXPTIME-hard if restricted to only node-generating dtgds of Type (G2) and of Type (G4) as input. We start with the basic idea of the reduction and provide the details afterwards.

» Idea. The following construction deviates from the previous because acceptance markers cannot be 'collected' anymore on the node representing a configuration (only node-generating dtgds are allowed). To circumvent this restriction, each configuration is now represented by up to four nodes, $k, k_1, k_2, k^*$. Intentionally, the nodes only differ in the presence of fact $\texttt{Acc}$, $\texttt{Acc}_1$ or $\texttt{Acc}_2$, respectively, as derived in the second phase.

Subset $\Sigma_1$ is responsible for the generation of (only) one node per configuration—either with or without $\texttt{Acc}$. Subset $\Sigma_2$ generates further nodes $k_1$ and $k_2$ for configurations with markers $\texttt{Acc}_1$ and $\texttt{Acc}_2$, if appropriate. Based on these markers, it generates another node $k^*$, depending on whether the state is universal or existential.

» Details. Dependency $\tau$ demands that there is a node $k^*$ representing the initial configuration *with an acceptance flag*, if there is some node $k$ representing the initial configuration. It is defined as follows.

$$\begin{aligned}\mathsf{body}(\tau) &\stackrel{\text{def}}{=} \big(\mathcal{A}_{\text{succ}} \cup \mathcal{A}_{\text{alph}} \cup \mathcal{C}_{q_0,0,\alpha(w)}\big)@\kappa\\ \mathsf{head}(\tau) &\stackrel{\text{def}}{=} \big(\mathcal{C}_{q_0,0,\alpha(w)} \cup \{\texttt{Acc}\}\big)@\kappa^*\end{aligned}$$

Subset $\Sigma_1$ is the same set of dependencies as used in the proof of a), containing only node-generating dtgds of Type (G2). Subset $\Sigma_2$ however is defined differently, using node-generating dtgds of Type (G4) instead of data-collecting dtgds. Again, this subset contains two forms of dependencies. First, to propagate acceptance markers for sucessor configurations, it contains a dtgd $\sigma^{\text{trans}}_{q,h,i,j}$ for every state $q \in Q$, every symbol $a_h \in A$, every position $i \in \{0, \dots, n+1\}$ and both $j \in \{1, 2\}$. As before, we exemplify the construction for a transition $\delta_j(q, a_h)$ that leads to state $\hat{q}$, overwrites the cell content by a symbol $a_g \in A$ and moves one step to the right.

If the head is already on the last position, $i = n + 1$, then $\sigma^{\text{trans}}_{q,h,i,j} \stackrel{\text{def}}{=} \top$. Otherwise, this dtgd is defined by

$$\begin{aligned}\mathsf{body}(\sigma^{\text{trans}}_{q,h,i,j}) &\stackrel{\text{def}}{=} \mathcal{A}_{\text{succ}} \cup \{\texttt{Alph}_h(u_i), \texttt{Alph}_g(\hat{u}_i)\}\\ &\qquad \cup\ \mathcal{C}_{q,i,\boldsymbol{u}}@\kappa \cup \mathcal{C}_{\hat{q},i+1,\hat{\boldsymbol{u}}}@\mu\\ \mathsf{head}(\sigma^{\text{trans}}_{q,h,i,j}) &\stackrel{\text{def}}{=} \big(\mathcal{C}_{q,i,\boldsymbol{u}} \cup \{\texttt{Acc}_j\}\big)@\kappa_j\end{aligned}$$

where $\hat{\boldsymbol{u}} = (u_1, \dots, u_{i-1}, \hat{u}_i, u_{i+1}, \dots, u_n)$ represents the new tape content, as above. Furthermore, subset $\Sigma_2$ contains dtgd $\sigma^{\text{acc}}_q$ for each universal state $q$.

$$\begin{aligned}\mathsf{body}(\sigma^{\text{acc}}_q) &\stackrel{\text{def}}{=} \big(\mathcal{C}_{q,i,\boldsymbol{u}} \cup \{\texttt{Acc}_1\}\big)@\kappa_1 \cup \big(\mathcal{C}_{q,i,\boldsymbol{u}} \cup \{\texttt{Acc}_2\}\big)@\kappa_2\\ \mathsf{head}(\sigma^{\text{acc}}_q) &\stackrel{\text{def}}{=} \big(\mathcal{C}_{q,i,\boldsymbol{u}} \cup \{\texttt{Acc}\}\big)@\kappa^*,\end{aligned}$$

Analogously, $\Sigma_2$ contains two dtgds $\sigma^{\text{acc}}_{q,1}$ and $\sigma^{\text{acc}}_{q,2}$ for each existential state $q$. Again, the correctness of this construction can be proven with a straightforward induction.

⋙ Proof of c). $\textsc{Imp}(\mathsf{DTGD}_{[\mathsf{df}]} \cup \mathsf{DEGD})$ is already EXPTIME-hard if restricted to only node-generating dtgds of Type (G2) and degds of Type (E4) as input. We start with the basic idea of the reduction and provide the details afterwards.

» Idea. The following construction resembles that in the proof of b). Whereas there are up to four nodes generated *conditionally* in the previous construction, now exactly four nodes $k, k_1, k_2, k^*$ are generated *unconditionally* in the beginning. Each of these nodes has a specific fact: $\texttt{Eval}$, $\texttt{Acc}$, $\texttt{Acc}_1$ or $\texttt{Acc}_2$. Later, some of these nodes are merged conditionally: a configuration is viewed as accepting if there is a node that represents it and where both $\texttt{Eval}$ and $\texttt{Acc}$ are present.

» Details. Dependency $\tau$ requires that, if there is a node representing the initial configuration, there is also a node that represents the initial configuration *and* contains $\texttt{Eval}$ and $\texttt{Acc}$.

$$\begin{aligned}\mathsf{body}(\tau) &\stackrel{\text{def}}{=} \mathcal{A}_{\text{succ}} \cup \mathcal{A}_{\text{alph}} \cup \mathcal{C}_{q_0,0,\alpha(w)}@\kappa\\ \mathsf{head}(\tau) &\stackrel{\text{def}}{=} (\mathcal{C}_{q_0,0,\alpha(w)} \cup \{\texttt{Eval}, \texttt{Acc}\})@\kappa^*\end{aligned}$$

Subset $\Sigma_1$ contains four dependencies $\sigma^{\text{init}}_{q,eval}$, $\sigma^{\text{init}}_{q,1}$, $\sigma^{\text{init}}_{q,2}$, $\sigma^{\text{init}}_{q,*}$ of Type (G2) for each state $q \in Q$ and each position $i \in \{0, \ldots, n+1\}$. Dependency $\sigma^{\text{init}}_{q,eval}$ is defined as follows.

$$\begin{aligned}\mathsf{body}(\sigma^{\text{init}}_{q,eval}) &\stackrel{\text{def}}{=} \mathcal{A}_{\text{succ}} \cup \mathcal{A}_{\text{alph}}\\ \mathsf{head}(\sigma^{\text{init}}_{q,eval}) &\stackrel{\text{def}}{=} (\mathcal{C}_{q,i,\boldsymbol{u}} \cup \{\texttt{Eval}\})@\kappa\end{aligned}$$

The other dependencies are defined analogously (with $\texttt{Acc}_1$, $\texttt{Acc}_2$ or $\texttt{Acc}$ in the head).

Subset $\Sigma_2$ contains degds of Type (E4). First, for each accepting state $q$, there is a degd $\sigma^{\text{acc}}_{q,i,*}$ for each position $i \in \{0, \ldots, n+1\}$.

$$\begin{aligned}\mathsf{body}(\sigma^{\text{acc}}_{q,i,*}) &\stackrel{\text{def}}{=} (\mathcal{C}_{q,i,\boldsymbol{u}} \cup \{\texttt{Eval}\})@\kappa \cup (\mathcal{C}_{q,i,\boldsymbol{u}} \cup \{\texttt{Acc}\})@\kappa^*\\ \mathsf{head}(\sigma^{\text{acc}}_{q,i,*}) &\stackrel{\text{def}}{=} (\kappa = \kappa^*)\end{aligned}$$

These dependencies require the $\texttt{Eval}$- and $\texttt{Acc}$-representations of the same configuration to be merged. Second, for each state $q$, each symbol $a_h \in A$, each position $i \in \{0, \ldots, n+1\}$ and both $j \in \{1, 2\}$, a degd $\sigma^{\text{trans}}_{q,h,i,j}$ of Type (E4).

$$\begin{aligned}\mathsf{body}(\sigma^{\text{trans}}_{q,h,i,j}) &\stackrel{\text{def}}{=} \mathcal{A}_{\text{succ}} \cup (\mathcal{C}_{\hat{q},i+1,\hat{\boldsymbol{u}}} \cup \{\texttt{Eval}, \texttt{Acc}\})@\mu\\ &\quad\cup\ (\mathcal{C}_{q,i,\boldsymbol{u}} \cup \{\texttt{Eval}\})@\kappa \cup (\mathcal{C}_{q,i,\boldsymbol{u}} \cup \{\texttt{Acc}_j\})@\kappa_j\\ \mathsf{head}(\sigma^{\text{trans}}_{q,h,i,j}) &\stackrel{\text{def}}{=} (\kappa = \kappa_j)\end{aligned}$$

Third, there is a degd $\sigma^{\text{acc}}_{q,i}$ of Type (E4) for each universal state $q$ and each position $i \in \{0, \dots, n+1\}$.

$$\mathsf{body}(\sigma^{\text{acc}}_{q,i}) \stackrel{\text{def}}{=} (\mathcal{C}_{q,i,\boldsymbol{u}} \cup \{\mathtt{Eval}, \mathtt{Acc}_1, \mathtt{Acc}_2\})@\kappa \cup (\mathcal{C}_{q,i,\boldsymbol{u}} \cup \{\mathtt{Acc}\})@\kappa^*$$
$$\mathsf{head}(\sigma^{\text{acc}}_{q,i}) \stackrel{\text{def}}{=} (\kappa = \kappa^*)$$

Analogously, there are two degds $\sigma^{\text{acc}}_{q,i,1}$ and $\sigma^{\text{acc}}_{q,i,2}$ for each existential state. As before, the correctness of this construction can be proven with a straightforward induction.

⋙ Proof of d). $\textsc{Imp}(\mathsf{DTGD}_{[\mathsf{df}]} \cup \mathsf{DEGD})$ is already EXPTIME-hard if restricted to only node-generating dtgds of Type (G2) and (G3) and degds of Type (E3) as input. We start with the basic idea of the reduction and provide the details afterwards.

» Idea. Different from the previous approaches, this time nodes are generated more hesitantly. In the beginning, only nodes for configurations with accepting states are generated. Then, nodes for *predecessor* configurations are generated with acceptance markers $\mathtt{Acc}_1$ or $\mathtt{Acc}_2$, respectively. Each configuration is effectively represented by at most one node, due to degds of Type (E3). In particular, both acceptance markers are guaranteed to be on the same node—if present—and thus can be referenced by a node-generating dtgd of Type (G3) for the conditional marking with $\mathtt{Acc}$.

» Details. Dependency $\tau$ requires that there is a node representing the initial configuration and that it is marked as accepting.

$$\mathsf{body}(\tau) \stackrel{\text{def}}{=} \mathcal{A}_{\text{succ}} \cup \mathcal{A}_{\text{alph}}$$
$$\mathsf{head}(\tau) \stackrel{\text{def}}{=} (\mathcal{C}_{q_0,0,w} \cup \{\mathtt{Acc}\})@\kappa$$

Subset $\Sigma_1$ contains the same dependencies as in the proof of a), *albeit not for all but only for accepting states*, that is, for every $q \in F$ and every position $i \in \{0, \dots, n+1\}$.

Subset $\Sigma_2$ contains three types of dependencies. First, for each state $q$, each symbol $a_h \in A$, each position $i \in \{0, \dots, n+1\}$ and both $j \in \{1, 2\}$, there is a dependency $\sigma^{\text{trans}}_{q,h,i,j}$ as defined next.

$$\mathsf{body}(\sigma^{\text{trans}}_{q,h,i,j}) \stackrel{\text{def}}{=} (\mathcal{C}_{\hat{q},i+1,\hat{\boldsymbol{u}}} \cup \{\mathtt{Acc}\})@\mu$$
$$\mathsf{head}(\sigma^{\text{trans}}_{q,h,i,j}) \stackrel{\text{def}}{=} (\mathcal{C}_{q,i,\boldsymbol{u}} \cup \{\mathtt{Acc}_j\})@\kappa$$

Second, there are dependencies to mark universal and existential states accepting. For every universal state $q$ and every position $i \in \{0, \dots, n+1\}$, there is dependency $\sigma^{\text{acc}}_{q,i}$ as follows.

$$\mathsf{body}(\sigma^{\text{acc}}_{q,i}) \stackrel{\text{def}}{=} (\mathcal{C}_{q,i,\boldsymbol{u}} \cup \{\mathtt{Acc}_1, \mathtt{Acc}_2\})@\mu$$
$$\mathsf{head}(\sigma^{\text{acc}}_{q,i}) \stackrel{\text{def}}{=} (\mathcal{C}_{q,i,\boldsymbol{u}} \cup \{\mathtt{Acc}\})@\kappa$$

Dependencies $\sigma^{\text{acc}}_{q,i,1}$ and $\sigma^{\text{acc}}_{q,i,2}$ for every existential state $q$ and every position $i$ are defined analogously.

Third, for every state $q$ and every position $i \in \{0, \ldots, n+1\}$, there is a degd $\sigma^{\text{merge}}_{q,i}$ of Type (E3) that requires representations of the same configuration to be located on the same node.

$$\mathsf{body}(\sigma^{\text{merge}}_{q,i}) \stackrel{\text{def}}{=} \mathcal{C}_{q,i,\boldsymbol{u}}@\kappa \cup \mathcal{C}_{q,i,\boldsymbol{u}}@\mu$$
$$\mathsf{head}(\sigma^{\text{merge}}_{q,i}) \stackrel{\text{def}}{=} (\kappa = \mu)$$

The correctness of this construction can be proven, again, by induction.

