Theory Comput Syst
DOI 10.1007/s00224-010-9259-6

Query Languages for Data Exchange: Beyond Unions of Conjunctive Queries

Marcelo Arenas · Pablo Barceló · Juan Reutter

Abstract The class of unions of conjunctive queries (UCQ) has been shown to be particularly well-behaved for data exchange; its certain answers can be computed in polynomial time (in terms of data complexity). However, this is not the only class with this property; the certain answers to any DATALOG program can also can be computed in polynomial time. The problem is that both UCQ and DATALOG do not allow negated atoms, as adding an unrestricted form of negation to these languages yields to intractability.

In this paper, we propose a language called DATALOG$^{\mathbf{C}(\neq)}$ that extends DATALOG with a restricted form of negation, and study some of its fundamental properties. In particular, we show that the certain answers to a DATALOG$^{\mathbf{C}(\neq)}$ program can be computed in polynomial time (in terms of data complexity), and that every union of conjunctive queries with at most one inequality or negated relational atom per disjunct, can be efficiently rewritten as a DATALOG$^{\mathbf{C}(\neq)}$ program in the context of data exchange. Furthermore, we show that this is also the case for a syntactic restriction of the class of unions of conjunctive queries with at most two inequalities per disjunct. This syntactic restriction is given by two conditions that are optimal, in the sense that computing certain answers becomes intractable if one removes any of them. Finally, we provide a thorough analysis of the combined complexity of computing certain answers to DATALOG$^{\mathbf{C}(\neq)}$ programs and other related query languages. In particular, we show that this problem is EXPTIME-complete for DATALOG$^{\mathbf{C}(\neq)}$, even if one restricts to conjunctive queries with single inequalities, which is a fragment of DATALOG$^{\mathbf{C}(\neq)}$

M. Arenas (✉)
Dept. of Computer Science, Pontificia Universidad Católica de Chile, Santiago, Chile
e-mail: marenas@ing.puc.cl

P. Barceló
Dept. of Computer Science, Universidad de Chile, Santiago, Chile

J. Reutter
School of Informatics, University of Edinburgh, Edinburgh, UK

Published online: 23 March 2010

by the result mentioned above. Furthermore, we show that the combined complexity is CONEXPTIME-complete for the case of conjunctive queries with k inequalities, for every $k \geq 2$.

Keywords Data exchange · Certain answers · Query languages · Datalog · Combined complexity

1 Introduction

Data exchange is the problem of computing an instance of a *target* schema, given an instance of a *source* schema and a specification of the relationship between source and target data. Although data exchange is considered to be an old database problem, its theoretical foundations have only been laid out very recently by the seminal work of Fagin, Kolaitis, Miller and Popa [9]. Both the study of data exchange and schema mappings have become an active area of research during the last years in the database community (see e.g. [4, 8–10, 13, 14, 18, 19]).

In formal terms, a data exchange setting is a triple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, where $\mathbf{S}$ is a *source* schema, $\mathbf{T}$ is a *target* schema, and Σ_{st} is a mapping defined as a set of *source-to-target* dependencies of the form $\forall \bar{x}(\phi_{\mathbf{S}}(\bar{x}) \rightarrow \exists \bar{y} \psi_{\mathbf{T}}(\bar{x}, \bar{y}))$, where $\phi_{\mathbf{S}}$ and $\psi_{\mathbf{T}}$ are conjunctions of relational atoms over $\mathbf{S}$ and $\mathbf{T}$, respectively (some studies have also included target constraints, but here we focus on data exchange settings without dependencies over $\mathbf{T}$). Given a source instance I, the goal in data exchange is to materialize a target instance J that is a *solution* for I, that is, J together with I must conform to the mapping Σ_{st}.

An important issue in data exchange is that the existing specification languages usually do not completely determine the relationship between source and target data and, thus, there may be many solutions for a given source instance. This immediately raises the question of which solution should be materialized. Initial work on data exchange [9] has identified a class of "good" solutions, called *universal* solutions. In formal terms, a solution is universal if it can be homomorphically embedded into every other solution. It was proved in [9] that for the class of data exchange settings studied in this paper, a particular universal solution—called the *canonical* universal solution—can be computed in polynomial time. It is important to notice that in this result the complexity is measured in terms of the size of the source instance, and the data exchange specification Σ_{st} is assumed to be fixed. Thus, this result is stated in terms of *data* complexity [22].

A second important issue in data exchange is query answering. Queries in the data exchange context are posed over the target schema, and—given that there may be many solutions for a source instance—there is a general agreement in the literature that their semantics should be defined in terms of *certain* answers [1, 9, 15, 16]. More formally, given a data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ and a query Q over $\mathbf{T}$, a tuple $\bar{t}$ is said to be a certain answer to Q over I under $\mathcal{M}$, if $\bar{t}$ belongs to the evaluation of Q over every possible solution J for I under $\mathcal{M}$.

The definition of certain answers is highly non-effective, as it involves computing the intersection of (potentially) infinitely many sets. Thus, it becomes particularly

important to understand for which classes of relevant queries, the certain answers can be computed efficiently. In particular, it becomes relevant to understand whether it is possible to compute the certain answers to any of these classes by using some materialized solution. Fagin, Kolaitis, Miller, and Popa [9] have shown that this is the case for the class of union of conjunctive queries (UCQ); the certain answers to each union of conjunctive queries Q over a source instance I can be computed in polynomial time by directly posing Q over the canonical universal solution for I. Again, it is important to notice that this result is stated in terms of data complexity, that is, the complexity is measured in terms of the size of the source instance, and both the data exchange specification Σ_{st} and the query Q are assumed to be fixed.

The good properties of UCQ for data exchange can be completely explained by the fact that unions of conjunctive queries are preserved under homomorphisms. But this is not the only language that satisfies this condition, as queries definable in DATALOG, the recursive extension of UCQ, are also preserved under homomorphisms. Thus, DATALOG retains several of the good properties of UCQs for data exchange purposes. In particular, the certain answers to a DATALOG program Π over a source instance I, can be computed efficiently by first materializing the canonical universal solution J for I, and then evaluating Π over J (since the data complexity of a DATALOG program is polynomial).

Unfortunately, both UCQ and DATALOG keeps us in the realm of the positive, while most database query languages are equipped with negation. Thus, the first goal of this paper is to investigate what forms of negation can be added to DATALOG while keeping all the good properties of DATALOG, and UCQ, for data exchange. It should be noticed that this is not a trivial problem, as there is a trade-off between expressiveness and complexity in this context. On the one hand, one would like to have a query language expressive enough to be able to pose interesting queries in the data exchange context. But, on the other hand, it has been shown that adding an unrestricted form of negation to DATALOG (or even to conjunctive queries) yields to intractability of the problem of computing certain answers [1, 9]. In this respect, the following are our main contributions.

- We introduce a query language called $\text{DATALOG}^{\mathbf{C}(\neq)}$ that extends DATALOG with a restricted form of negation, and that has the same good properties for data exchange as DATALOG. In particular, we prove that the certain answers to a $\text{DATALOG}^{\mathbf{C}(\neq)}$ program Π over a source instance I can be computed by evaluating Π over the canonical universal solution for I. As a corollary, we obtain that computing certain answers to a $\text{DATALOG}^{\mathbf{C}(\neq)}$ program can be done in polynomial time (in terms of data complexity).
- To show that $\text{DATALOG}^{\mathbf{C}(\neq)}$ can be used to express interesting queries in the data exchange context, we prove that every union of conjunctive queries with at most one inequality or negated relational atom per disjunct, can be efficiently expressed as a $\text{DATALOG}^{\mathbf{C}(\neq)}$ program in the context of data exchange.
- It follows from the previous result that the certain answers to every union of conjunctive queries with at most one inequality or negated relational atom per disjunct, can be computed in polynomial time (in terms of data complexity). Although this corollary is not new (it is a simple extension of a result in [9]), the

use of DATALOG$^{\mathbf{C}(\neq)}$ in the context of data exchange opens the possibility of finding new tractable classes of query languages with negation. In fact, we also use DATALOG$^{\mathbf{C}(\neq)}$ to find a tractable fragment of the class of conjunctive queries with two inequalities.

It is known that for the class of conjunctive queries with inequalities, the problem of computing certain answers is CONP-complete [1, 9] (in terms of data complexity). In fact, it has been shown that the intractability holds even for the case of two inequalities [20]. However, very little is known about tractable fragments of these classes. In this paper, we provide a syntactic restriction for the class of unions of conjunctive queries with at most two inequalities per disjunct, and prove that every query conforming to it can be expressed as a DATALOG$^{\mathbf{C}(\neq)}$ program in the context of data exchange. It immediately follows that the data complexity of computing certain answers to a query conforming to this restriction is polynomial.

The syntactic restriction mentioned above is given by two conditions. We conclude this part of the investigation by showing that these conditions are optimal for tractability, in the sense that computing certain answers becomes intractable if one removes any of them. It should be noticed that this gives a new proof of the fact that the problem of computing certain answer to a conjunctive query with two inequalities is CONP-complete.

The study of the complexity of computing certain answers to DATALOG$^{\mathbf{C}(\neq)}$ programs will not be complete if one does not consider the notion of *combined* complexity. Although the notion of data complexity has shown to be very useful in understanding the complexity of evaluating a query language, one should also study the complexity of this problem when none of its parameters is considered to be fixed. This corresponds to the notion of combined complexity introduced in [22], and it means the following in the context of data exchange. Given a data exchange setting $\mathcal{M}$, a query Q over the target and a source instance I, one considers I as well as Q and $\mathcal{M}$ as part of the input when computing the certain answers to Q over I under $\mathcal{M}$. In this paper, we study this problem and establish the following results.

- We show that the combined complexity of the problem of computing certain answers to DATALOG$^{\mathbf{C}(\neq)}$ programs is EXPTIME-complete, even if one restricts to the class of conjunctive queries with single inequalities (which is a fragment of DATALOG$^{\mathbf{C}(\neq)}$ by the result mentioned above). This refines a result in [14] that shows that the combined complexity of the problem of computing certain answers to *unions* of conjunctive queries with at most one inequality per disjunct is EXPTIME-complete.
- We also consider the class of conjunctive queries with an arbitrary number of inequalities per disjunct. More specifically, we show that the combined complexity of the problem of computing certain answers is CONEXPTIME-complete for the case of conjunctive queries with k inequalities, for every $k \geq 2$.
- One of the reasons for the high combined complexity of the previous problems is the fact that if data exchange settings are not considered to be fixed, then one has to deal with canonical universal solutions of exponential size. A natural way to reduce the size of these solutions is to focus on the class of LAV data exchange settings [16], which are frequently used in practice.

For the case of DATALOG$^{\mathbf{C}(\neq)}$ programs, the combined complexity is inherently exponential, and thus focusing on LAV settings does not reduce the complexity of computing certain answers. However, we show in the paper that if one focuses on LAV settings, then the combined complexity is considerably lower for the class of conjunctive queries with inequalities. More specifically, we show that the combined complexity goes down to NP-complete for the case of conjunctive queries with single inequalities, and to Π_2^p-complete for the case of conjunctive queries with k inequalities, for every $k \geq 2$.

Proviso. As we mentioned above, target dependencies are usually considered in the data exchange literature in addition to source-to-target dependencies. Those target dependencies represent the usual database constraints that exchanged data must satisfy. We decided not to include target dependencies in this work for the sake of readability, but we certainly think that this is a class that deserves attention. In fact, we are currently working on extending the setting presented in this paper to take into account usual target constraints studied in the data exchange literature (e.g. *equality-generating* dependencies and *tuple-generating* dependencies).

Organization of the paper. In Sect. 2, we introduce the terminology used in the paper. In Sect. 3, we define the syntax and semantics of DATALOG$^{\mathbf{C}(\neq)}$ programs. In Sect. 4, we study some of the fundamental properties of DATALOG$^{\mathbf{C}(\neq)}$ programs concerning complexity and expressiveness. In Sect. 5, we study a syntactic restriction that leads to tractability of the problem of computing certain answers for unions of conjunctive queries with two inequalities. In Sect. 6, we provide a thorough analysis of the combined complexity of computing certain answers to DATALOG$^{\mathbf{C}(\neq)}$ programs and other related query languages. Concluding remarks are in Sect. 7.

2 Background

A *schema* $\mathbf{R}$ is a finite set $\{R_1, \ldots, R_k\}$ of relation symbols, with each R_i having a fixed arity $n_i > 0$. Let $\mathbf{D}$ be a countably infinite domain. An *instance* I of $\mathbf{R}$ assigns to each relation symbol R_i of $\mathbf{R}$ a finite n_i-ary relation $R_i^I \subseteq \mathbf{D}^{n_i}$. The *domain* $\mathrm{dom}(I)$ of instance I is the set of all elements that occur in any of the relations R_i^I. We often define instances by simply listing the tuples attached to the corresponding relation symbols.

We assume familiarity with first-order logic (FO) and DATALOG. In this paper, CQ is the class of conjunctive queries and UCQ is the class of unions of conjunctive queries. If we extend these classes by allowing inequalities or negation (of relational atoms), then we use superscripts $\neq$ and $\neg$, respectively. Thus, for example, CQ$^{\neq}$ is the class of conjunctive queries with inequalities, and UCQ$^{\neg}$ is the class of unions of conjunctive queries with negated relational atoms but no inequalities. As usual in the database literature, we assume that every query Q in UCQ$^{\neq,\neg}$ is *safe*: (1) if Q_1 and Q_2 are disjuncts of Q, then Q_1 and Q_2 have the same free variables, (2) if Q_1 is a disjunct of Q and $x \neq y$ is a conjunct of Q_1, then x and y appear in some non-negated relational atoms of Q_1, (3) if Q_1 is a disjunct of Q and $\neg R(\bar{x})$ is a conjunct of Q_1, then every variable in $\bar{x}$ appears in a non-negated relational atom of Q_1.