In each case, the set $\Sigma$ of dependencies obeys the stated syntactical restrictions and implies dependency $\tau$ if and only if Turing machine $\mathcal{M}$ accepts input $w$. □

This concludes our complexity-theoretical investigation of the implication problem for distribution dependencies.

## 5.3. Related work and bibliographical remarks

In this chapter, we have introduced and studied distribution dependencies as a means to specify placement strategies—or properties thereof—for distributed relational data *declaratively*.

Naturally, distribution dependencies are not the only approach to combine typical questions in distributed data management with the extensive literature on constraints. Two approaches known to the author, *data exchange* and *Webdamlog*, share particular similarities, which we briefly discuss next.

The *data exchange* setting [ABLM14], for instance, can be viewed as the use of distribution dependencies on (a global instance and) exactly one local instance. Here, the focus lies on the computation of solutions, which are usually determined via the chase. On the one hand the 'distribution' in this setting is hence rather restricted, while on the other hand the allowed use of dependencies is sometimes more generous.

On the contrary, the Datalog-dialect *Webdamlog*—from which we borrow the concept of annotated atoms—, has been designed explicitly to capture the distribution of relational facts over *multiple* nodes. While our investigation addresses the implication problem, the research on Webdamlog centred around a study on the expressivity of some fragments, yielding a hierarchy [ABGA11], and the implementation of systems that support the dialect [AAM+13, AAS13, MSAM15]. Furthermore, not only the problems studied are different but there are also significant differences in the interpretation (or evaluation) of Webdamlog rules compared to distribution dependencies. In particular, Webdamlog rules are evaluated under a fixpoint operator,

which even allows facts to vanish over time, whereas, for distribution dependencies, the chase behaves monotonically—modulo the unification of terms via degds.

Ignoring any particular focus on *distributed* relational data, there is a long tradition in studying the implication problem, using the chase in particular, for various classes of constraints. The results in Section 5.2 contribute to this tradition and it is therefore natural to ask how they relate to previous results. Below, we try to shed light on these relationships.

Motivated by their practical importance, constraints for relational databases have been studied early on. Starting with functional dependencies, introduced by Codd [Cod71], several other types of constraints emerged. Many of them have later been subsumed by tuple- and equality-generating dependencies [BV84] (and embedded dependencies [Fag82]). Similarly, the chase procedure has been used for simpler types of constraints first, then *named* by Maier, Mendelzon and Sagiv [MMS79] and eventually extended to tgds and egds [BV84].[7]

**Distribution dependencies as 'classical' dependencies.** The use of node variables in annotated atoms, like $\kappa$ in $R(x, y)@\kappa$, fits seamlessly into the (standard) model of relational databases. The purpose of the notational deviation is solely meant to emphasise the different roles of attributes—and to clarify the respective syntactical restrictions studied.

In general, a distribution dependency over schema $\mathcal{S}$ is straightforwardly translated into a classical dependency over schema $\mathcal{S} \cup \mathcal{S}'$, where $\mathcal{S}$ is used for 'global' facts while $\mathcal{S}'$ is used for 'local' facts. For each relation symbol $R$ in $\mathcal{S}$ with arity $k$, there is a fresh relation symbol $R'$ in $\mathcal{S}'$ with arity $k+1$. A classical atom $R(x_1, \ldots, x_k)$ is then, using a node variable $\kappa$, translated into the 'localised' atom $R'(\kappa, x_1, \ldots, x_k)$. For instance, the classical dtgd $R(x, y), S(x)@\kappa \to R(x, y)@\kappa$ yields $R(x, y), S'(\kappa, x) \to R'(\kappa, x, y)$. This corresponds to the application of $\mathsf{ext}_\kappa$, with an additional renaming of the relation.

Due to this translation, our study in Section 5.2 implicitly provides complexity-theoretical results on certain fragments of classical dependencies. Notable properties of these fragments are:

1. the bipartition of the variable sets and their positions (attributes),
2. the restricted use of existential quantification,
3. the bounded arity of atoms (mostly) and
4. the bounded context of certain node variables.

All these aspects with exception of the last one, to the best of the author's knowledge,

[7] More details on the historical background of constraints and the chase can be found in overview articles and standard works [FV84, LÖ18, AHV95], for instance.

have been addressed already in earlier research, even though in different combinations and with varying degree.

First, with its distinction of node and data variables, distributed dependencies can be regarded as a weak form of 'typed' dependencies, where each variable may occur at most once in each atom and only in the same position for each relation. While node variables may only occur—once—in the first position of $\mathcal{S}'$-atoms, data variables may appear in all other positions, even repeatedly. Although the distinction between typed and untyped dependencies does not always help to lower complexity (see Theorem* 5.2.1, for instance), it is helpful in the fragments studied above.

Second, existential quantification is allowed only in a very limited form in *data-full* distribution dependencies: existential variables may only occur in the first position in atoms over $\mathcal{S}'$. In the literature, restrictions to the use of existential variables are a common approach to yield decidable fragments of tgds (and egds). Often, however, the criteria there are more involved and may even depend on the interaction with other dependencies in a given set of dependencies. For data-full distribution dependencies with (weakly) bounded context, on the contrary, we have established a simple syntactic restriction *per dependency*.

Next, we show that the restrictions underlying other prominent fragments are orthogonal to those studied in Section 5.2.

**Example 5.3.1 (Orthogonal restrictions).** The set $\Sigma = \{\sigma_1, \sigma_2\}$ of distribution dependencies, consisting of

$$\sigma_1 \stackrel{\text{def}}{=} R(x)@\kappa \to T(x)@\mu \text{ and,}$$
$$\sigma_2 \stackrel{\text{def}}{=} T(x)@\kappa, T(y)@\kappa, T(z)@\mu, T(w)@\mu \to U(x,y,z,w)@\kappa$$

has bounded context. According to the translation above, this set induces a set $\Sigma' = \{\sigma'_1, \sigma'_2\}$ of classical tgds, where

$$\sigma'_1 \stackrel{\text{def}}{=} R(\kappa, x) \to T(\mu, x) \text{ and,}$$
$$\sigma'_2 \stackrel{\text{def}}{=} T(\kappa, x), T(\kappa, y), T(\mu, z), T(\mu, w) \to U(\kappa, x, y, z, w),$$

which is neither *sticky* nor *weakly guarded* nor *warded*. Set $\Sigma'$ is not sticky [CGP10] because the 'marked' variable $\mu$ occurs more than once in $\sigma'_2$. It is not weakly guarded [CGK13] because a single atom cannot contain both variables $\kappa$ and $\mu$, which occur in 'affected' positions of $\sigma'_2$. It is not warded [GP15, BGPS19] because the 'dangerous' variable $\kappa$ appears in more than one atom in the body of $\sigma'_2$.

Furthermore, orthogonality to the restriction of *weak acyclicity* is witnessed by another set of distribution dependencies. For instance, by the set that contains

$$\tau_1 \stackrel{\text{def}}{=} R(x,y)@\kappa \to S(x,y)@\mu \text{ and}$$
$$\tau_2 \stackrel{\text{def}}{=} S(x,x)@\kappa \to R(x,x)@\mu,$$

two data-full distribution dependencies with bounded context. This set is not weakly acyclic because its 'dependency graph' contains a cycle through special edges between the node attributes of both relations [FKMP05, Definition 3.7]. ■

Lastly, to achieve the lower complexity bounds of NP and PSPACE, we bound the amount of data that can be associated with each node and, in particular, the amount of data that can be referenced from different nodes. The arity bound is a first—and usual—way for such a restriction whose influence becomes visible also in other problems (containment, parallel correctness and more). Yet, alone it is insufficient when existential variables are allowed and, hence, complemented by the context bounds in this thesis.

**Bibliographical remarks.** The complexity results in Section 5.2 are joint work of the author with Frank Neven and Thomas Schwentick and accepted for publication [GNS20].[8] Moreover, in their work, the authors have studied distribution dependencies with constants and comparison atoms (like $\neq, \leq, <$ and so on) too. This may be deemed practically interesting as it allows, for instance, a more direct modelling of range-based partitionings.

For these extended distribution dependencies, the fragments considered in Section 5.2 have a similar worst-case complexity. For dependencies with bounded context, complexity rises from NP to $\Pi_2^p$, while the PSPACE- and EXPTIME-bounds carry over, as do the proof ideas. Basically, there are only two differences. First, chase sequences may fail because of violated comparison atoms. Second, *multiple* chase sequences have to be considered. This accounts for constants and the 'relative' order of terms (constants and variables). The latter results in an additional—universally quantified—extra input for our NP-fragment while it does not affect the higher bounds.

[8] Bas Ketsman also contributed to an early precursor of this work, which considered conjunctive queries over distributed databases with (classical) functional and/or inclusion dependencies for the local and global databases. The expressivity of this formalisation is rather restricted. For instance, parallel-completeness can only be formulated for conjunctive queries that are 'guarded', that is, have an atom that comprises all variables. This excludes even simple 'cross products' like the full query $H(x, y) \leftarrow R(x), S(y)$.

# 6. Conclusion

This thesis provides a detailed analysis of three problems on automatic reasoning about relational data that is distributed in a network of servers and the evaluation of queries in this network, with a strong emphasis on complexity theoretical aspects: *parallel correctness*, *parallel-correctness transfer* and the *implication* problem.

Since overviews of the results can be found in the corresponding chapters, we summarise the author's contributions presented in this thesis only coarsely. Then, we point out open problems and further directions.

## 6.1. Overview of results

Our study of parallel correctness in Chapter 3 starts with the $\Pi_2^p$-completeness result for the equivalent parallel completeness problem on unions of conjunctive queries. Then, for queries that additionally allow negation, we show that the complexity remains unchanged only if the arity of relations is bounded in advance. In the general case, it becomes coNEXPTIME-complete—a result that also influenced the knowledge about the containment problem for the classes $\mathsf{CQ}^\neg$ and $\mathsf{UCQ}^\neg$. After that, we consider fragments like full queries and polarised queries. For full queries, we obtain a lower bound of coNP and argue that the given reduction connects complexity results between containment and parallel correctness in a more general way, for many query classes with suitable closure properties. For polarised queries, we provide characterisations of parallel soundness and parallel completeness and show that these problems are $\Pi_2^p$-complete again, independent of *a priori* bounds on the arity of relations.

Then, in Chapter 4, our study continues with parallel-correctness transfer, which abstracts from specific policies to families of policies induced by another query. We show that, for unions of conjunctive queries, this problem is $\Pi_3^p$-complete in general. For the 'semantic' fragment of strongly minimal queries, we prove it to be $\Pi_2^p$-complete if at least one of the input queries may contain disequalities. Lastly, for polarised queries with negation we provide a characterisation and a matching $\Pi_3^p$-completeness result for parallel-*soundness* transfer.

Finally, we study the implication problem for distribution dependencies in Chapter 5. There, we consider several fragments of data-full dependencies, based on comparatively simple syntactic restrictions, which rely on the notion of a variable *context*. For dependencies with *bounded context*, we prove the implication problem to be

NP-complete, while it is PSPACE-complete for the slightly more generous class of dependencies with *weakly bounded context*. Eventually, we show for most of the remaining possible extensions that they lead to an EXPTIME-complete implication problem.

## 6.2. Open problems and further directions

Even in combination with other existing work known to the author, this thesis is far from giving a complete picture of the complexity landscape of the problems under consideration; several questions remain open. Some reside on a more technical level, others on more abstract levels.

Technically, several questions result from the 'gaps' in the previous overview, even for the one specific setting considered in this thesis: relational databases in the MPC model, evaluated in a naïve fashion in just one round. For instance, while we do provide a characterisation of parallel-soundness transfer for polarised queries, we do *not* provide a characterisation of parallel-completeness transfer between them. Specifically, we do not know how the policy-dependent notion of $\mathbb{P}$-minimal valuations may translate into a notion that depends on two queries only. More generally, we do not know any (non-trivial) characterisations or complexity bounds for parallel-correctness transfer and its variants for the class $\mathsf{UCQ}^\neg$ in general. Also, for distribution dependencies, there are certain combinations of fragments (say, data-full dependencies of Types (G2) and (E3)) whose exact complexity is not determined.

On a higher level, there are also many possibilities for variations of the setting. We have mentioned some already in Sections 3.4 and 4.4 on related work: different query semantics (bag instead of set semantics), different evaluation mechanisms (in ordered networks), different query languages (spanners). Furthermore, these works also pointed to other, related problems: the question of parallelisability (for conjunctive queries under bag semantics) or a more general variant of parallel correctness (like splittability for spanners), where a different query may be evaluated locally, for instance.

Other aspects—not or only barely considered so far—are presumably of theoretical and practical interest. In this thesis, we have considered the distributed evaluation in *one* round. In practice, computations usually span *multiple* rounds. While it is possible to generalise some of the problems studied here by simply 'chaining' them if the number of rounds is known in advance, more elaborate approaches and different questions are likely to arise—especially if there is no bound on the number of rounds. Also, a closer look on distribution policies might be worthwhile. In particular, it seems interesting to compare policies with respect to the load they produce or the amount of communication that is necessary to satisfy them, coming from some distribution. Such quantitative aspects are probably of high practical relevance.