2.1 Data Exchange Settings and Solutions

As is customary in the data exchange literature, we consider instances with two types of values: constants and nulls [9, 10]. More precisely, let $\mathbf{C}$ and $\mathbf{N}$ be infinite and disjoint sets of constants and nulls, respectively, and assume that $\mathbf{D} = \mathbf{C} \cup \mathbf{N}$. If we refer to a schema $\mathbf{S}$ as a *source* schema, then whenever we consider an instance I of $\mathbf{S}$, we will assume that $\mathrm{dom}(I) \subseteq \mathbf{C}$. On the other hand, if we refer to a schema $\mathbf{T}$ as a *target* schema, then for every instance J of $\mathbf{T}$, it holds that $\mathrm{dom}(J) \subseteq \mathbf{C} \cup \mathbf{N}$. Slightly abusing notation, we also use $\mathbf{C}$ to denote a built-in unary predicate such that $\mathbf{C}(a)$ holds if and only if a is a constant, that is $a \in \mathbf{C}$ refer to a source schema and $\mathbf{T}$ to

A *data exchange setting* is a tuple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, where $\mathbf{S}$ is a source schema, $\mathbf{T}$ is a target schema, $\mathbf{S}$ and $\mathbf{T}$ do not have predicate symbols in common and Σ_{st} is a set of FO-dependencies over $\mathbf{S} \cup \mathbf{T}$ (in [9] and [10] a more general class of data exchange settings is presented, that also includes *target* dependencies). As usual in the data exchange literature (e.g., [9, 10]), we restrict the study to data exchange settings in which Σ_{st} consists of a set of *source-to-target tuple-generating* dependencies. A source-to-target tuple-generating dependency (st-tgd) is an FO-sentence of the form $\forall \bar{x}\,(\phi(\bar{x}) \rightarrow \exists \bar{y}\,\psi(\bar{x}, \bar{y}))$, where $\phi(\bar{x})$ is a conjunction of relational atoms over $\mathbf{S}$ and $\psi(\bar{x}, \bar{y})$ is a conjunction of relational atoms over $\mathbf{T}$.[1] A *source* (resp. *target*) instance K for $\mathcal{M}$ is an instance of $\mathbf{S}$ (resp. $\mathbf{T}$). We usually denote source instances by $I, I', I_1, \ldots$, and target instances by $J, J', J_1, \ldots$.

The class of data exchange settings considered in this paper is usually called GLAV (global-&-local-as-view) in the database literature [16]. One of the restricted forms of this class that has been extensively studied for data integration and exchange is the class of LAV settings. Formally, a LAV setting (local-as-view) [16] is a data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, in which every st-tgd in Σ_{st} is of the form $\forall \bar{x}\,(S(\bar{x}) \rightarrow \psi(\bar{x}))$, for some $S \in \mathbf{S}$ (it is important to notice that variables of tuple $\bar{x}$ are not assumed to be pairwise distinct).

An instance J of $\mathbf{T}$ is said to be a *solution* for an instance I under $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, if the instance $K = (I, J)$ of $\mathbf{S} \cup \mathbf{T}$ satisfies Σ_{st}, where $S^K = S^I$ for every $S \in \mathbf{S}$ and $T^K = T^J$ for every $T \in \mathbf{T}$. If $\mathcal{M}$ is clear from the context, we shall say that J is a solution for I.

Example 2.1 Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a data exchange setting. Assume that $\mathbf{S}$ consists of one binary relation symbol P, and $\mathbf{T}$ consists of two binary relation symbols Q and R. Further, assume that Σ_{st} consists of st-tgds $P(x, y) \rightarrow Q(x, y)$ and $P(x, y) \rightarrow \exists z R(x, z)$. Then $\mathcal{M}$ is also a LAV setting.

Let $I = \{P(a, b),\ P(a, c)\}$ be a source instance. Then $J_1 = \{Q(a, b),\ Q(a, c),\ R(a, b)\}$ and $J_2 = \{Q(a, b),\ Q(a, c),\ R(a, n)\}$, where $n \in \mathbf{N}$, are solutions for I. In fact, I has infinitely many solutions.

[1] We usually omit universal quantification in front of st-tgds and express them simply as $\phi(\bar{x}) \rightarrow \exists \bar{y}\,\psi(\bar{x}, \bar{y})$.

2.2 Universal Solutions and Canonical Universal Solution

It has been argued in [9] that the preferred solutions in data exchange are the *universal* solutions. In order to define this notion, we first have to revise the concept of *homomorphism* in data exchange. Let K_1 and K_2 be instances of the same schema $\mathbf{R}$. A *homomorphism* h from K_1 to K_2 is a function $h : \text{dom}(K_1) \to \text{dom}(K_2)$ such that: (1) $h(c) = c$ for every $c \in \mathbf{C} \cap \text{dom}(K_1)$, and (2) for every $R \in \mathbf{R}$ and every tuple $\bar{a} = (a_1, \ldots, a_k) \in R^{K_1}$, it holds that $h(\bar{a}) = (h(a_1), \ldots, h(a_k)) \in R^{K_2}$. Notice that this definition of homomorphism slightly differs from the usual one, as the additional constraint that homomorphisms are the identity on the constants is imposed.

Let $\mathcal{M}$ be a data exchange setting, I a source instance and J a solution for I under $\mathcal{M}$. Then J is a *universal solution* for I under $\mathcal{M}$, if for every solution J' for I under $\mathcal{M}$, there exists a homomorphism from J to J'.

Example 2.2 (Example 2.1 continued) Solution J_2 is a universal solution for I, while J_1 is not since there is no homomorphism from J_1 to J_2.

It follows from [9] that for the class of data exchange settings studied in this paper, every source instance has universal solutions. In particular, one of these solutions—called the *canonical universal solution*—can be constructed in polynomial time from the given source instance (assuming the setting to be fixed), using the *chase* procedure [5]. We shall define canonical universal solutions directly as in [4, 18].

In what follows, we show how to compute the canonical universal solution of a source instance I in a data exchange setting $(\mathbf{S}, \mathbf{T}, \Sigma_{st})$. For each st-tgd in Σ_{st} of the form:

$$\phi(\bar{x}, \bar{y}) \quad \to \quad \exists \bar{w}\, (T_1(\bar{x}_1, \bar{w}_1) \wedge \cdots \wedge T_k(\bar{x}_k, \bar{w}_k)),$$

where $\bar{x} = \bar{x}_1 \cup \cdots \cup \bar{x}_k$ and $\bar{w} = \bar{w}_1 \cup \cdots \cup \bar{w}_k$, and for each tuple $\bar{a}$ from $\text{dom}(I)$ of length $|\bar{x}|$, find all tuples $\bar{b}_1, \ldots, \bar{b}_m$ such that $I \models \phi(\bar{a}, \bar{b}_i)$, $i \in [1, m]$. Then choose m tuples $\bar{n}_1, \ldots, \bar{n}_m$ of length $|\bar{w}|$ of fresh distinct null values over $\mathbf{N}$. Relation T_i ($i \in [1, k]$) in the canonical universal solution for I contains tuples $(\pi_{\bar{x}_i}(\bar{a}), \pi_{\bar{w}_i}(\bar{n}_j))$, for each $j \in [1, m]$, where $\pi_{\bar{x}_i}(\bar{a})$ refers to the components of $\bar{a}$ that occur in the positions of $\bar{x}_i$. Furthermore, relation T_i in the canonical universal solution for I only contains tuples that are obtained by applying this algorithm.

Notice that the algorithm for constructing the canonical universal solution, as defined above, corresponds to what is known as the *naïve* chase applied to the st-tgds in the setting. In the naïve chase all dependencies are fired in parallel. Our definition differs from the one given in [9], where a canonical universal solution is obtained by using the *standard* chase procedure. The standard chase procedure fires st-tgds one by one, but only populates the target instance J with new facts $T(\bar{t})$ such that $T(\bar{t})$ cannot be *deduced* from J itself. The problem with using the standard chase in data exchange is that its result is not necessarily unique (it depends on the order in which the chase steps are applied), and thus, there may be multiple non-isomorphic canonical universal solutions. Clearly, under our definition, the canonical universal solution is unique up to isomorphism and can be computed in polynomial time from I. For a fixed data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, we denote by CAN the transformation

from source instances to target instances, such that $\text{CAN}(I)$ is the canonical universal solution for I under $\mathcal{M}$.

2.3 Certain Answers

Queries in a data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ are posed over the target schema $\mathbf{T}$. Given that there may be (even infinitely) many solutions for a given source instance I with respect to $\mathcal{M}$, the standard approach in the data exchange literature is to define the semantics of the query based on the notion of certain answers [1, 9, 15, 16].

Let I be a source instance. For a query Q of arity $n \geq 0$, in any of our logical formalisms, we denote by $\text{certain}_{\mathcal{M}}(Q, I)$ the set of *certain answers* of Q over I under $\mathcal{M}$, that is, the set of n-tuples $\bar{t}$ such that $\bar{t} \in Q(J)$, for every J that is a solution for I under $\mathcal{M}$. If $n = 0$, then we say that Q is *Boolean*, and $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{true}$ iff Q holds for every J that is a solution for I under $\mathcal{M}$. We write $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$ if it is not the case that $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{true}$.

Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a data exchange setting and Q a query over $\mathbf{T}$. The main problem studied in this paper is:

PROBLEM	:	CERTAIN-ANSWERS($\mathcal{M}$, Q).
INPUT	:	A source instance I and a tuple $\bar{t}$ of constants from I.
QUESTION	:	Is $\bar{t} \in \text{certain}_{\mathcal{M}}(Q, I)$?

Since in the above definition both the setting and the query are fixed, it corresponds (in terms of Vardi's taxonomy [22]) to the *data* complexity of the problem of computing certain answers. Later, in Sect. 6, we also study the *combined* complexity of this problem.

3 Extending Query Languages for Data Exchange: DATALOG$^{\text{C}(\neq)}$ Programs

The class of unions of conjunctive queries is particularly well-behaved for data exchange; the certain answers of each union of conjunctive queries Q can be computed by directly posing Q over an arbitrary universal solution [9]. More formally, given a data exchange setting $\mathcal{M}$, a source instance I, a universal solution J for I under $\mathcal{M}$, and a tuple $\bar{t}$ of constants, $\bar{t} \in \text{certain}_{\mathcal{M}}(Q, I)$ if and only if $\bar{t} \in Q(J)$. This implies that for each data exchange setting $\mathcal{M}$, the problem CERTAIN-ANSWERS($\mathcal{M}$, Q) can be solved in polynomial time if Q is a union of conjunctive queries (because the canonical universal solution for I can be computed in polynomial time and Q has polynomial time data complexity).

The fact that the certain answers of a union of conjunctive queries Q can be computed by posing Q over a universal solution, can be fully explained by the fact that Q is *preserved* under homomorphisms, that is, for every pair of instances J, J', homomorphism h from J to J', and tuple $\bar{a}$ of elements in J, if $\bar{a} \in Q(J)$, then $h(\bar{a}) \in Q(J')$. But UCQ is not the only class of queries that is preserved under homomorphisms; also DATALOG, the *recursive* extension of the class UCQ, has this

property. Since DATALOG has polynomial time data complexity, we have that the certain answers of each DATALOG query Q can be obtained efficiently by first computing a universal solution J, and then evaluating Q over J. Thus, DATALOG preserves all the good properties of UCQ for data exchange.

Unfortunately, both UCQ and DATALOG keep us in the realm of the positive (i.e. negated atoms are not allowed in queries), while most database query languages are equipped with negation. It seems then natural to extend UCQ (or DATALOG) in the context of data exchange with some form of negation. Indeed, query languages with different forms of negation have been considered in the data exchange context [3, 7], as they can be used to express interesting queries. Next, we show an example of this fact.

Example 3.1 Consider a data exchange setting with $\mathbf{S} = \{E(\cdot,\cdot), A(\cdot), B(\cdot)\}$, $\mathbf{T} = \{G(\cdot,\cdot), P(\cdot), R(\cdot)\}$ and

$$\Sigma_{st} = \{E(x,y) \rightarrow G(x,y), A(x) \rightarrow P(x), B(x) \rightarrow R(x)\}.$$

Notice that if I is a source instance, then the canonical universal solution $\text{CAN}(I)$ for I is such that $E^I = G^{\text{CAN}(I)}$, $A^I = P^{\text{CAN}(I)}$ and $B^I = R^{\text{CAN}(I)}$.

Let $Q(x)$ be the following $\text{UCQ}^\neg$ query over $\mathbf{T}$:

$$\exists x \exists y (P(x) \wedge R(y) \wedge G(x,y)) \vee \exists x \exists y \exists z (G(x,z) \wedge G(z,y) \wedge \neg G(x,y)).$$

It is not hard to prove that for every source instance I, $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{true}$ iff there exist elements $a, b \in \text{dom}(\text{CAN}(I))$ such that a belongs to $P^{\text{CAN}(I)}$, b belongs to $R^{\text{CAN}(I)}$ and (a,b) belongs to the transitive closure of the relation $G^{\text{CAN}(I)}$. That is, $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{true}$ iff there exist elements $a, b \in \text{dom}(I)$ such that a belongs to A^I, b belongs to B^I and (a,b) belongs to the transitive closure of the relation E^I.

It is well-known (see e.g. [17]) that there is no union of conjunctive queries (indeed, not even an FO-query) that defines the transitive closure of a graph. Thus, if Q and $\mathcal{M}$ are as in the previous example, then there is no union of conjunctive queries Q' such that $Q'(\text{CAN}(I)) = \text{certain}_{\mathcal{M}}(Q', I) = \text{certain}_{\mathcal{M}}(Q, I)$, for every source instance I. It immediately follows that negated relational atoms add expressive power to the class UCQ in the context of data exchange (see also [4]). And not only that, it follows from [9] that inequalities also add expressive power to UCQ in the context of data exchange.

In this section, we propose a language that can be used to pose queries with negation, and that has all the good properties of UCQ for data exchange.

3.1 $\text{DATALOG}^{\mathbf{C}(\neq)}$ Programs

Unfortunately, adding an unrestricted form of negation to DATALOG (or even to CQ) not only destroys preservation under homomorphisms, but also easily yields to intractability of the problem of computing certain answers [1, 9]. More precisely, there is a setting $\mathcal{M}$ and a query Q in $\text{CQ}^{\neq}$ such that the problem $\text{CERTAIN-ANSWERS}(\mathcal{M}, Q)$ cannot be solved in polynomial time (unless PTIME =

NP). In particular, the set of certain answers of Q cannot be computed by evaluating Q over a polynomial-time computable universal solution. Next we show that there is a natural way of adding negation to DATALOG while keeping all of the good properties of this language for data exchange. In Sect. 4, we show that such a restricted form of negation can be used to express many relevant queries (some including negation) for data exchange.