# Bibliography


[AAM+13] Abiteboul, Serge ; Antoine, Émilien ; Miklau, Gerome ; Stoyanovich, Julia ; Testard, Jules: Rule-based application development using Webdamlog. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, 2013, 965–968

[AAS13] Abiteboul, Serge ; Antoine, Émilien ; Stoyanovich, Julia: The Webdamlog System Managing Distributed Knowledge on the Web. In: *CoRR* abs/1304.4187 (2013). http://arxiv.org/abs/1304.4187

[ABGA11] Abiteboul, Serge ; Bienvenu, Meghyn ; Galland, Alban ; Antoine, Émilien: A rule-based language for web data management. In: *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, 2011, 293–304

[ABLM14] Arenas, Marcelo ; Barceló, Pablo ; Libkin, Leonid ; Murlak, Filip: *Foundations of Data Exchange*. Cambridge University Press, 2014 http://www.cambridge.org/9781107016163. – ISBN 9781107016163

[AGK+15] Ameloot, Tom J. ; Geck, Gaetano ; Ketsman, Bas ; Neven, Frank ; Schwentick, Thomas: Parallel-Correctness and Transferability for Conjunctive Queries. In: *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, 2015, 47–58

[AGK+16] Ameloot, Tom J. ; Geck, Gaetano ; Ketsman, Bas ; Neven, Frank ; Schwentick, Thomas: Data partitioning for single-round multi-join evaluation in massively parallel systems. In: *SIGMOD Record* 45 (2016), Nr. 1, 33–40. http://dx.doi.org/10.1145/2949741.2949750. – DOI 10.1145/2949741.2949750

[AGK+17a] Ameloot, Tom J. ; Geck, Gaetano ; Ketsman, Bas ; Neven, Frank ; Schwentick, Thomas: Parallel-Correctness and Transferability for Conjunctive Queries. In: *J. ACM* 64 (2017), Nr. 5, 36:1–36:38. http://dx.doi.org/10.1145/3106412. – DOI 10.1145/3106412

[AGK+17b] Ameloot, Tom J. ; Geck, Gaetano ; Ketsman, Bas ; Neven, Frank ; Schwentick, Thomas: Reasoning on data partitioning for single-round multi-join evaluation in massively parallel systems. In: *Commun. ACM* 60 (2017), Nr. 3, 93–100. http://dx.doi.org/10.1145/3041063. – DOI 10.1145/3041063

[AHV95] Abiteboul, Serge ; Hull, Richard ; Vianu, Victor: *Foundations of Databases*. Addison-Wesley, 1995 http://webdam.inria.fr/Alice/. – ISBN 0–201–53771–0

[AU10] Afrati, Foto N. ; Ullman, Jeffrey D.: Optimizing joins in a map-reduce environment. In: *EDBT 2010, 13th International Conference on Extending Database Technology, Lausanne, Switzerland, March 22-26, 2010, Proceedings*, 2010, 99–110

[BGPS19] Berger, Gerald ; Gottlob, Georg ; Pieris, Andreas ; Sallinger, Emanuel: The Space-Efficient Core of Vadalog. In: *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019.*, 2019, 270–284

[BKS13] Beame, Paul ; Koutris, Paraschos ; Suciu, Dan: Communication steps for parallel query processing. In: *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013*, 2013, 273–284

[BKS14] Beame, Paul ; Koutris, Paraschos ; Suciu, Dan: Skew in parallel query processing. In: *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, 2014, 212–223

[BKS17] Beame, Paul ; Koutris, Paraschos ; Suciu, Dan: Communication Steps for Parallel Query Processing. In: *J. ACM* 64 (2017), Nr. 6, 40:1–40:58. http://dx.doi.org/10.1145/3125644. – DOI 10.1145/3125644

[BV81] Beeri, Catriel ; Vardi, Moshe Y.: The Implication Problem for Data Dependencies. In: *Automata, Languages and Programming, 8th Colloquium, Acre (Akko), Israel, July 13-17, 1981, Proceedings*, 1981, 73–85

[BV84] Beeri, Catriel ; Vardi, Moshe Y.: A Proof Procedure for Data Dependencies. In: *J. ACM* 31 (1984), Nr. 4, 718–741. http://dx.doi.org/10.1145/1634.1636. – DOI 10.1145/1634.1636

[CGK13] Calì, Andrea ; Gottlob, Georg ; Kifer, Michael: Taming the Infinite Chase: Query Answering under Expressive Relational Constraints. In: *J. Artif. Intell. Res.* 48 (2013), 115–174. http://dx.doi.org/10.1613/jair.3873. – DOI 10.1613/jair.3873

[CGP10] Calì, Andrea ; Gottlob, Georg ; Pieris, Andreas: Advanced Processing for Ontological Queries. In: *PVLDB* 3 (2010), Nr. 1, 554–565. http://dx.doi.org/10.14778/1920841.1920912. – DOI 10.14778/1920841.1920912

[CLM81] Chandra, Ashok K. ; Lewis, Harry R. ; Makowsky, Johann A.: Embedded Implicational Dependencies and their Inference Problem. In: *Proceedings of the 13th Annual ACM Symposium on Theory of Computing, May 11-13, 1981, Milwaukee, Wisconsin, USA*, 1981, 342–354

[CM77] Chandra, Ashok K. ; Merlin, Philip M.: Optimal Implementation of Conjunctive Queries in Relational Data Bases. In: *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*, 1977, 77–90

[Cod70] Codd, E. F.: A Relational Model of Data for Large Shared Data Banks. In: *Commun. ACM* 13 (1970), Nr. 6, 377–387. http://dx.doi.org/10.1145/362384.362685. – DOI 10.1145/362384.362685

[Cod71] Codd, E. F.: Further Normalization of the Data Base Relational Model. In: *IBM Research Report, San Jose, California* RJ909 (1971)

[DG08] Dean, Jeffrey ; Ghemawat, Sanjay: MapReduce: simplified data processing on large clusters. In: *Commun. ACM* 51 (2008), Nr. 1, 107–113. http://dx.doi.org/10.1145/1327452.1327492. – DOI 10.1145/1327452.1327492

[DGS+90] DeWitt, David J. ; Ghandeharizadeh, Shahram ; Schneider, Donovan A. ; Bricker, Allan ; Hsiao, Hui-I ; Rasmussen, Rick: The Gamma Database Machine Project. In: *IEEE Trans. Knowl. Data Eng.* 2 (1990), Nr. 1, 44–62. http://dx.doi.org/10.1109/69.50905. – DOI 10.1109/69.50905

[DKM+19] Doleschal, Johannes ; Kimelfeld, Benny ; Martens, Wim ; Nahshon, Yoav ; Neven, Frank: Split-Correctness in Information Extraction. In: *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019.*, 2019, 149–163

[Fag82] Fagin, Ronald: Horn clauses and database dependencies. In: *J. ACM* 29 (1982), Nr. 4, 952–985. http://dx.doi.org/10.1145/322344.322347. – DOI 10.1145/322344.322347

[FKMP05] Fagin, Ronald ; Kolaitis, Phokion G. ; Miller, Renée J. ; Popa, Lucian: Data exchange: semantics and query answering. In: *Theor. Comput. Sci.* 336 (2005), Nr. 1, 89–124. http://dx.doi.org/10.1016/j.tcs.2004.10.033. – DOI 10.1016/j.tcs.2004.10.033

[FKRV15] Fagin, Ronald ; Kimelfeld, Benny ; Reiss, Frederick ; Vansummeren, Stijn: Document Spanners: A Formal Approach to Information Extraction. In: *J. ACM* 62 (2015), Nr. 2, 12:1–12:51. http://dx.doi.org/10.1145/2699442. – DOI 10.1145/2699442

[FKT86] Fushimi, Shinya ; Kitsuregawa, Masaru ; Tanaka, Hidehiko: An Overview of The System Software of A Parallel Relational Database Machine GRACE. In: *VLDB'86 Twelfth International Conference on Very Large Data Bases, August 25-28, 1986, Kyoto, Japan, Proceedings.*, 1986, 209–219

[FV84] Fagin, Ronald ; Vardi, Moshe Y.: The Theory of Data Dependencies - An Overview. In: *Automata, Languages and Programming, 11th Colloquium, Antwerp, Belgium, July 16-20, 1984, Proceedings*, 1984, 1–22

[GKNS16] Geck, Gaetano ; Ketsman, Bas ; Neven, Frank ; Schwentick, Thomas: Parallel-Correctness and Containment for Conjunctive Queries with Union and Negation. In: *19th International Conference on Database Theory, ICDT 2016, Bordeaux, France, March 15-18, 2016*, 2016, 9:1–9:17

[GKNS19] Geck, Gaetano ; Ketsman, Bas ; Neven, Frank ; Schwentick, Thomas: Parallel-Correctness and Containment for Conjunctive Queries with Union and Negation. In: *ACM Trans. Comput. Logic* 20 (2019), Juni, Nr. 3, 18:1–18:24. http://dx.doi.org/10.1145/3329120. – DOI 10.1145/3329120. – ISSN 1529–3785

[GMSV93] Gaifman, Haim ; Mairson, Harry G. ; Sagiv, Yehoshua ; Vardi, Moshe Y.: Undecidable Optimization Problems for Database Logic Programs. In: *J. ACM* 40 (1993), Nr. 3, 683–713. http://dx.doi.org/10.1145/174130.174142. – DOI 10.1145/174130.174142

[GNS20] Geck, Gaetano ; Neven, Frank ; Schwentick, Thomas: Distribution Constraints: The Chase for Distributed Data. In: *Accepted for the 23rd*

*International Conference on Database Theory, ICDT 2020, Copenhagen, Denmark, March 30-April 2, 2020*, 2020

[GP15] Gottlob, Georg ; Pieris, Andreas: Beyond SPARQL under OWL 2 QL Entailment Regime: Rules to the Rescue. In: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 2015, 2999–3007

[KAK18] Ketsman, Bas ; Albarghouthi, Aws ; Koutris, Paraschos: Distribution Policies for Datalog. In: *21st International Conference on Database Theory, ICDT 2018, March 26-29, 2018, Vienna, Austria*, 2018, 17:1–17:22

[KBS16] Koutris, Paraschos ; Beame, Paul ; Suciu, Dan: Worst-Case Optimal Algorithms for Parallel Query Processing. In: *19th International Conference on Database Theory, ICDT 2016, Bordeaux, France, March 15-18, 2016*, 2016, 8:1–8:18

[KNV18] Ketsman, Bas ; Neven, Frank ; Vandevoort, Brecht: Parallel-Correctness and Transferability for Conjunctive Queries under Bag Semantics. In: *21st International Conference on Database Theory, ICDT 2018, March 26-29, 2018, Vienna, Austria*, 2018, 18:1–18:16

[KS11] Koutris, Paraschos ; Suciu, Dan: Parallel evaluation of conjunctive queries. In: *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, 2011, 223–234

[KS17] Ketsman, Bas ; Suciu, Dan: A Worst-Case Optimal Multi-Round Algorithm for Parallel Computation of Conjunctive Queries. In: *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, 2017, 417–428

[LM07] Leclère, Michel ; Mugnier, Marie-Laure: Some Algorithmic Improvements for the Containment Problem of Conjunctive Queries with Negation. In: *Database Theory - ICDT 2007, 11th International Conference, Barcelona, Spain, January 10-12, 2007, Proceedings*, 2007, 404–418

[LÖ18] Liu, Ling (Hrsg.) ; Özsu, M. T. (Hrsg.): *Encyclopedia of Database Systems, Second Edition*. Springer, 2018. http://dx.doi.org/10.1007/978-1-4614-8265-9. http://dx.doi.org/10.1007/978-1-4614-8265-9. – ISBN 978–1–4614–8266–6

[Mar09] Marnette, Bruno: Generalized schema-mappings: from termination to tractability. In: *Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009, June 19 - July 1, 2009, Providence, Rhode Island, USA*, 2009, 13–22

[MMS79] Maier, David ; Mendelzon, Alberto O. ; Sagiv, Yehoshua: Testing Implications of Data Dependencies. In: *ACM Trans. Database Syst.* 4 (1979), Nr. 4, 455–469. http://dx.doi.org/10.1145/320107.320115. – DOI 10.1145/320107.320115

[MSAM15] Moffitt, Vera Z. ; Stoyanovich, Julia ; Abiteboul, Serge ; Miklau, Gerome: Collaborative Access Control in WebdamLog. In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, 2015, 197–211

[MST12] Mugnier, Marie-Laure ; Simonet, Geneviève ; Thomazo, Michaël: On the complexity of entailment in existential conjunctive first-order logic with atomic negation. In: *Inf. Comput.* 215 (2012), 8–31. http://dx.doi.org/10.1016/j.ic.2012.03.001. – DOI 10.1016/j.ic.2012.03.001

[Nic78] Nicolas, Jean-Marie: First Order Logic Formalization for Functional, Multivalued and Mutual Dependencies. In: *Proceedings of the 1978 ACM SIGMOD International Conference on Management of Data, Austin, Texas, USA, May 31 - June 2, 1978*, 1978, 40–46

[NSSV19] Neven, Frank ; Schwentick, Thomas ; Spinrath, Christopher ; Vandevoort, Brecht: Parallel-Correctness and Parallel-Boundedness for Datalog Programs. In: *22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal*, 2019, 14:1–14:19

[One12] Onet, Adrian C.: *The chase procedure and its applications*. https://spectrum.library.concordia.ca/974476/. Version: June 2012. – PhD Thesis

[ÖV11] Özsu, M. T. ; Valduriez, Patrick: *Principles of Distributed Database Systems, Third Edition*. Springer, 2011. http://dx.doi.org/10.1007/978-1-4419-8834-8. http://dx.doi.org/10.1007/978-1-4419-8834-8. – ISBN 978–1–4419–8833–1

[SB18] Schwarzkopf, Malte ; Bailis, Peter: Research for practice: cluster scheduling for datacenters. In: *Commun. ACM* 61 (2018), Nr. 5, 50–53. http://dx.doi.org/10.1145/3154011. – DOI 10.1145/3154011

[SCH+18] Samwel, Bart ; Cieslewicz, John ; Handy, Ben ; Govig, Jason ; Venetis, Petros ; Yang, Chanjun ; Peters, Keith ; Shute, Jeff ; Tenedorio, Daniel ; Apte, Himani ; Weigel, Felix ; Wilhite, David ; Yang, Jiacheng ; Xu, Jun ; Li, Jiexing ; Yuan, Zhan ; Chasseur, Craig ; Zeng, Qiang ; Rae, Ian ; Biyani, Anurag ; Harn, Andrew ; Xia, Yang ; Gubichev, Andrey ; El-Helw, Amr ; Erling, Orri ; Yan, Zhepeng ; Yang, Mohan ; Wei, Yiqun ; Do, Thanh ; Zheng, Colin ; Graefe, Goetz ; Sardashti, Somayeh ; Aly, Ahmed M. ; Agrawal, Divy ; Gupta, Ashish ; Venkataraman, Shivakumar: F1 Query: Declarative Querying at Scale. In: *PVLDB* 11 (2018), Nr. 12, 1835–1848. http://dx.doi.org/10.14778/3229863.3229871. – DOI 10.14778/3229863.3229871

[SKN18] Sundarmurthy, Bruhathi ; Koutris, Paraschos ; Naughton, Jeffrey F.: Exploiting Data Partitioning To Provide Approximate Results. In: *Proceedings of the 5th ACM SIGMOD Workshop on Algorithms and Systems for MapReduce and Beyond, BeyondMR@SIGMOD 2018, Houston, TX, USA, June 15, 2018*, 2018, 5:1–5:5