Definition 3.2 (DATALOG$^{\mathbf{C}(\neq)}$ programs) A *constant-inequality Datalog rule* is a rule of the form:

$$S(\bar{x}) \quad \leftarrow \quad S_1(\bar{x}_1), \ldots, S_\ell(\bar{x}_\ell), \mathbf{C}(y_1), \ldots, \mathbf{C}(y_m), u_1 \neq v_1, \ldots, u_n \neq v_n,$$

where

(a) $S, S_1, \ldots, S_\ell$ are (non necessarily distinct) predicate symbols,
(b) every variable in $\bar{x}$ is mentioned in some tuple $\bar{x}_i$ $(i \in [1, \ell])$,
(c) every variable y_j $(j \in [1, m])$ is mentioned in some tuple $\bar{x}_i$ $(i \in [1, \ell])$, and
(d) every variable u_j $(j \in [1, n])$, and every variable v_j, is equal to some variable y_i $(i \in [1, m])$.

Moreover, a *constant-inequality Datalog program* (DATALOG$^{\mathbf{C}(\neq)}$ program) Π is a finite set of constant-inequality Datalog rules.

For example, the following is a constant-inequality Datalog program:

$$\begin{aligned} R(x, y) \quad &\leftarrow \quad T(x, z), S(z, y), \mathbf{C}(x), \mathbf{C}(z), x \neq z, \\ S(x) \quad &\leftarrow \quad U(x, u, v, w), \mathbf{C}(x), \mathbf{C}(u), \mathbf{C}(v), \mathbf{C}(w), u \neq v, u \neq w. \end{aligned}$$

For a rule of the form (3.2), we say that $S(\bar{x})$ is its head. The set of predicates of a DATALOG$^{\mathbf{C}(\neq)}$ program Π, denoted by $Pred(\Pi)$, is the set of predicate symbols mentioned in Π, while the set of intensional predicates of Π, denoted by $IPred(\Pi)$, is the set of predicates symbols $R \in Pred(\Pi)$ such that $R(\bar{x})$ appears as the head of some rule of Π.

Fix a DATALOG$^{\mathbf{C}(\neq)}$ program Π and let I be a database instance of the relational schema $Pred(\Pi)$. Then $\mathcal{T}(I)$ is an instance of $Pred(\Pi)$ such that for every $R \in Pred(\Pi)$ and every tuple $\bar{t}$, it holds that $\bar{t} \in R^{\mathcal{T}(I)}$ if and only if there exists a rule $R(\bar{x}) \leftarrow R_1(\bar{x}_1), \ldots, R_\ell(\bar{x}_\ell), \mathbf{C}(y_1), \ldots, \mathbf{C}(y_m), u_1 \neq v_1, \ldots, u_n \neq v_n$ in Π and a variable assignment σ such that (a) $\sigma(\bar{x}) = \bar{t}$, (b) $\sigma(\bar{x}_i) \in R_i^I$, for every $i \in [1, \ell]$, (c) $\sigma(y_i)$ is a constant, for every $i \in [1, m]$, and (d) $\sigma(u_i) \neq \sigma(v_i)$, for every $i \in [1, n]$. Operator $\mathcal{T}$ is used to define the semantics of constant-inequality Datalog programs. More precisely, define $\mathcal{T}_\Pi^0(I)$ to be I and $\mathcal{T}_\Pi^{n+1}(I)$ to be $\mathcal{T}(\mathcal{T}_\Pi^n(I)) \cup \mathcal{T}_\Pi^n(I)$, for every $n \geq 0$. Then the evaluation of Π over I is defined as $\mathcal{T}_\Pi^\infty(I) = \bigcup_{n \geq 0} \mathcal{T}_\Pi^n(I)$.

A constant-inequality Datalog program Π is said to be defined over a relational schema $\mathbf{R}$ if $\mathbf{R} = Pred(\Pi) \setminus IPred(\Pi)$ and ANSWER $\in IPred(\Pi)$. Given an instance I of $\mathbf{R}$ and a tuple $\bar{t}$ in $\mathrm{dom}(I)^n$, where n is the arity of ANSWER, we say that $\bar{t} \in \Pi(I)$ if $\bar{t} \in \text{ANSWER}^{\mathcal{T}_\Pi^\infty(I_0)}$, where I_0 is an extension of I defined as: $R^{I_0} = R^I$ for $R \in \mathbf{R}$ and $R^{I_0} = \emptyset$ for $R \in IPred(\Pi)$.

As we mentioned before, the homomorphisms in data exchange are not arbitrary; they are the identity on the constants. Thus, given that inequalities are witnessed by constants in DATALOG$^{\mathbf{C}(\neq)}$ programs, we have that these programs are preserved under homomorphisms. From this we conclude that the certain answers to a DATALOG$^{\mathbf{C}(\neq)}$ program Π can be computed by directly evaluating Π over a universal solution.

Proposition 3.3 *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a data exchange setting, I a source instance, J a universal solution for I under $\mathcal{M}$, and Π a* DATALOG$^{\mathbf{C}(\neq)}$ *program over $\mathbf{T}$. Then for every tuple $\bar{t}$ of constants, $\bar{t} \in \text{certain}_{\mathcal{M}}(\Pi, I)$ iff $\bar{t} \in \Pi(J)$.*

This proposition will be used in Sect. 4 to show that DATALOG$^{\mathbf{C}(\neq)}$ programs preserve the good properties of conjunctive queries for data exchange.

4 On the Complexity and Expressiveness of DATALOG$^{\mathbf{C}(\neq)}$ Programs

We start this section by studying the expressive power of DATALOG$^{\mathbf{C}(\neq)}$ programs. In particular, we show that these programs are expressive enough to capture the class of unions of conjunctive queries with at most one negated atom per disjunct. This class has proved to be relevant for data exchange, as its restriction with inequalities (that is the class of queries in UCQ$^{\neq}$ with at most one inequality per disjunct) not only can express relevant queries but also is one of the few known extensions of the class UCQ for which the problem of computing certain answers is tractable [9]. Indeed, as it is shown in [9], this class remains tractable even in the presence of restricted classes of target dependencies.

Theorem 4.1 *Let Q be a* UCQ$^{\neq,\neg}$ *query over a schema $\mathbf{T}$, with at most one inequality or negated relational atom per disjunct. Then there exists a* DATALOG$^{\mathbf{C}(\neq)}$ *program Π_Q over $\mathbf{T}$ such that for every data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ and instance I of $\mathbf{S}$, $\text{certain}_{\mathcal{M}}(Q, I) = \text{certain}_{\mathcal{M}}(\Pi_Q, I)$. Moreover, Π_Q can be effectively constructed from Q in polynomial time.*

Before presenting the proof of Theorem 4.1, we sketch the proof by means of an example.

Example 4.2 Let $\mathcal{M}$ be a data exchange setting such that $\mathbf{S} = \{E(\cdot,\cdot), A(\cdot)\}$, $\mathbf{T} = \{G(\cdot,\cdot), P(\cdot)\}$ and

$$\Sigma_{st} = \{E(x, y) \rightarrow \exists z(G(x, z) \wedge G(z, y)), A(x) \rightarrow P(x)\}.$$

Also, let $Q(x)$ be the following query in UCQ$^{\neq,\neg}$:

$$(P(x) \wedge G(x, x)) \vee \exists y(G(x, y) \wedge x \neq y) \vee \exists y \exists z(G(x, z) \wedge G(z, y) \wedge \neg G(x, y)).$$

We construct a DATALOG$^{\mathbf{C}(\neq)}$ program Π_Q such that $\text{certain}_{\mathcal{M}}(Q, I) = \text{certain}_{\mathcal{M}}(\Pi_Q, I)$. The set of intensional predicates of the DATALOG$^{\mathbf{C}(\neq)}$ program

Π_Q is $\{U_1(\cdot,\cdot,\cdot), U_2(\cdot,\cdot), \text{dom}(\cdot), \text{EQUAL}(\cdot,\cdot,\cdot), \text{ANSWER}(\cdot)\}$. The program Π_Q over $\mathbf{T}$ is defined as follows.

- First, the program collects in $\text{dom}(x)$ all the elements that belong to the active domain of the instance of $\mathbf{T}$ where Π_Q is evaluated:

$$\text{dom}(x) \leftarrow G(x,z), \tag{1}$$
$$\text{dom}(x) \leftarrow G(z,x), \tag{2}$$
$$\text{dom}(x) \leftarrow P(x). \tag{3}$$

- Second, the program Π_Q includes the following rules that formalize the idea that $\text{EQUAL}(x,y,z)$ holds if x and y are the same elements:

$$\text{EQUAL}(x,x,z) \leftarrow \text{dom}(x), \text{dom}(z), \tag{4}$$
$$\text{EQUAL}(x,y,z) \leftarrow \text{EQUAL}(y,x,z), \tag{5}$$
$$\text{EQUAL}(x,y,z) \leftarrow \text{EQUAL}(x,w,z), \text{EQUAL}(w,y,z). \tag{6}$$

  Predicate EQUAL includes an extra argument that keeps track of the element z where the query is being evaluated. Notice that we cannot simply use the rule $\text{EQUAL}(x,x,z) \leftarrow$ to say that EQUAL is reflexive, as $\text{DATALOG}^{\mathbf{C}(\neq)}$ programs are *safe*, i.e. every variable that appears in the head of a rule also has to appear in its body.
- Third, Π_Q includes the rules:

$$U_1(x,y,z) \leftarrow G(x,y), \text{dom}(z), \tag{7}$$
$$U_2(x,z) \leftarrow P(x), \text{dom}(z), \tag{8}$$
$$U_1(x,y,z) \leftarrow U_1(u,v,z), \text{EQUAL}(u,x,z), \text{EQUAL}(v,y,z), \tag{9}$$
$$U_2(x,z) \leftarrow U_2(u,z), \text{EQUAL}(u,x,z). \tag{10}$$

  Intuitively, the first two rules create in U_1 and U_2 a copy of G and P, respectively, but again with an extra argument for keeping track of the element where Π_Q is being evaluated. The last two rules allow to replace equal elements in the interpretation of U_1 and U_2.
- Fourth, Π_Q includes the following rule for the third disjunct of $Q(x)$:

$$U_1(x,y,x) \leftarrow U_1(x,z,x), U_1(z,y,x). \tag{11}$$

  Intuitively, this rule expresses that if a is an element that does not belong to the set of certain answers to $Q(x)$, then for every pair of elements b and c such that (a,b) and (b,c) belong to the interpretation of G, it must be the case that (a,c) also belongs to it.
- Fifth, Π_Q includes the following rule for the second disjunct of $Q(x)$:

$$\text{EQUAL}(x,y,x) \leftarrow U_1(x,y,x). \tag{12}$$

Intuitively, this rule expresses that if a is an element that does not belong to the set of certain answers to $Q(x)$, then for every element b such that the pair (a, b) belongs to the interpretation of G, it must be the case that $a = b$.

- Finally, Π_Q includes two rules for collecting the certain answers to $Q(x)$:

$$\text{ANSWER}(x) \quad \leftarrow \quad U_2(x, x), U_1(x, x, x), \mathbf{C}(x), \tag{13}$$

$$\text{ANSWER}(x) \quad \leftarrow \quad \text{EQUAL}(y, z, x), \mathbf{C}(y), \mathbf{C}(z), y \neq z. \tag{14}$$

Intuitively, rule (13) says that if a constant a belongs to the interpretation of P and (a, a) belongs to the interpretation of G, then a belongs to the set of certain answers to $Q(x)$. Indeed, this means that if J is an arbitrary solution where the program is being evaluated, then a belongs to the evaluation of the first disjunct of $Q(x)$ over J.

Rule (14) says that if in the process of evaluating Π_Q with parameter a, two distinct constants b and c are declared to be equal ($\text{EQUAL}(b, c, a)$ holds), then a belongs to the set of certain answers to $Q(x)$. We show the application of this rule with an example. Let I be a source instance, and assume that (a, n) and (n, b) belong to G in the canonical universal solution for I, where n is a null value. By applying rule (1), we have that $\text{dom}(a)$ holds in $\text{CAN}(I)$. Thus, we conclude by applying rule (7) that $U_1(a, n, a)$ and $U_1(n, b, a)$ hold in $\text{CAN}(I)$ and, therefore, we obtain by using rule (12) that $\text{EQUAL}(a, n, a)$ holds in $\text{CAN}(I)$. Notice that this rule is trying to prove that a is not in the certain answers to $Q(x)$ and, hence, it forces n to be equal to a. Now by using rule (5), we obtain that $\text{EQUAL}(n, a, a)$ holds in $\text{CAN}(I)$. But we also have that $\text{EQUAL}(b, b, a)$ holds in $\text{CAN}(I)$ (by applying rules (2) and (4)). Thus, by applying rule (9), we obtain that $U_1(a, b, a)$ holds in $\text{CAN}(I)$. Therefore, by applying rule (12) again, we obtain that $\text{EQUAL}(a, b, a)$ holds in $\text{CAN}(I)$. This time, rule (12) tries to prove that a is not in the certain answers to $Q(x)$ by forcing constants a and b to be the same value. But this cannot be the case since a and b are distinct constants and, thus, rule (14) is used to conclude that a is in the certain answers to $Q(x)$. It is important to notice that this conclusion is correct. If J is an arbitrary solution for I, then we have that there exists a homomorphism $h : \text{CAN}(I) \rightarrow J$. Given that a and b are distinct constants, we have that $a \neq h(n)$ or $b \neq h(n)$. It follows that there is an element c in J such that $a \neq c$ and the pair (a, c) belongs to the interpretation of G. Thus, we conclude that a belongs to the evaluation of the second disjunct of $Q(x)$ over J.

It is now an easy exercise to show that the set of certain answers to $Q(x)$ coincide with the set of certain answers to Π_Q, for every source instance I.

We now present the proof of Theorem 4.1.

Proof Assume that $\mathbf{T} = \{T_1, \ldots, T_k\}$, where each T_i has arity $n_i > 0$, and that $Q(\bar{x}) = Q_1(\bar{x}) \vee \cdots \vee Q_\ell(\bar{x})$, where $\bar{x} = (x_1, \ldots, x_m)$ and each $Q_i(\bar{x})$ is a conjunctive query with at most one inequality or negated relational atom. Then the set of intensional predicates of $\text{DATALOG}^{\mathbf{C}(\neq)}$ program Π_Q is

$$\{U_1, \ldots, U_k, \text{DOM}, \text{EQUAL}, \text{ANSWER}\},$$

where each U_i ($i \in [1,k]$) has arity $n_i + m$, DOM has arity 1, EQUAL has arity $2+m$ and ANSWER has arity m. Moreover, the set of rules of Π_Q is defined as follows.