[SKS05] Silberschatz, Abraham ; Korth, Henry F. ; Sudarshan, S.: *Database System Concepts, 5th Edition*. McGraw-Hill Book Company, 2005. – ISBN 978–0–07–295886–7

[SSS10] Schweikardt, Nicole ; Schwentick, Thomas ; Segoufin, Luc: Algorithms and Theory of Computation Handbook. Version: 2010. http://dl.acm.org/citation.cfm?id=1882723.1882742. Chapman & Hall/CRC, 2010. – ISBN 978–1–58488–820–8, Kapitel Database Theory: Query Languages, 19–19

[Sto76] Stockmeyer, Larry J.: The Polynomial-Time Hierarchy. In: *Theor. Comput. Sci.* 3 (1976), Nr. 1, 1–22. http://dx.doi.org/10.1016/0304-3975(76)90061-X. – DOI 10.1016/0304–3975(76)90061–X

[SVS+13] Shute, Jeff ; Vingralek, Radek ; Samwel, Bart ; Handy, Ben ; Whipkey, Chad ; Rollins, Eric ; Oancea, Mircea ; Littlefield, Kyle ; Menestrina, David ; Ellner, Stephan ; Cieslewicz, John ; Rae, Ian ; Stancescu, Traian ; Apte, Himani: F1: A Distributed SQL Database That Scales. In: *PVLDB* 6 (2013), Nr. 11,

1068–1079. http://dx.doi.org/10.14778/2536222.2536232. – DOI 10.14778/2536222.2536232

[Ull00] Ullman, Jeffrey D.: Information integration using logical views. In: *Theor. Comput. Sci.* 239 (2000), Nr. 2, 189–210. http://dx.doi.org/10.1016/S0304-3975(99)00219-4. – DOI 10.1016/S0304–3975(99)00219–4

[Val90] Valiant, Leslie G.: A Bridging Model for Parallel Computation. In: *Commun. ACM* 33 (1990), Nr. 8, 103–111. http://dx.doi.org/10.1145/79173.79181. – DOI 10.1145/79173.79181

[XRZ+13] Xin, Reynold S. ; Rosen, Josh ; Zaharia, Matei ; Franklin, Michael J. ; Shenker, Scott ; Stoica, Ion: Shark: SQL and rich analytics at scale. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, 2013, 13–24

[ZBS15] Zamanian, Erfan ; Binnig, Carsten ; Salama, Abdallah: Locality-aware Partitioning in Parallel Database Systems. In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, 2015, 17–30

[ZCF+10] Zaharia, Matei ; Chowdhury, Mosharaf ; Franklin, Michael J. ; Shenker, Scott ; Stoica, Ion: Spark: Cluster Computing with Working Sets. In: *2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud'10, Boston, MA, USA, June 22, 2010*, 2010

# Appendix

## A. Missing proofs

Proofs or arguments that are missing in the main text can be found here.

### A.1. Parallel correctness

**Lemma A.1.** For query $Q'$ and policy $\mathbb{P}'$ derived from query $Q$ and policy $\mathbb{P}$ as in the proof of the constant elimination lemma (Lemma 3.2.12), the following properties hold.

(P1) If $V'$ is a minimal valuation for $Q'$ and $V'(y_1, \ldots, y_s) = (c_1, \ldots, c_s)$, then $V'$ is also a minimal valuation for $Q$.

(P2) If $V$ is a minimal valuation for $Q$, then $V[y_1 \mapsto c_1, \ldots, y_s \mapsto c_s]$ is minimal for $Q'$.

*Proof.* We prove both implications by contradiction.

⋙ Property (P1). Let $V'$ be an arbitrary minimal valuation for query $Q'$ that maps variables $(y_1, \ldots, y_s)$ to the constants $(c_1, \ldots, c_s)$ of the original query $Q$. Since every variable in $Q$ also occurs in $Q'$, the mapping $V'$ is a valuation for query $Q$. It is also minimal as the following argument shows.

Assume, towards a contradiction, that $V'$ is not minimal for $Q$. Then, there exists a valuation $U$ for $Q$ such that $U <_Q^{\text{pos}} V'$ holds. This particularly implies $U(\mathsf{head}(Q)) = V'(\mathsf{head}(Q))$ and $U(\mathsf{pos}(Q)) \subsetneq V'(\mathsf{pos}(Q))$. From valuation $U$, we can derive another valuation $U'$ for $Q'$ in turn, $U' \stackrel{\text{def}}{=} U[y_1 \mapsto c_1, \ldots, y_s \mapsto c_s]$. This valuation, however, contradicts the minimality of $V'$ for $Q'$, as argued next.

First, all valuations agree on the head variables and on the required $\mathtt{Const}_i$-facts by definition of $U'$ and $U <_Q^{\text{pos}} V'$. More precisely, we have $U'(\mathsf{head}(Q)) = U(\mathsf{head}(Q)) = V'(\mathsf{head}(Q))$ and

$$U'(\{\mathtt{Const}_1(y_1), \ldots, \mathtt{Const}_s(y_s)\}) = V'(\mathtt{Const}_1(y_1), \ldots, \mathtt{Const}_s(y_s)\}).$$

Then, for the remaining atoms, we can infer

$$U'(\sigma(\mathsf{pos}(Q'))) = U'(\sigma(\mathsf{pos}(Q))) = U(\mathsf{pos}(Q)) \subsetneq V'(\mathsf{pos}(Q)) = V'(\mathsf{pos}(Q'))$$

because $U'(y_i) = U'(\sigma(c_i)) = c_i = U(c_i)$ for every $i \in \{1, \ldots, s\}$. Hence, $U' <^{\text{pos}}_{Q'} V'$, contradicting the choice of $V'$, as claimed.

Therefore, valuation $U$ cannot exist and valuation $V'$ is indeed minimal for query $Q$.

⋙ Property (P2). Let $V$ be an arbitary minimal valuation for query $Q$. Clearly, the mapping $V' \stackrel{\text{def}}{=} V[y_1 \mapsto c_1, \ldots, y_s \mapsto c_s]$ is a valuation for query $Q'$ because, by definition, $\mathsf{var}(Q') = \mathsf{var}(Q) \uplus \{y_1, \ldots, y_s\}$. Furthermore, valuation $V'$ is minimal, as shown next.

For a contradiction, assume that a valuation $U'$ for $Q'$ exists such that $U' <^{\text{pos}}_{Q'} V'$ holds. Since every $\mathtt{Const}_i$-fact occurs only once in $Q'$, this implies $U'(y_1, \ldots, y_s) = (c_1, \ldots, c_s) = V'(y_1, \ldots, y_s)$ and thus both valuations require the same sets of $\mathtt{Const}_i$-facts, $U'(\{\mathtt{Const}_1(y_1), \ldots, \mathtt{Const}_s(y_s)\} = V'(\{\mathtt{Const}_1(y_1), \ldots, \mathtt{Const}_s(y_s)\})$

With $U'(\mathsf{pos}(Q')) \subsetneq V'(\mathsf{pos}(Q'))$, this implies $U'(\sigma(\mathsf{pos}(Q))) \subsetneq V'(\sigma(\mathsf{pos}(Q)))$, which is equivalent to $U'(\mathsf{pos}(Q)) \subsetneq V(\mathsf{pos}(Q))$ because $U'(\sigma(c_i)) = V'(\sigma(c_i)) = c_i$ for every $i \in \{1, \ldots, s\}$. Since valuation $U'$ agrees with $V'$ and $V$ on the head variables of query $Q$, this results in $U' <^{\text{pos}}_{Q} V$, contradicting the choice of $V$.

Therefore, valuation $U'$ cannot exist and valuation $V'$ is minimal for query $Q'$.

Hence, both properties hold as claimed. □

**Lemma A.2 (Relation $\leq^{\mathbb{P}}_Q$ is a preorder).** Relation $\leq^{\mathbb{P}}_Q$ is a preorder for every polarised query $Q \in \mathsf{UCQ}^{\neg,\neq}_{\text{dom}}$ and every policy $\mathbb{P}$.

*Proof.* Fix a query $Q$ and a policy $\mathbb{P}$. By Definition 3.3.17 (Properties (1), (2) and (3a)), relation $\leq^{\mathbb{P}}_Q$ subsumes preorder $\leq_Q$ and is therefore reflexive.

It remains to show that relation $\leq^{\mathbb{P}}_Q$ is transitive. To this end, let $V_1$, $V_2$ and $V_3$ be arbitrary valuations for $Q$ such that $V_1 \leq^{\mathbb{P}}_Q V_2$ and $V_2 \leq^{\mathbb{P}}_Q V_3$ hold. We claim that then $V_1 \leq^{\mathbb{P}}_Q V_3$ holds too. For Properties (1) and (2) in Definition 3.3.17 this is immediate because

$$V_1(\mathsf{head}(Q)) = V_3(\mathsf{head}(Q)) = V_3(\mathsf{head}(Q))$$

and

$$V_1(\mathsf{pos}(Q)) \subseteq V_2(\mathsf{pos}(Q)) \subseteq V_3(\mathsf{pos}(Q)),$$

where the left-hand equality/inclusion follows from $V_1 \leq^{\mathbb{P}}_Q V_2$ and the right-hand equality/inclusion from $V_2 \leq^{\mathbb{P}}_Q V_3$. In the remainder of this proof, we show that $V_1$ and $V_3$ also satisfy Property (3) in Definition 3.3.17 by the following case distinction on the relationship between $V_1$ and $V_2$.

⋙ 1. Case ($V_1(\mathsf{neg}(Q)) \subseteq V_2(\mathsf{neg}(Q))$). If, additionally, $V_2(\mathsf{neg}(Q)) \subseteq V_3(\mathsf{neg}(Q))$ holds, then transitivity of inclusion implies $V_1(\mathsf{neg}(Q)) \subseteq V_3(\mathsf{neg}(Q))$ and Property (3a) is satisfied straightaway.

Otherwise, valuation $V_2$ and $V_3$ satisfy Property (3b). Thus, there exists a node $k_2 \in \mathbb{P}^{-1}(V_2(\mathsf{pos}(Q)))$ such that

$$V_2(\mathsf{neg}(Q)) \cap \mathbb{P}(k_2) \subseteq V_3(\mathsf{neg}(Q)) \tag{1}$$
$$V_2(\mathsf{neg}(Q)) \cap \mathbb{P}(k_2) \subseteq V_3(\mathsf{neg}(Q)) \cap \mathbb{P}(k_3) \tag{2}$$

for every $k_3 \in \mathbb{P}^{-1}(V_3(\mathsf{pos}(Q)))$. Since $V_1(\mathsf{pos}(Q)) \subseteq V_2(\mathsf{pos}(Q))$, node $k_2$ is also in the set $\mathbb{P}^{-1}(V_1(\mathsf{body}(Q)))$. Therefore, this node witnesses the satisfaction of Property (3b) by $V_1$ and $V_3$, as the next two inclusions demonstrate. First,

$$V_1(\mathsf{neg}(Q)) \cap \mathbb{P}(k_2) \subseteq V_2(\mathsf{neg}(Q)) \cap \mathbb{P}(k_2) \subseteq V_3(\mathsf{neg}(Q))$$

follows by the case assumption, $V_1(\mathsf{neg}(Q)) \subseteq V_2(\mathsf{neg}(Q))$, and Inclusion (1). Second, for every node $k_3 \in \mathbb{P}^{-1}(V_3(\mathsf{pos}(Q)))$, inclusion

$$V_1(\mathsf{neg}(Q)) \cap \mathbb{P}(k_2) \subseteq V_2(\mathsf{neg}(Q)) \cap \mathbb{P}(k_2) \subseteq V_3(\mathsf{neg}(Q)) \cap \mathbb{P}(k_3)$$

follows by the case assumption and Inclusion (2).

Therefore, valuation $V_1$ and $V_3$ satisfy Property (3).

⋙ 2. Case (otherwise). By assumption, valuations $V_1$ and $V_2$ satisfy Property (3) but not Property (3a). Hence, they satisfy Property (3b): there is a node $k_1 \in \mathbb{P}^{-1}(V_1(\mathsf{pos}(Q)))$ such that

$$V_1(\mathsf{neg}(Q)) \cap \mathbb{P}(k_1) \subseteq V_2(\mathsf{neg}(Q)) \text{ and} \tag{3}$$
$$V_1(\mathsf{neg}(Q)) \cap \mathbb{P}(k_1) \subseteq V_2(\mathsf{neg}(Q)) \cap \mathbb{P}(k_2) \tag{4}$$

for every node $k_2 \in \mathbb{P}^{-1}(V_2(\mathsf{pos}(Q)))$.

If $V_2(\mathsf{neg}(Q)) \subseteq V_3(\mathsf{neg}(Q))$ holds, then Inclusions (3) and (4) directly imply

$$V_1(\mathsf{neg}(Q)) \cap \mathbb{P}(k_1) \subseteq V_3(\mathsf{neg}(Q)) \text{ and}$$
$$V_1(\mathsf{neg}(Q)) \cap \mathbb{P}(k_1) \subseteq V_3(\mathsf{neg}(Q)) \cap \mathbb{P}(k_3)$$

for every node $k_3 \in \mathbb{P}^{-1}(V_3(\mathsf{pos}(Q)))$ because $\mathbb{P}^{-1}(V_3(\mathsf{pos}(Q))) \subseteq \mathbb{P}^{-1}(V_2(\mathsf{pos}(Q)))$, since valuation $V_3$ requires at least all facts required by valuation $V_2$. Hence, valuations $V_1$ and $V_3$ satisfy Property (3b).