- For every predicate $T_i \in \mathbf{T}$, Π_Q includes the following k rules:

$$\begin{aligned}
\textsc{Dom}(x) &\leftarrow T_i(x, y_2, y_3, \ldots, y_{n_i-1}, y_{n_i}),\\
\textsc{Dom}(x) &\leftarrow T_i(y_1, x, y_3, \ldots, y_{n_i-1}, y_{n_i}),\\
&\ldots\\
\textsc{Dom}(x) &\leftarrow T_i(y_1, y_2, y_3, \ldots, y_{n_i-1}, x).
\end{aligned}$$

- Π_Q includes the following rules for predicate EQUAL:

$$\begin{aligned}
\textsc{Equal}(x, x, z_1, \ldots, z_m) &\leftarrow \textsc{Dom}(x), \textsc{Dom}(z_1), \ldots, \textsc{Dom}(z_m),\\
\textsc{Equal}(x, y, z_1, \ldots, z_m) &\leftarrow \textsc{Equal}(y, x, z_1, \ldots, z_m),\\
\textsc{Equal}(x, y, z_1, \ldots, z_m) &\leftarrow \textsc{Equal}(x, w, z_1, \ldots, z_m),\\
&\quad\ \textsc{Equal}(w, y, z_1, \ldots, z_m).
\end{aligned}$$

- For every predicate U_i, Π_Q includes the following rules:

$$\begin{aligned}
U_i(y_1, \ldots, y_{n_i}, z_1, \ldots, z_m) &\leftarrow T_i(y_1, \ldots, y_{n_i}), \textsc{Dom}(z_1), \ldots, \textsc{Dom}(z_m),\\
U_i(y_1, \ldots, y_{n_i}, z_1, \ldots, z_m) &\leftarrow U_i(w_1, \ldots, w_{n_i}, z_1, \ldots, z_m),\\
&\quad\ \textsc{Equal}(w_1, y_1, z_1, \ldots, z_m), \ldots,\\
&\quad\ \textsc{Equal}(w_{n_i}, y_{n_i}, z_1, \ldots, z_m).
\end{aligned}$$

- Let $i \in [1, \ell]$. First, assume that $Q_i(\bar{x})$ does not contain any negated atom. Then $Q_i(\bar{x})$ is equal to $\exists \bar{u}(T_{p_1}(\bar{u}_1) \wedge \cdots \wedge T_{p_n}(\bar{u}_n))$, where $p_j \in [1,k]$ and every variable in $\bar{u}_j$ is mentioned in either $\bar{u}$ or $\bar{x}$, for every $j \in [1,n]$. In this case, program Π_Q includes the following rule:

$$\textsc{Answer}(\bar{x}) \leftarrow U_{p_1}(\bar{u}_1, \bar{x}), \ldots, U_{p_n}(\bar{u}_n, \bar{x}), \mathbf{C}(x_1), \ldots, \mathbf{C}(x_m). \quad (15)$$

  Notice that this rule is well defined since the set $\bar{x}$ is the set of free variables of $\exists \bar{u}(T_{p_1}(\bar{u}_1) \wedge \cdots \wedge T_{p_n}(\bar{u}_n))$. Second, assume that $Q_i(\bar{x})$ contains a negated relational atom. Then $Q_i(\bar{x})$ is equal to $\exists \bar{u}(T_{p_1}(\bar{u}_1) \wedge \cdots \wedge T_{p_n}(\bar{u}_n) \wedge \neg T_{p_{n+1}}(\bar{u}_{n+1}))$, where $p_j \in [1,k]$ and every variable in $\bar{u}_j$ is mentioned in either $\bar{u}$ or $\bar{x}$, for every $j \in [1, n+1]$. In this case, program Π_Q includes the following rule:

$$U_{p_{n+1}}(\bar{u}_{n+1}, \bar{x}) \leftarrow U_{p_1}(\bar{u}_1, \bar{x}), \ldots, U_{p_n}(\bar{u}_n, \bar{x}). \quad (16)$$

  This rule is well defined since $\exists \bar{u}(T_{p_1}(\bar{u}_1) \wedge \cdots \wedge T_{p_n}(\bar{u}_n) \wedge \neg T_{p_{n+1}}(\bar{u}_{n+1}))$ is a safe query. Finally, assume that $Q_i(\bar{x})$ contains an inequality. Then $Q_i(\bar{x})$ is equal to $\exists \bar{u}(T_{p_1}(\bar{u}_1) \wedge \cdots \wedge T_{p_n}(\bar{u}_n) \wedge v_1 \neq v_2)$, where $p_j \in [1,k]$ and every variable in

$\bar{u}_j$ is mentioned in either $\bar{u}$ or $\bar{x}$, for every $j \in [1, n]$, and v_1, v_2 are mentioned in $\bar{u}$ or $\bar{x}$. In this case, program Π_Q includes the following rules:

$$\text{EQUAL}(v_1, v_2, \bar{x}) \leftarrow U_{p_1}(\bar{u}_1, \bar{x}), \ldots, U_{p_n}(\bar{u}_n, \bar{x}), \tag{17a}$$

$$\text{ANSWER}(\bar{x}) \leftarrow \text{EQUAL}(u, v, \bar{x}), \mathbf{C}(u), \mathbf{C}(v), u \neq v, \mathbf{C}(x_1), \ldots, \mathbf{C}(x_m). \tag{17b}$$

We note that the first rule above is well defined since $\exists \bar{u}(T_{p_1}(\bar{u}_1) \wedge \cdots \wedge T_{p_n}(\bar{u}_n) \wedge v_1 \neq v_2)$ is a safe query.

Let $\bar{a}$ be a tuple of elements from the domain of a source instance I. Each predicate U_i in Π_Q is used as a copy of T_i but with m extra arguments that store tuple $\bar{a}$. These predicates are used when testing whether $\bar{a}$ is a certain answer for Q over I. More specifically, the rules of Π_Q try to construct from $\text{CAN}(I)$ a solution J for I such that $\bar{a} \notin Q(J)$. Thus, if in a solution J for I, it holds that $\bar{a} \in Q(J)$ because $\bar{a} \in Q_i(J)$, where $Q_i(\bar{x})$ is equal to $\exists \bar{u}(T_{p_1}(\bar{u}_1) \wedge \cdots \wedge T_{p_n}(\bar{u}_n) \wedge \neg T_{p_{n+1}}(\bar{u}_{n+1}))$, then Π_Q uses rule (16) to create a new solution where the negative atom of Q_i does not hold. In the same way, if in a solution J for I, it holds that $\bar{a} \in Q(J)$ because $\bar{a} \in Q_i(J)$, where $Q_i(\bar{x})$ is equal to $\exists \bar{u}(T_{p_1}(\bar{u}_1) \wedge \cdots \wedge T_{p_n}(\bar{u}_n) \wedge v_1 \neq v_2)$, then Π_Q uses rule (17a) to create a new solution where the values assigned to v_1 and v_2 are equal (predicate EQUAL is used to store this fact). If v_1 or v_2 is assigned a null value, then it is possible to create a solution where the values assigned to these variables are the same. But this is not possible if both v_1 and v_2 are assigned constant values. In fact, it follows from [9] that this implies that it is not possible to find a solution J' for I where $\bar{a} \notin Q(J')$, and in this case rule (17b) is used to indicate that $\bar{a}$ is a certain answer for Q over I.