Otherwise, there is a node $k_2 \in \mathbb{P}^{-1}(V_2(\mathsf{pos}(Q)))$ such that

$$V_2(\mathsf{neg}(Q)) \cap \mathbb{P}(k_2) \subseteq V_3(\mathsf{neg}(Q)) \text{ and}$$
$$V_2(\mathsf{neg}(Q)) \cap \mathbb{P}(k_2) \subseteq V_3(\mathsf{neg}(Q)) \cap \mathbb{P}(k_3)$$

for every node $k_3 \in \mathbb{P}^{-1}(V_3(\mathsf{pos}(Q)))$. The choice of nodes $k_1$ and $k_2$ implies then

$$V_1(\mathsf{neg}(Q)) \cap \mathbb{P}(k_1) \subseteq V_3(\mathsf{neg}(Q)) \text{ and}$$
$$V_1(\mathsf{neg}(Q)) \cap \mathbb{P}(k_1) \subseteq V_3(\mathsf{neg}(Q)) \cap \mathbb{P}(k_3)$$

in particular because the node on the right-hand side of Inclusion (4) is quantified universally. Hence, valuations $V_1$ and $V_3$ satisfy Property (3b).

This completes the case distinction and proves that valuations $V_1$ and $V_3$ satisfy also Property (3). □

**Proposition A.3.** $\text{PSound}(\mathsf{CQ}^{\neg}_{\text{dom}}, \mathcal{P}_{\text{list}})$, restricted to polarised queries, is $\Pi^p_2$-hard, even over a single-node network.

*Proof.* The following hardness proof describes a reduction from $\Pi_2$-QBF. This reduction is similar to the reduction for the $\Pi^p_2$-hardness of parallel completeness for $\text{CQ}_{\text{dom}}$ (Proposition 3.2.10). The correctness argument however is based on Condition (PSound-pol) now, provided by the characterisation in Proposition 3.3.15.

Every input formula $\varphi = \forall \boldsymbol{x} \exists \boldsymbol{y} \psi$ for $\Pi_2$-QBF with a propositional (quantifier-free) subformula $\psi$ in 3-CNF can be mapped to a query $Q \in \mathsf{CQ}^{\neg}_{\text{dom}}$ and a policy $\mathbb{P} \in \mathcal{P}_{\text{list}}$ in the following way. Let $\boldsymbol{x} = (x_1, \ldots, x_r)$ and $\boldsymbol{y} = (y_1, \ldots, y_s)$ and let $\psi$ consist of clauses $C_1, \ldots, C_p$, where $C_j = (\ell_{j,1} \vee \ell_{j,2} \vee \ell_{j,3})$ for every $j \in \{1, \ldots, p\}$. The database schema underlying query and policy contains a binary relation $\mathtt{Neg}$ for and a ternary relation $\mathtt{C}$ for the literals and the clauses of $\psi$, respectively. Data values $0, 1 \in \mathsf{dom}$ are used as truth values.

Set $\mathcal{F}_{\text{sat}} \stackrel{\text{def}}{=} \mathcal{F}_{\text{neg}} \cup \mathcal{F}_{\text{clau}}$ comprises two sets of facts,

$$\mathcal{F}_{\text{neg}} \stackrel{\text{def}}{=} \{\mathtt{Neg}(0,1), \mathtt{Neg}(1,0)\} \text{ and}$$
$$\mathcal{F}_{\text{clau}} \stackrel{\text{def}}{=} \{\mathtt{C}(\boldsymbol{w}) \mid j \in \{1, \ldots, p\} \text{ and } \boldsymbol{w} \in \mathbb{B}^+\}$$

which encode complementary truth values and satisfied clauses, respectively,—where the set $\mathbb{B}^+ \stackrel{\text{def}}{=} \{0,1\}^3 - \{(0,0,0)\}$ contains all triples of truth values with at least one true component.

Query $Q$ refers to variables $x_1, \ldots, x_r, y_1, \ldots, y_z$ and $\bar{x}_1, \ldots, \bar{x}_r, \bar{y}_1, \ldots, \bar{y}_s$ that represent the positive and negative literals, respectively, that may occur in $\psi$. The query's head $\mathsf{head}(Q) = H(x_1, \ldots, x_r)$ comprises all positive literals over the universally quantified variables while its body comprises positive atoms $\mathsf{pos}(Q) \stackrel{\text{def}}{=} \mathcal{F}_{\text{neg}} \cup \mathcal{A}_{\text{lit}}$ and negated atoms $\mathsf{neg}(Q) \stackrel{\text{def}}{=} \mathcal{F}_{\text{clau}} \cup \mathcal{A}_{\text{clau}}$ for

$$\mathcal{A}_{\text{lit}} \stackrel{\text{def}}{=} \{\mathtt{Neg}(x_i, \bar{x}_i) \mid i \in \{1, \ldots, r\}\} \cup \{\mathtt{Neg}(y_i, \bar{y}_i) \mid i \in \{1, \ldots, s\}\} \text{ and}$$
$$\mathcal{A}_{\text{clau}} \stackrel{\text{def}}{=} \{\mathtt{C}(\ell_{j,1}, \ell_{j,2}, \ell_{j,3}) \mid j \in \{1, \ldots, p\}\},$$

where—with slight abuse of notation—each literal $\ell_{j,h} = x_i$ is interpreted as variable $x_i$ and each literal $\ell_{j,h} = \neg x_i$ as variable $\bar{x}_i$ (and analogously for literals over propositions $y_1, \ldots, y_s$).

Single-node policy $\mathbb{P} \stackrel{\text{def}}{=} \{k_{\text{sat}}\} \times \mathcal{F}_{\text{sat}}$ marks node $k_{\text{sat}}$ responsible for all facts in $\mathcal{F}_{\text{sat}}$.

Query $Q$ is obviously polarised (with schema $\mathcal{S}^+ = \{\texttt{Neg}\}$ and $\mathcal{S}^- = \{\texttt{C}\}$) and can be computed from $\varphi$ in polynomial time, just like the representation of $\mathbb{P}$ as a list of pairs. Furthermore, the mapping is total. Hence, it remains to prove that it satisfies the reduction property: $\varphi \in \Pi_2$-QBF holds if and only if $(Q, \mathbb{P}) \in \textsc{PSound}(\texttt{CQ}^{\neg}_{\text{dom}}, \mathcal{P}_{\text{list}})$.

⋙ Only if. Assume that $\varphi \in \Pi_2$-QBF. We show that query $Q$ is parallel-sound under policy $\mathbb{P}$. To this end, let $\mathcal{G}$ be an arbitrary instance over $\mathbb{P}$. Since policy $\mathbb{P}$ is defined over a single node, it suffices to show that $Q((\mathbb{P} \triangleright \mathcal{G})(k_{\text{sat}})) \subseteq Q(\mathcal{G})$. To this end, let $V$ be an arbitrary valuation that satisfies $Q$ on node $k_{\text{sat}}$.

Obviously, valuation $V$ requires exactly facts $\mathcal{F}_{\text{neg}}$, that is, $V(\mathsf{pos}(Q)) = \mathcal{F}_{\text{neg}}$, because $\mathcal{F}_{\text{neg}} \subseteq \mathsf{pos}(Q)$ and node $k_{\text{sat}}$ is only responsible for these two facts over $\mathcal{S}^+$. In particular, $\mathcal{F}_{\text{neg}} \subseteq \mathcal{G}$ and $V(\mathcal{A}_{\text{lit}}) \subseteq \{\texttt{Neg}(0,1), \texttt{Neg}(1,0)\}$ holds. Furthermore, $\mathcal{F}_{\text{clau}} \cap \mathcal{G} = \emptyset$ because query $Q$ prohibits facts $\mathcal{F}_{\text{clau}} \subseteq \mathsf{neg}(Q)$, valuation $V$ is satisfying and node $k_{\text{sat}}$ is fully responsible for these facts.

Then, valuation $V$ induces a truth assignment in a natural fashion: let $\beta_{\boldsymbol{x}}$ be defined by $\beta_{\boldsymbol{x}}(x_i) = V(x_i)$ for every $i \in \{1, \ldots, r\}$. Since $\varphi \in \Pi_2$-QBF by assumption, there is a truth assignment $\beta_{\boldsymbol{y}}$ on $\boldsymbol{y}$ such that $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}} \models \psi$. From this combined truth assignment, a valuation $V'$ can be derived where $V'(x_i) = \beta_{\boldsymbol{x}}(x_i)$, for every $i \in \{1, \ldots, r\}$, and $V'(y_i) = \beta_{\boldsymbol{y}}(y_i)$, for every $i \in \{1, \ldots, s\}$, and variables $\bar{x}_1, \ldots, \bar{x}_r, \bar{y}_1, \ldots, \bar{y}_s$ are mapped to the respective complementary values. In particular, valuations $V$ and $V'$ agree on the head variables $x_1, \ldots, x_r$. Valuation $V'$ satisfies $Q$ on $\mathcal{G}$ because, as previously argued, $\mathcal{F}_{\text{neg}} \subseteq \mathcal{G}$ and $\mathcal{F}_{\text{clau}} \cap \mathcal{G} = \emptyset$ and furthermore, $V'(\mathsf{pos}(Q)) \subseteq \mathcal{F}_{\text{neg}}$ by definition of $V'$ and $V'(\mathsf{neg}(Q)) \subseteq \mathcal{F}_{\text{clau}}$ as argued next. The values of $\ell_{j,1}, \ell_{j,2}, \ell_{j,3}$ under $V'$ faithfully represent the truth values of the literals in the $j$-th clause of formula $\psi$ under assignment $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$. This assignment is satisfying by assumption and therefore does not map all three literals of any clause to 0. Hence, every atom in $\mathcal{A}_{\text{clau}}$ is mapped to some fact in $\mathcal{F}_{\text{clau}}$.

Therefore, valuation $V'$ witnesses that the fact locally derived by $V$ is also derived globally. This implies that query $Q$ is parallel-sound under policy $\mathbb{P}$.

⋙ If. Towards a proof by contraposition, assume that $\varphi \notin \Pi_2$-QBF. The following argument shows that query $Q$ is then *not* parallel-sound under $\mathbb{P}$, which is witnessed by instance $\mathcal{G} \stackrel{\text{def}}{=} \{\texttt{Neg}(0,1), \texttt{Neg}(1,0), \texttt{C}(0,0,0)\}$.

By assumption, there exists a truth assignment $\beta_{\boldsymbol{x}}$ on $\boldsymbol{x}$ such that there is no truth assignment $\beta_{\boldsymbol{y}}$ on $\boldsymbol{y}$ such that $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$ satisfies $\psi$. Let $V$ be the valuation for $Q$ induced by $\beta_{\boldsymbol{x}}$ on the universally quantified variables together with $V(y_i) = 0$ and $V(\bar{y}_i) = 1$ for every $i \in \{1, \ldots, s\}$. Obviously, valuation $V$ satisfies $Q$ on the local

instance $(\mathbb{P} \triangleright \mathcal{G})(k_{\text{sat}})$ of node $k_{\text{sat}}$ because the required facts $V(\mathsf{pos}(Q)) = \mathcal{F}_{\text{neg}}$ are contained in both $\mathcal{G}$ and $\mathbb{P}(k_{\text{sat}})$ while none of the prohibited facts occurs in the local instance—node $k_{\text{sat}}$ is not responsible for $\mathtt{C}(0,0,0)$, the only $\mathtt{C}$-fact in instance $\mathcal{G}$.

However, fact $V(\mathsf{head}(Q))$ is not derived globally. For the sake of contradiction, assume that a globally satisfying valuation $V'$ exists that agrees with $V$ on the head variables. Because $V'(\mathcal{A}_{\text{lit}}) \subseteq \mathcal{F}_{\text{neg}}$, valuation $V'$ unambiguously induces truth assignments $\gamma_{\boldsymbol{x}}$ on $\boldsymbol{x}$ and $\gamma_{\boldsymbol{y}}$ on $\boldsymbol{y}$. From $V'(\mathsf{head}(Q)) = V(\mathsf{head}(Q))$ we can infer that assignment $\gamma_{\boldsymbol{x}}$ is identical to $\beta_{\boldsymbol{x}}$. Moreover, valuation $V'$ maps every atom in $\mathcal{A}_{\text{lit}}$ to a fact in $\mathcal{F}_{\text{clau}}$ because $V'(\mathsf{neg}(Q)) \cap \mathcal{G} = \emptyset$ implies $V'(\mathsf{neg}(Q)) \cap \{\mathtt{C}(0,0,0)\} = \emptyset$. But then the existence of truth assignment $\gamma_{\boldsymbol{y}}$ contradicts the choice of $\beta_{\boldsymbol{x}}$. Hence, there is no such valuation $V'$.

Therefore, valuation $V$ witnesses the local derivation of a fact that is not derived globally. Thus, query $Q$ is *not* parallel-sound under policy $\mathbb{P}$.

This proves the mapping to be a polynomial reduction from $\Pi_2$-QBF. □

**Lemma A.4 (Constant elimination and parallel soundness).** The canonical extension of the reduction in the proof of Lemma 3.2.12 to queries from $\mathsf{UCQ}^{\neg,\neq}_{\text{dom}}$ also preserves parallel soundness.

*Proof.* Let $Q$ be a query from $\mathsf{UCQ}^{\neg,\neq}_{\text{dom}}$ with constants $c_1, \ldots, c_s$ and let $\mathbb{P}$ be a policy. The canonical extension of the reduction yields a constant-free query $Q'$, where the substitution $\sigma = [c_1/y_1, \ldots, c_s/y_s]$ is applied to the head, the positive atoms and the disequality atoms as in the original proof and, additionally, to the negated atoms, $\mathsf{neg}(Q') \stackrel{\text{def}}{=} \sigma(\mathsf{neg}(Q))$. Policy $\mathbb{P}'$ is the same as before.

We claim that query $Q$ is parallel-sound under policy $\mathbb{P}$ if and only if query $Q'$ is parallel-sound under policy $\mathbb{P}'$ and prove both implications separately.

⋙ Only if. Assume that $Q$ is parallel-sound under $\mathbb{P}$. We show that then $Q'$ is parallel-sound under $\mathbb{P}'$ too. To this end, let $\mathcal{G}'$ be an arbitrary global instance over $\mathbb{P}'$ and let $V'$ be valuation that satisfies $Q'$ on the local instance $(\mathbb{P}' \triangleright \mathcal{G}')(k)$ of some node $k \in \mathsf{net}(\mathbb{P}')$. Node $k$ is either a node from the original policy, $k \in \mathsf{net}(\mathbb{P})$, or node newly introduced by $\mathbb{P}'$, that is, $k \in \{\ell_1, \ldots, \ell_s\}$.

In the latter case, the construction of $\mathbb{P}'$ guarantees that node $k$ is responsible for all facts prohibited by $V'$. Thus, $V'(\mathsf{neg}(Q')) \cap (\mathbb{P}' \triangleright \mathcal{G}')(k) = \emptyset$ implies $V'(\mathsf{neg}(Q')) \cap \mathcal{G}' = \emptyset$. Clearly, the facts required by $V'$ are part of the local instance only when they also occur in the global instance. Hence, valuation $V'$ satisfies query $Q'$ also on $\mathcal{G}'$.

In the former case, valuation $V'$ requires facts $\mathtt{Const}_1(c_1), \ldots, \mathtt{Const}_s(c_s)$, that is, it maps variables $y_1, \ldots, y_s$ to data values $c_1, \ldots, c_s$. In particular, mapping $V'$ is then a valuation for the original query $Q$, which requires (here and in the following: modulo $\mathtt{Const}_i$-facts) and prohibits the same facts as $V'$ for $Q'$, respectively. Since query $Q$ is

parallel-sound under policy $\mathbb{P}$, there is a valuation $W$ for $Q$ that derives the same fact as $V'$ for $Q$ and is satisfying on the global instance $\mathcal{G}$ that results from $\mathcal{G}'$ by removal of all $\texttt{Const}_i$-facts. The induced valuation $W' \stackrel{\text{def}}{=} W[y_1 \mapsto c_1, \ldots, y_s \mapsto c_s]$ for $Q'$, in turn, requires, prohibits and derives the same facts for $Q'$ as $W$ for $Q$. In summary, valuation $W'$ satisfies query $Q'$ globally (because $\mathcal{G}'$ contains the $\texttt{Const}_i$-facts, also required by $V'$) and derives the same fact as the locally satisfying valuation $V'$.

In both cases, the locally satisfying valuation $V'$ for $Q'$ has a globally satisfying counterpart. Therefore, query $Q'$ is parallel-sound under policy $\mathbb{P}'$.

⋙ If. Now, assume that query $Q'$ is parallel-sound under policy $\mathbb{P}'$. We show that this implies parallel soundness of query $Q$ under policy $\mathbb{P}$. Let $\mathcal{G}$ be an arbitrary global instance over $\mathbb{P}$. Furthermore, let $V$ be a valuation that satisfies $Q$ on the local instance $(\mathbb{P} \triangleright \mathcal{G})(k)$ of some node $k \in \mathsf{net}(\mathbb{P})$.

Then, the mapping $V' \stackrel{\text{def}}{=} V[y_1 \mapsto c_1, \ldots, y_s \mapsto c_s]$ is a valuation for query $Q'$. We consider a slightly extended global instance, $\mathcal{G}' \stackrel{\text{def}}{=} \mathcal{G} \cup \{\texttt{Const}_1(c_1), \ldots, \texttt{Const}_s(c_s)\}$. Clearly, valuation $V'$ satisfies query $Q'$ on the corresponding local instance $(\mathbb{P}' \triangleright \mathcal{G}')(k)$ of the same node $k$. Furthermore, since $Q'$ is parallel-sound under $\mathbb{P}'$, there is a valuation $W'$ for $Q'$ that derives the same fact on $\mathcal{G}'$. Moreover, valuation $W'$ maps variables $y_1, \ldots, y_s$, just like valuation $V'$, to data values $c_1, \ldots, c_s$. Therefore, it is also a valuation for the original query $Q$, deriving, requiring and prohibiting the same facts. In particular, it satisfies query $Q$ on the global instance $\mathcal{G}$ and derives the same fact as valuation $V$ for $Q$. Hence, query $Q$ is parallel-sound under policy $\mathbb{P}$.

As claimed, query $Q$ is parallel-sound under policy $\mathbb{P}$ if and only if query $Q'$ is parallel-sound under policy $\mathbb{P}'$. □

## A.2. Parallel-correctness transfer

The following is restatement of Proposition 4.2.14. The proof below is very simple and straightforward adaptation of the proof for (standard) transfer [AGK+15].

**Proposition A.5 (Characterisation of mild transfer).** For queries $Q_1, Q_2 \in \mathsf{UCQ}^{\neq}$, parallel completeness transfers mildly from $Q_1$ to $Q_2$ if and only if they satisfy Condition (mPComp-T).

*Proof.* We prove both implications separately.

⋙ If. Assume that Condition (mPComp-T) is satisfied by queries $Q_1$ and $Q_2$. We show that parallel completeness transfers mildly from $Q_1$ to $Q_2$. To this end, let $\mathbb{P}$ be a policy such that, first, query $Q_1$ is parallel complete under $\mathbb{P}$ and, second, the policy's universe contains at least as many data values as any disjunct of $Q_1$ contains variables.

Let $V_2$ be a minimal valuation for $Q_2$ over $\mathbb{P}$. By assumption, there exists a minimal valuation $V_1$ for $Q_1$ that is a fact-cover of $V_2$, that is, with $V_2(\mathsf{pos}(Q_2)) \subseteq V_1(\mathsf{pos}(Q_1))$. Since query $Q_1$ is generic, we can assume that $V_1$ refers to values from $\mathsf{univ}(\mathbb{P})$ only. In other words, $V_1$ is a valuation *over* $\mathbb{P}$ (this argument is different in the standard version). Then, there exists a node $k \in \mathsf{net}(\mathbb{P})$ that is responsible for all facts $V_1(\mathsf{pos}(Q_1))$ because query $Q_1$ is parallel-complete under $\mathbb{P}$. The containment relationship above than implies that node $k$ is also responsible for all facts required by $V_2$. Hence, query $Q_2$ is parallel-complete under policy $\mathbb{P}$ by Condition (PComp).

⋙ Only if. For a proof by contraposition, assume now that Condition (mPComp-T) is violated by queries $Q_1$ and $Q_2$. We show that parallel completeness does *not* transfer mildly from $Q_1$ to $Q_2$.

Since $Q_1$ and $Q_2$ violate Condition (mPComp-T), there is a minimal valuation $V_2$ for $Q_2$ such that there exists no minimal valuation $V_1$ for $Q_1$ that is a fact-cover of $V_2$. Let $U$ be a set of at least $s$ data values, where $s$ is the maximum number of variables in any disjunct from $Q_1$, that comprises $\mathsf{dom}(V_2(\mathsf{pos}(Q_2)))$. Furthermore, let $\mathcal{S}$ denote the schema underlying query $Q_2$. We define policy $\mathbb{P}$ based on the facts $V_2(\mathsf{pos}(Q_2)) = \{f_1, \ldots, f_n\}$.

$$\mathbb{P} \stackrel{\text{def}}{=} \bigcup_{i=1}^{n} \{k_i\} \times (\mathsf{facts}(\mathcal{S}, U) - \{f_i\})$$

Clearly, query $Q_2$ is *not* parallel-complete under policy $\mathbb{P}$ by Condition (PComp) because no node $k_i$ is responsible for all facts required by the minimal valuation $V_2$. Query $Q_1$, however, is parallel-complete under $\mathbb{P}$ by Condition (PComp), as the following argument shows. Let $V_1$ be an arbitrary minimal valuation for $Q_1$ over $\mathbb{P}$. Then, by assumption, it does *not* require all facts $f_1, \ldots, f_n$. Say it does not require $f_i$, then node $k_i$ is responsible for all facts $V_1(\mathsf{pos}(Q_1))$.

Therefore, policy $\mathbb{P}$ witnesses that parallel completeness does *not* transfer mildly from $Q_1$ to $Q_2$.

Hence, parallel completeness transfers mildly from $Q_1$ to $Q_2$ if and only if Condition (mPComp-T) is satisfied. □

**Proposition A.6.** Decision problem PSound-T($\mathsf{CQ}^{\neg}_{[\mathsf{pol}]}$) is $\Pi_3^p$-hard.

The following proof is an adaptation of the proof of the $\Pi_3^p$-hardness of PComp-T($\mathsf{CQ}$) stated in Proposition 4.2.7.

*Proof.* We define a mapping from inputs for $\Pi_3$-QBF to inputs of PSound-T($\mathsf{CQ}^{\neg}_{[\mathsf{pol}]}$) and show that it is a polynomial reduction.

Every input formula $\varphi = \forall \boldsymbol{x} \exists \boldsymbol{y} \forall \boldsymbol{z}\ \psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ for $\Pi_3$-QBF with a quantifier-free subformula $\psi$ in 3-DNF is mapped to a pair $(Q_1, Q_2)$ of queries in the following way. Let $\boldsymbol{x} = (x_1, \ldots, x_r)$, $\boldsymbol{y} = (y_1, \ldots, y_s)$ and $\boldsymbol{z} = (z_1, \ldots, z_t)$ be the propositions

of $\varphi$ and let $\psi$ consist of clauses $C_1, \ldots, C_p$ where $C_j = (\ell_{j,1} \wedge \ell_{j,2} \wedge \ell_{j,3})$ for every $j \in \{1, \ldots, p\}$. Without loss of generality, we assume that each proposition occurs at least in one clause.

The schema for the queries contains unary relations $\texttt{Res}, \texttt{False}, \texttt{True}$ for 'result' and truth values and also unary relations $\texttt{XVal}_h$, for every $h \in \{1, \ldots, r\}$, for mappings of the propositions. Additionally, there is a binary relation $\texttt{Neg}$, a ternary relation $\texttt{Or}$ and a quaternary relation $\texttt{And}$ to guide the 'evaluation' of the represented subformula $\psi$.

Query $Q_2$ refers to 'truth variables' $w_0, w_1$, intended to represent truth values *false* and *true*, and it refers to variables $x_1, \ldots, x_r$ for the respective propositions. The query is defined by

$$\begin{aligned}
\mathsf{head}(Q_2) &\stackrel{\text{def}}{=} H(w_0, w_1, x_1, \ldots, x_r),\\
\mathsf{pos}(Q_2) &\stackrel{\text{def}}{=} \mathcal{A}_{\text{fix}} \text{ and}\\
\mathsf{neg}(Q_2) &\stackrel{\text{def}}{=} \mathcal{A}_{\text{sat}} \cup \mathcal{A}_{\boldsymbol{y}}^{0,1} \cup \{\texttt{Res}(w_0), \texttt{Res}(w_1)\},
\end{aligned}$$

where the set of atoms

$$\mathcal{A}_{\text{fix}} \stackrel{\text{def}}{=} \{\texttt{False}(w_0), \texttt{True}(w_1)\} \cup \{\texttt{XVal}_h(x_h) \mid h \in \{1, \ldots, r\}\}$$

is intended to fix the values of the query's variables. Furthermore, set

$$\begin{aligned}
\mathcal{A}_{\text{sat}} \stackrel{\text{def}}{=}\ & \{\texttt{Neg}(w_0, w_1), \texttt{Neg}(w_1, w_0)\}\\
& \cup \{\texttt{And}(\boldsymbol{w}, w_0) \mid \boldsymbol{w} \in \mathbb{W}^-\} \cup \{\texttt{And}(w_1, w_1, w_1, w_1)\}\\
& \cup \{\texttt{Or}(w_0, w_0, w_0), \texttt{Or}(w_0, w_1, w_1), \texttt{Or}(w_1, w_0, w_1), \texttt{Or}(w_1, w_1, w_1)\}
\end{aligned}$$

represents the logical functions negation, conjunction and disjunction if the last position is considered as the output for the previous positions. To this end, the set $\mathbb{W}^- \stackrel{\text{def}}{=} \{w_0, w_1\}^3 - \{(w_1, w_1, w_1)\}$ contains all triples of truth variables with at least one false component. Set

$$\mathcal{A}_{\boldsymbol{y}}^{0,1} \stackrel{\text{def}}{=} \{\texttt{YVal}_1(w_0), \texttt{YVal}_1(w_1), \ldots, \texttt{YVal}_s(w_0), \texttt{YVal}_s(w_1)\}$$

is intended to demand—from guarding valuations for $Q_1$—each pair of literal variables $(y_i, \bar{y}_i)$ for a proposition $y_i$ to be mapped to (complementary) truth values. Note also that query $Q_2$ is strongly minimal because it is full.

Query $Q_1$ refers to the same variables as $Q_2$ and some additional variables. First, it refers to 'literal variables' $\pi, \bar{\pi}$ for every $\pi \in \{x_1, \ldots, x_r, y_1, \ldots, y_s, z_1, \ldots, z_t\}$. Second, it refers to variables $\alpha_1, \ldots, \alpha_p$ and $\omega_1, \ldots, \omega_p$, with the intention that variable $\alpha_j$ represents the truth value of clause $C_j$ and variable $\omega_j$ represents the truth value of the partial disjunction $C_1 \vee \ldots \vee C_j$ for every $j \in \{1, \ldots, p\}$. The query is defined by

$$\mathsf{head}(Q_1) \stackrel{\text{def}}{=} H(w_0, w_1, x_1, \ldots, x_r, y_1, \ldots, y_s, \bar{y}_1, \ldots, \bar{y}_s),$$
$$\mathsf{pos}(Q_1) \stackrel{\text{def}}{=} \mathcal{A}_{\text{fix}} \text{ and}$$
$$\mathsf{neg}(Q_1) \stackrel{\text{def}}{=} \mathcal{A}_{\text{sat}} \cup \mathcal{A}_\psi \cup \mathcal{A}_{\boldsymbol{y}} \cup \{\mathtt{Res}(w_0), \mathtt{Res}(\omega_p)\}.$$

Note the difference between variables $w_1$ and $\omega_p$ in the second `Res`-fact of the queries. Set

$$\begin{aligned}\mathcal{A}_\psi \stackrel{\text{def}}{=} &\{\mathtt{Neg}(\pi, \bar{\pi}) \mid \text{for each proposition } \pi \text{ in } \psi\} \\ &\cup \{\mathtt{And}(\ell_{j,1}, \ell_{j,2}, \ell_{j,3}, \alpha_j) \mid j \in \{1, \ldots, p\}\} \\ &\cup \{\mathtt{Or}(\alpha_1, \alpha_1, \omega_1), \mathtt{Or}(\omega_1, \alpha_2, \omega_2), \ldots, \mathtt{Or}(\omega_{p-1}, \alpha_p, \omega_p)\}\end{aligned}$$

is intended to represent the literals of subformula $\psi$ as well as its (conjunctive) clauses and the results of their partial disjunctions. Finally, set

$$\mathcal{A}_{\boldsymbol{y}} \stackrel{\text{def}}{=} \{\mathtt{YVal}_1(y_1), \mathtt{YVal}_1(\bar{y}_1), \ldots, \mathtt{YVal}_s(y_s), \mathtt{YVal}_s(\bar{y}_s)\}$$

is intended to force guarding valuations to be (unambiguously) interpretable as truth assignments over propositions $y_1, \ldots, y_s$.

The pair $(Q_1, Q_2)$ of queries can obviously be computed from formula $\varphi$ in polynomial time and the mapping defined in this way is total. It remains to prove that the mapping satisfies the reduction property.

As in the original proof (for parallel-completeness transfer), we are interested in valuations with range $\{0, 1\}$ and particularly in such that map $(w_0, w_1)$ to $(0, 1)$ and $(\pi, \bar{\pi})$ to either $(0, 1)$ or $(1, 0)$ for each proposition $\pi$. We call these valuations *boolean*. To facilitate the following arguments, we call a boolean valuation $V_1$ for query $Q_1$ *compatible* if it respects the intended meaning of the query's further variables: variables $\alpha_1, \ldots, \alpha_p$ and $\omega_1, \ldots, \omega_p$ are mapped to the truth values of the represented clauses and partial disjunctions, which is unambiguously possible for boolean valuations. Thus, a boolean valuation $V_1$ for $Q_1$ is compatible if and only if $V_1(\mathcal{A}_\psi) \subseteq V_1(\mathcal{A}_{\text{sat}})$ holds.

We will repeatedly define a boolean valuation $V$ from a truth assignment $\beta$ and vice versa. It is in this sense that we say that the mappings are *induced*. This means that $V(w_0, w_1) = (0, 1)$ holds and also $V(\pi) = \beta(\pi)$ for every proposition $\pi$ in the context—implicitly defined by the mapping or stated explicitly. If valuation $V$ is for query $Q_1$, it furthermore maps $\bar{\pi}$ to the complement of $V(\pi)$ for every $\bar{\pi}$ under consideration.

We show now that $\varphi \in \Pi_3$-QBF holds if and only if $(Q_1, Q_2) \in \text{PSound-T}(\mathsf{CQ}^\neg_{[\mathsf{pol}]})$. As usual, we prove both implications separately.

⋙ If. For a proof by contraposition, we assume that $\varphi \notin \Pi_3$-QBF and show that Condition (PSound-T-pol) is violated for the pair $(Q_1, Q_2)$ of queries for $\varphi$.

By our assumption $\varphi \notin \Pi_3$-QBF, there exists a truth assignment $\beta_{\boldsymbol{x}}$ for $\boldsymbol{x}$ such that, for every truth assignment $\beta_{\boldsymbol{y}}$ for $\boldsymbol{y}$, there is a truth assignment $\beta_{\boldsymbol{z}}$ for $\boldsymbol{z}$ such

that the combined assignment $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}} \cup \beta_{\boldsymbol{z}}$ does *not* satisfy subformula $\psi$ of $\varphi$. Using this assignment $\beta_{\boldsymbol{x}}$, we define valuation $V_2$ for $Q_2$ by

$$V_2 \stackrel{\text{def}}{=} \{w_0 \mapsto 0, w_1 \mapsto 1, x_1 \mapsto \beta_{\boldsymbol{x}}(x_1), \ldots, x_r \mapsto \beta_{\boldsymbol{x}}(x_r)\},$$

which is minimal since $Q_2$ is strongly minimal.

There is, however, no valuation $V_1$ for $Q_1$ that is both minimal and guarding for $V_2$. Note also that a *single* guard would be sufficient because each valuation for $Q_1$ that satisfies the requirement condition maps variables $w_0, w_1$ and $x_1, \ldots, x_r$ just like $V_2$. But then, $V_1$ prohibits every fact prohibited by $V_2$, with the possible exception of the single fact $V_2(\texttt{Res}(w_1))$, as a result of the following Properties (P1) and (P3) below.

Towards a proof by contradiction, assume that a valuation $V_1$ exists that is minimal and a guard for $V_2$. Guarding then particularly implies $V_1(\mathsf{pos}(Q_1)) \subseteq V_2(\mathsf{pos}(Q_2))$ and $V_2(\mathsf{neg}(Q_2)) \subseteq V_1(\mathsf{neg}(Q_1))$, and hence the following three properties.

(P1) Valuation $V_1$ agrees with $V_2$ on variables $w_0, w_1$ and $x_1, \ldots, x_r$.

(P2) Valuation $V_1$ maps $\omega_p$ to 1.

(P3) Valuation $V_1$ maps $(y_i, \bar{y}_i)$ either to $(0, 1)$ or to $(1, 0)$ for every $i \in \{1, \ldots, s\}$.

Property (P1) is implied because of the atoms in $\mathcal{A}_{\text{fix}}$, which are the only atoms that refer to relations $\texttt{True}$, $\texttt{False}$ and $\texttt{XVal}_h$, for $h \in \{1, \ldots, r\}$. Property (P2) is implied because the prohibited fact $V_2(\texttt{Res}(w_1)) = \texttt{Res}(1)$ can only be prohibited via $V_1(\texttt{Res}(\omega_p))$, not via the only other $\texttt{Res}$-fact $V_1(\texttt{Res}(w_0)) = \texttt{Res}(0)$. Finally, Property (P3) is implied because, for each $i \in \{1, \ldots, s\}$, facts $\texttt{YVal}_i(0), \texttt{YVal}_i(1)$ are prohibited by $V_2$ due to $\mathcal{A}^{0,1}_{\boldsymbol{y}}$. They can only be prohibited by $V_1$ if variables $y_i, \bar{y}_i$ are mapped to complementary values from $\{0, 1\}$.

Since Property (P3) holds, valuation $V_1$ unambiguously induces a truth assignment $\beta_{\boldsymbol{y}}$ on $\boldsymbol{y}$, where $\beta_{\boldsymbol{y}}(y_1, \ldots, y_s) = V_1(y_1, \ldots, y_s)$. From our starting assumption, it follows that there is an assignment $\beta_{\boldsymbol{z}}$ such that $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}} \cup \beta_{\boldsymbol{z}}$ does *not* satisfy $\psi$. Let $U_1$ be the valuation induced by $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}} \cup \beta_{\boldsymbol{z}}$. We claim that $U_1 <_{Q_1} V_1$, which contradicts the assumed minimality of $V_1$. First, both valuations obviously agree on the head variables. Second, they require the same facts for $\mathcal{A}_{\text{fix}}$. Since both valuations are compatible, they prohibit the same facts on $\mathcal{A}_{\text{sat}}$. For the same reason, valuation $V_1$ prohibits no additional facts for $\mathcal{A}_\psi$. However, $U_1(\omega_p) = 0$ by the choice of $\beta_{\boldsymbol{z}}$, while $V_1(\omega_p) = 1$, and thus $U_1$ prohibits one $\texttt{Res}$-fact less than $V_1$. Valuation $V_1$ is hence not minimal, the desired contradiction. Therefore, valuation $V_2$ has no minimal guard for $Q_1$.

⋙ Only if. Assume that $\varphi \in \Pi_3$-QBF. We show that every (minimal) valuation $V_2$ for $Q_2$ has a guarding minimal valuation $V_1$ for $Q_1$. Then, by Condition (PSound-T-pol), parallel soundness transfers from $Q_1$ to $Q_2$.

Let $V_2$ be an arbitrary valuation for $Q_2$, which maps the truth variables to some data values $V_2(w_0) = c_0$ and $V_2(w_1) = c_1$. Without loss of generality, we may assume

that $c_0 = 0$ and $c_1 \in \{0, 1\}$ because the guarding conditions are not affected by the application of a bijection over $\mathsf{dom}$ on both valuations. We distinguish three cases, depending on how $V_2$ maps the head variables $w_0, w_1, x_1, \ldots, x_r$.

» 1. Case ($V_2$ is boolean). Valuation $V_2$ induces a truth assignment $\beta_{\boldsymbol{x}}$ on $\boldsymbol{x}$. Since $\varphi \in \Pi_3$-QBF, there is an assignment $\beta_{\boldsymbol{y}}$ on $\boldsymbol{y}$ such that, for every assignment $\beta_{\boldsymbol{z}}$ on $\boldsymbol{z}$, it holds $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}} \cup \beta_{\boldsymbol{z}} \models \psi$. In particular, this holds for the assignment $\beta_{\boldsymbol{z}}$ that maps every proposition $z_1, \ldots, z_t$ to 0. Let $V_1$ be the valuation induced by these assignments, $\beta_{\boldsymbol{x}}$, $\beta_{\boldsymbol{y}}$ and $\beta_{\boldsymbol{z}}$. We claim that this valuation guards $V_2$ and that it is minimal.

First, we argue that $V_1$ guards $V_2$. Clearly, both valuations agree on the head variables $w_0, w_1, x_1, \ldots, x_r$ of $Q_2$ by definition and thus require the same facts, namely $V_1(\mathsf{pos}(Q_1)) = V_1(\mathcal{A}_{\text{fix}}) = V_2(\mathcal{A}_{\text{fix}}) = V_2(\mathsf{pos}(Q_2))$. Furthermore, $V_2(\mathcal{A}^{0,1}_{\boldsymbol{y}}) = V_1(\mathcal{A}_{\boldsymbol{y}})$ because $V_1$ is boolean. Compatibility of $V_1$ and the choice of $\beta_{\boldsymbol{y}}$ further ensure that $V_1(\omega_p) = 1$. Therefore, the only additional fact $V_2(\mathtt{Res}(w_1)) = \mathtt{Res}(1)$ prohibited by $V_2$ is also prohibited by $V_1$. Hence, $V_2(\mathsf{neg}(Q_2)) \subseteq V_1(\mathsf{neg}(Q_1))$.

Now, we argue that valuation $V_1$ is minimal. Towards a contradiction, assume that there exists a valuation $U_1$ such that $U_1 <_{Q_1} V_1$. Then, $U_1$ requires the same facts, $U_1(\mathsf{pos}(Q_1)) = V_1(\mathsf{pos}(Q_1))$, because both valuations agree on the head variables and thus $U_1$ requires strictly fewer facts, $U_1(\mathsf{neg}(Q_1)) \subsetneq V_1(\mathsf{neg}(Q_1))$. In particular, this implies $U_1(\mathsf{neg}(Q_1)) \subseteq U_1(\mathcal{A}_{\text{sat}}) = V_1(\mathcal{A}_{\text{sat}} \cup \mathcal{A}_\psi)$. Hence, valuation $U_1$ encodes a valid truth assignment on all propositions. This truth assignment agrees particularly with $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}}$ on $x_1, \ldots, x_r$ and $y_1, \ldots, y_s$ because valuations $U_1$ and $V_1$ map the head variables of $Q_1$ identically. However, strict containment implies $U(\{\mathtt{Res}(w_0), \mathtt{Res}(\omega_p)\}) = \{\mathtt{Res}(0)\} \subsetneq \{\mathtt{Res}(0), \mathtt{Res}(1)\} = V_1(\{\mathtt{Res}(w_0), \mathtt{Res}(\omega_p)\})$ and thus $U_1(\omega_p) = 0$. Therefore, valuation $U_1$ represents a different truth assignment $\beta'_{\boldsymbol{z}}$ for propositions $z_1, \ldots, z_t$. In particular one such that $\beta_{\boldsymbol{x}} \cup \beta_{\boldsymbol{y}} \cup \beta'_{\boldsymbol{z}}$ does *not* satisfy $\psi$. This contradicts the choice of $\beta_{\boldsymbol{y}}$ and thus proves $V_1$ minimal.

» 2. Case ($V_2(w_1) = 0$ and $V_2(\{x_1, \ldots, x_r\}) \subseteq \{0, 1\}$). Let $W_1$ be the valuation for $Q_1$ that agrees with $V_2$ on variables $w_0, w_1$ and $x_1, \ldots, x_r$ such that it maps all other variables to 0. Let $V_1$ be a minimal valuation with $V_1 \leq_{Q_1} W_1$. Then, $V_1$ also agrees with $V_2$ on the head variables of $Q_2$, implying that they require the same facts, $V_1(\mathcal{A}_{\text{fix}}) = V_2(\mathcal{A}_{\text{fix}})$. By the case assumption, valuation $V_2$ prohibits only facts $\{\mathtt{YVal}_1(0), \ldots, \mathtt{YVal}_s(0)\}$ for $\mathcal{A}^{0,1}_{\boldsymbol{y}}$, which are also prohibited by $V_1$ for $\mathcal{A}_{\boldsymbol{y}}$ because $W_1$ prohibits exactly these $\mathtt{YVal}_i$-facts and valuation $V_1$ prohibits no additional facts. Finally, fact $V_2(\mathtt{Res}(w_1)) = \mathtt{Res}(0)$ is also prohibited by $V_1$ since $\mathtt{Res}(w_0) \in \mathsf{neg}(Q_1)$.

Therefore, valuation $V_1$ is minimal and guarding for $V_2$.

» 3. Case ($V_2(\{x_1, \ldots, x_r\}) \not\subseteq \{0, 1\}$). To show that there is a guarding valuation

for $V_2$, we have to deal carefully with the data values from $V_2$ that are not interpreted as false or true. The following argument works independently of the mapping $V_2(w_1) \in \{c_0, c_1\} \subseteq \{0, 1\}$.

By the case assumption, there is at least one variable $x_h$ that is mapped by $V_2$ to a data value different from 0 and 1. We call each variable $x_h$ with this property *foul*. Similarly, we call a clause *foul* if it contains $x_h$ or $\neg x_h$ for a foul variable $x_h$. Furthermore, we fix one of the data values that a foul variable is mapped to, say $d \stackrel{\text{def}}{=} V_2(x_h)$ for the minimal index $h$ of a foul variable.

As an intermediate step, we define a valuation $W_1$ for $Q_1$, which is not necessarily minimal, in two steps. First, valuation $W_1$ satisfies the following equations on the truth and literal variables

$$W_1(w_0, w_1, x_1, \ldots, x_r) = V_2(w_0, w_1, x_1, \ldots, x_r) \tag{5}$$
$$W_1(y_1, \ldots, y_s, z_1, \ldots, z_t) = (c_0, \ldots, c_0) \tag{6}$$
$$W_1(\bar{y}_1, \ldots, \bar{y}_s, \bar{z}_1, \ldots, \bar{z}_t) = (c_1, \ldots, c_1) \tag{7}$$
$$W_1(\bar{x}_i) = \overline{V_2(x_i)} \text{ for every non-foul variable } x_i \tag{8}$$
$$W_1(\bar{x}_i) = d \text{ for every foul variable } x_i, \tag{9}$$

where $\overline{c_0} = c_1$ and $\overline{c_1} = c_0$. Second, valuation $W_1$ is successively defined on the remaining variables. For each $j$ from 1 to $p$,

- variable $\alpha_j$ is mapped to
  - 1 if clause $C_j$ is foul or all its literals are mapped to 1, and to
  - 0 otherwise;
- variable $\omega_j$ is mapped to
  - 1 if $W_1(\alpha_i) = 1$ for some $i \in \{1, \ldots, j\}$, and to
  - 0 otherwise.

Although valuation $W_1$ guards valuation $V_2$, it is not necessarily minimal. In the remainder of this proof, we show that each minimal valuation 'below' $W_1$ is also a guard of $V_2$. To this end, let $V_1$ be a minimal valuation such that $V_1 \leq_{Q_1} W_1$ holds. We claim that $V_1$ guards $V_2$.

First, we consider the requirement restriction. Because valuations $V_1$ and $W_1$ agree on the head variables of $Q_1$ and $W_1$ maps variables $w_0, w_1$ and $x_1, \ldots, x_r$ like valuation $V_2$, valuations $V_1$ and $V_2$ require the same facts for the only positive atoms in $\mathcal{A}_{\text{fix}}$.

Next, we consider the prohibition restriction. Clearly, valuation $V_1$ prohibits the facts $V_2(\mathcal{A}_{\text{sat}})$ prohibited by $V_2$ because of the previously mentioned agreement on variables $w_0, w_1$. Similarly, because valuations $V_1$ and $W_1$ also agree on $y_1, \ldots, y_s, \bar{y}_1, \ldots, \bar{y}_s$, which are head variables of $Q_1$, valuation $V_1$ also prohibits the facts: $V_2(\mathcal{A}_{\boldsymbol{y}}^{0,1}) \subseteq \{\texttt{YVal}_1(c_0), \texttt{YVal}_1(c_1), \ldots, \texttt{YVal}_s(c_0), \texttt{YVal}_s(c_1)\} = V_1(\mathcal{A}_{\boldsymbol{y}})$.

It remains to show that $V_1$ also prohibits fact $V_2(\texttt{Res}(w_1)) = \texttt{Res}(c_1)$.

By the case assumption, there is an index $j_0 \in \{1, \dots, p\}$ such that the $j_0$-th clause is foul because of the foul variable $x_h$ with minimal index $h$. Without restriction, we assume that $x_h$ is referred to in position 1. Then, valuation $V_1$ prohibits a fact of the form $\texttt{And}(d, \cdot, \cdot, \cdot)$. More concretely, this fact is of the form $\texttt{And}(d, \cdot, \cdot, c_1)$ because $V_1 \leq_{Q_1} W_1$ and valuation $W_1$ by definition only prohibits And-facts with value $d$ that have value $c_1$ in the last position. Thus, $V_1(\alpha_{j_0}) = c_1$. Finally, valuation $W_1$ only prohibits Or-facts from $W_1(\mathcal{A}_{\text{sat}})$, where value $c_1$ in one of the first two positions enforces value $c_1$ in the last position. Thus, we can even conclude $V_1(\omega_{j_0}) = c_1$. By the same argument, this holds for every $j \geq j_0$ and thus for $j = p$ in particular. Therefore, valuation $V_1$ prohibits fact $V_1(\texttt{Res}(\omega_p)) = \texttt{Res}(c_1)$, as claimed.

This concludes the case distinction and proves that every valuation $V_2$ for $Q_2$ has a minimal guarding valuation for $Q_1$.

The mapping provided is thus indeed a polynomial reduction. □

## A.3. Distribution dependencies

The following might be folklore knowledge but the author could not find a suitable reference. In particular, the argument differs technically from the characterisation of Beeri and Vardi [BV84, Theorems 7 and 8] in the following way. The latter authors replace equality-generating dependencies by tuple-generating dependencies. The new tgds result in the addition of all combinations of facts for data values that can be 'unified' by the original egd—without actually unifying them. Instead, we *perform* the unification and, additionally, protocol all relevant unifications in a chase sequence $\boldsymbol{a}$ by a homomorphism $\varepsilon(\boldsymbol{a})$. We feel that our approach gets along better with the syntactical definition of the considered fragments of distribution dependencies.

**Proposition A.7 (Characterisation of implication).** For every set $\Sigma \cup \{\tau\}$ of data-full distribution dependencies, implication $\Sigma \models \tau$ holds if and only if there is a certifying application sequence for $\tau$ under $\Sigma$.

*Proof.* We consider a slightly different implication problem, $\Sigma \models \hat{\tau}$ where $\hat{\tau}$ results from $\tau$ by addition of some atoms that allow to protocol the influence of the application of equality-generating dependencies—*without* affecting the applicability of dependencies.

To this end, let $\xi_1, \dots, \xi_p$ be the node and data variables in $\mathsf{body}(\tau)$. Without restriction, assume that neither $\tau$ nor any dependency in $\Sigma$ refers to relations $\texttt{Var}_1, \dots, \texttt{Var}_p$. Then, the modified dependency $\hat{\tau}$ is defined by

$\mathsf{body}(\hat{\tau}) \stackrel{\text{def}}{=} \mathsf{body}(\tau) \cup \{\mathtt{Var}_1(\xi_1), \ldots, \mathtt{Var}_p(\xi_p)\}$ and
$\mathsf{head}(\hat{\tau}) \stackrel{\text{def}}{=} \mathsf{head}(\tau)$.

We claim that $\Sigma \models \tau$ if and only if $\Sigma \models \hat{\tau}$ and that, for every application sequence $\boldsymbol{a} = (a_1, \ldots, a_n)$, the chase result $\mathbb{D}_n$ contains atom $\mathtt{Var}_i(\xi_j)$ for all $i, j \in \{1, \ldots, p\}$ if and only if $\varepsilon(\xi_i) = \xi_j$ for the homomorphism $\varepsilon = \varepsilon(\boldsymbol{a})$. Note that $\mathbb{D}_n$ contains exactly one $\mathtt{Var}_i$-atom for every $i \in \{1, \ldots, p\}$ and that $\mathsf{body}(\hat{\tau})$ allows exactly the same application sequences as $\mathsf{body}(\tau)$ under $\Sigma$ since no dependency in $\Sigma$ refers to the additional atoms (and thus neither adds such an atom nor requires its presence).

In the following, we show, first, that $\Sigma \models \tau$ if and only if $\Sigma \models \hat{\tau}$, and, second, that $\Sigma \models \hat{\tau}$ if and only if there is an application sequence that certifies $\hat{\tau}$. By the previous argument, the latter holds if and only if the same application sequence certifies $\tau$.

⋙ First step. To show that both statements $\Sigma \models \tau$ and $\Sigma \models \hat{\tau}$ are equivalent, we prove that they mutually imply each other.

» If $\Sigma \models \tau$, then $\Sigma \models \hat{\tau}$. Assume that $\Sigma \models \tau$ holds. Towards a contradiction, assume that $\Sigma \not\models \hat{\tau}$. Then, there is a database $\mathbb{D}$ that satisfies $\Sigma$ and violates $\hat{\tau}$. Let $V$ be a valuation that witnesses the violation of $\hat{\tau}$. In particular, this valuation satisfies $V(\mathsf{body}(\hat{\tau})) \subseteq \mathbb{D}$ and, moreover, $V(\mathsf{body}(\tau)) \subseteq \mathbb{D}$ since $\mathsf{body}(\tau) \subseteq \mathsf{body}(\hat{\tau})$. Depending on the type of $\hat{\tau}$, we can draw the following conclusions.

- *If $\hat{\tau}$ is a degd* with $\mathsf{head}(\hat{\tau}) = (\xi = \eta)$, then $V(\xi) \neq V(\eta)$. But then, valuation $V$ witnesses also that $\mathbb{D}$ violates the original dependency $\tau$.
- *If $\tau$ is a dtgd*, then there is no extension $V'$ of $V$ to the variables in $\mathsf{head}(\hat{\tau})$ such that $V'(\mathsf{head}(\hat{\tau})) \subseteq \mathbb{D}$. But then, there is also no extension $W'$ of $V$ to the variables in $\mathsf{head}(\tau)$ such that $W'(\mathsf{head}(\tau)) \subseteq \mathbb{D}$ because $\mathsf{head}(\hat{\tau}) = \mathsf{head}(\tau)$ and thus $V$ witnesses also that $\mathbb{D}$ violates the original dependency $\tau$.

In both cases, dependency $\tau$ is violated, a contradiction. Therefore, $\Sigma \models \hat{\tau}$ is true.

» If $\Sigma \models \hat{\tau}$, then $\Sigma \models \tau$. For a proof by contraposition, assume that $\Sigma \not\models \tau$. Then, there exists a database $\mathbb{D}$ that satisfies $\Sigma$ but not $\tau$. Without restriction, we assume that $\mathbb{D}$ contains no $\mathtt{Var}_i$-atoms for any $i \in \{1, \ldots, p\}$. More precisely, there is a valuation $V$ that witnesses violation of $\tau$ in $\mathbb{D}$. In particular, $V(\mathsf{body}(\tau)) \subseteq \mathbb{D}$ holds. Furthermore, this valuation witnesses the violation of $\hat{\tau}$ in the extended database

$$\hat{\mathbb{D}} \stackrel{\text{def}}{=} \mathbb{D} \cup \{\mathtt{Var}_1(V(\xi_1)), \ldots, \mathtt{Var}_p(V(\xi_p))\},$$

as the following case distinction shows.

- *If $\hat{\tau}$ is a degd with* $\mathsf{head}(\hat{\tau}) = (\xi = \eta) = \mathsf{head}(\tau)$, then $V(\xi) \neq V(\eta)$ by assumption.

- *If $\hat{\tau}$ is a dtgd*, then there is no extension $V'$ of $V$ to the variables in $\mathsf{head}(\hat{\tau})$ with $V'(\mathsf{head}(\hat{\tau})) \subseteq \mathbb{D}$ because otherwise this would be an extension for $\tau$ too.

Nevertheless, $\hat{\mathbb{D}}$ satisfies $\Sigma$ because $\mathbb{D}$ does and the additional facts are irrelevant for the dependencies in $\Sigma$. Hence, $\hat{\mathbb{D}}$ witnesses $\Sigma \not\models \hat{\tau}$, which was to be shown.

⋙ Second step. We show that $\Sigma \models \hat{\tau}$ if and only if there exists a certifying application sequence for $\hat{\tau}$ under $\Sigma$.

» If. Let $\boldsymbol{a} = (a_1, \ldots, a_n)$ be a certifying application sequence for $\hat{\tau}$ and let $\mathbb{D}$ be an arbitrary database that satisfies $\Sigma$. Towards a contradiction, assume that $\mathbb{D}$ does not satisfy $\hat{\tau}$. Then, there exists a valuation $V$ such that $V(\mathsf{body}(\tau)) \subseteq \mathbb{D}$ such that

- *if $\hat{\tau}$ is a degd with* $\mathsf{head}(\hat{\tau}) = (\xi = \eta)$, then $V(\xi) \neq V(\eta)$; and
- *if $\hat{\tau}$ is a dtgd*, then there exists no extension $V'$ of $V$ such that $V'(\mathsf{head}(\hat{\tau})) \subseteq \mathbb{D}$.

Without loss of generality, we assume that $\mathbb{D}$ contains only those $\mathtt{Var}_i$-facts required by $V$. By Proposition* 5.2.11, the existence of homomorphism $V$ from $\mathsf{body}(\hat{\tau})$ to $\mathbb{D}$ guarantees the existence of a homomorphism $V_n$ from $\mathbb{D}_n$, the chase result for $\boldsymbol{a}$, to database $\mathbb{D}$—at least for *restricted* chase sequences. Indeed, Calì et al. have shown that this is also the case for *oblivous* chase sequences [CGK13, Theorem 2.13].

Clearly, $V(\xi_i) = V_n(\xi_i)$ for every $i \in \{1, \ldots, p\}$ since there are no other $\mathtt{Var}_i$-facts in $\mathbb{D}$ by assumption. Furthermore, since $\boldsymbol{a}$ certifies $\hat{\tau}$, we have

- *if $\hat{\tau}$ is a degd with* $\mathsf{head}(\hat{\tau}) = (\xi = \eta)$, then $\varepsilon(\xi) = \varepsilon(\eta)$ and thus $V_n(\xi) = V_n(\eta)$, which contradicts the choice of $V$; and
- *if $\hat{\tau}$ is a dtgd*, then there exists an extension $V'$ of $V_n$ such that $V'(\mathsf{head}(\hat{\tau})) \subseteq \mathbb{D}$ which contradicts again the choice of $V$.

Therefore, database $\mathbb{D}$ satisfies also $\hat{\tau}$ and, generally, $\Sigma \models \hat{\tau}$.

» Only if. For a proof by contraposition, assume that there is no chase sequence that certifies $\hat{\tau}$. We show that $\Sigma \not\models \hat{\tau}$. Assume that $\boldsymbol{a} = (a_1, \ldots, a_n)$ is a saturated restricted chase sequence starting from $\mathsf{body}(\hat{\tau})$ under $\Sigma$. Then, the corresponding chase result $\mathbb{D}_n$ satisfies $\Sigma$ as does the database $\mathbb{D} \overset{\text{def}}{=} V(\mathbb{D}_n)$ induced by some injective valuation $V$ (otherwise, a violation witnessing valuation could be transformed into a violation witnessing homomorphism for $\mathbb{D}_n$ by composition with $V^{-1}$). Furthermore, $\mathbb{D}$ violates $\hat{\tau}$ (otherwise, it would prove $\boldsymbol{a}$ to certify $\hat{\tau}$, again by composition with $V^{-1}$, a contradiction).

This concludes the proof. □

# Index of Definitions