By using the above observations, it is not difficult to prove that for every data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ and for every instance I of $\mathbf{S}$, it is the case that $\text{certain}_{\mathcal{M}}(Q, I) = \text{certain}_{\mathcal{M}}(\Pi_Q, I)$. This concludes the proof of the theorem. □

At this point, a natural question about $\text{DATALOG}^{\mathbf{C}(\neq)}$ programs is whether the different components of this language are really needed, that is, whether inequalities and recursion are essential for this language. Next, we show that this is indeed the case and, in particular, we conclude that both inequalities and recursion are essential for Theorem 4.1.

It was shown in [9] that there exist a data exchange setting $\mathcal{M}$ and a conjunctive query Q with one inequality for which there is no first-order query $Q^\star$ such that $\text{certain}_{\mathcal{M}}(Q, I) = Q^\star(\text{CAN}(I))$ holds, for every source instance I. Thus, given that a non-recursive $\text{DATALOG}^{\mathbf{C}(\neq)}$ program is equivalent to a first-order query, we conclude from Proposition 3.3 that recursion is necessary for capturing the class of unions of conjunctive queries with at most one negated atom per disjunct.

Proposition 4.3 ([9]) *There exist a data exchange setting $\mathcal{M}$ and a Boolean conjunctive query Q with a single inequality such that for every non-recursive* $\text{DATALOG}^{\mathbf{C}(\neq)}$ *program Π, it holds that $\text{certain}_{\mathcal{M}}(Q, I) \neq \text{certain}_{\mathcal{M}}(\Pi, I)$ for some source instance I.*

In the following proposition, we show that the use of inequalities is also necessary for capturing the class of unions of conjunctive queries with at most one negated atom per disjunct. We note that this cannot be obtained from the result in [9] mentioned above, as there are DATALOG$^{\mathbf{C}(\neq)}$ programs without inequalities that are not expressible in first-order logic. The proof of this proposition follows from the fact that DATALOG$^{\mathbf{C}(\neq)}$ programs without inequalities are preserved under homomorphisms, while conjunctive queries with inequalities are only preserved under one-to-one homomorphisms.

Proposition 4.4 *There exist a data exchange setting $\mathcal{M}$ and a Boolean conjunctive query Q with a single inequality such that for every* DATALOG$^{\mathbf{C}(\neq)}$ *program Π without inequalities*, $\text{certain}_{\mathcal{M}}(Q, I) \neq \text{certain}_{\mathcal{M}}(\Pi, I)$ *for some source instance I.*

Proof Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a data exchange setting defined as follows:

- The source schema $\mathbf{S}$ consists of one binary relation symbol M, and the target schema consists of one binary relation symbol N; and
- the set Σ_{st} of source-to-target dependencies consists only of the single std $M(x, y) \rightarrow N(x, y)$.

Moreover, let Q be the query $\exists x \exists y (N(x, y) \wedge x \neq y)$. We show that for every DATALOG$^{\mathbf{C}(\neq)}$ program Π without inequalities, $\text{certain}_{\mathcal{M}}(Q, I) \neq \text{certain}_{\mathcal{M}}(\Pi, I)$ for some instance I of $\mathbf{S}$.

For the sake of contradiction, assume that there exists a DATALOG$^{\mathbf{C}(\neq)}$ program Π_0 without inequalities such that for every source instance I, $\text{certain}_{\mathcal{M}}(Q, I) = \text{certain}_{\mathcal{M}}(\Pi_0, I)$ holds, and let $I_1 = \{M(a, b)\}$ and $I_2 = \{M(c, c)\}$. It is not hard to see that $\text{certain}_{\mathcal{M}}(Q, I_1) = \texttt{true}$ and $\text{certain}_{\mathcal{M}}(Q, I_2) = \texttt{false}$.

Let $J_1 = \{N(a, b)\}$ and $J_2 = \{N(c, c)\}$ be target instances. It is easy to see that J_1 and J_2 are universal solutions for I_1 and I_2, respectively. Given that $\text{certain}_{\mathcal{M}}(Q, I_1) = \texttt{true}$, we have that $\Pi_0(J_1) = \texttt{true}$. Let h be a function from $\text{dom}(J_1)$ to $\text{dom}(J_2)$ defined as $h(a) = h(b) = c$. Since Π_0 is a DATALOG$^{\mathbf{C}(\neq)}$ program without inequalities, it must be preserved under h (because h maps constants to constants, and maps the pair $(a, b) \in N^{J_1}$ into the pair $(h(a), h(b)) = (c, c) \in N^{J_2}$). We conclude that $\Pi_0(J_2) = \texttt{true}$. Hence, given that J_2 is a universal solution for I_2, we conclude from Proposition 3.3 that $\text{certain}_{\mathcal{M}}(\Pi_0, I_2) = \texttt{true}$. But we assume that $\text{certain}_{\mathcal{M}}(Q, I_2) = \text{certain}_{\mathcal{M}}(\Pi_0, I_2)$ and, therefore, we obtain a contradiction since $\text{certain}_{\mathcal{M}}(Q, I_2) = \texttt{false}$. □

Notice that as a corollary of Proposition 4.4 and Theorem 4.1, we obtain that DATALOG$^{\mathbf{C}(\neq)}$ programs are strictly more expressive than DATALOG$^{\mathbf{C}(\neq)}$ programs without inequalities.

We conclude this section by studying the complexity of the problem of computing certain answers to DATALOG$^{\mathbf{C}(\neq)}$ programs. It was shown in Proposition 3.3 that the certain answers of a DATALOG$^{\mathbf{C}(\neq)}$ program Π can be computed by directly posing Π over CAN(I). This implies that for each data exchange setting $\mathcal{M}$, the problem CERTAIN-ANSWERS$(\mathcal{M}, \Pi)$ can be solved in polynomial time if Π is a DATALOG$^{\mathbf{C}(\neq)}$ program (since CAN(I) can be computed in polynomial time and Π has polynomial time data complexity).

Proposition 4.5 *The problem* CERTAIN-ANSWERS($\mathcal{M}$, Π) *can be solved in polynomial time, for every data exchange setting* $\mathcal{M}$ *and* DATALOG$^{\mathbf{C}(\neq)}$ *program* Π.

From the previous proposition and Theorem 4.1, we conclude that the certain answers to a union of conjunctive queries with at most one negated atom per disjunct can also be computed in polynomial time. We note that this slightly generalizes one of the polynomial time results in [9], which is stated for the class of unions of conjunctive queries with at most one inequality per disjunct. The proof of the result in [9] uses different techniques, based on the chase procedure. In Sect. 5, we show that DATALOG$^{\mathbf{C}(\neq)}$ programs can also be used to express (some) unions of conjunctive queries with two inequalities per disjunct.

A natural question at this point is whether the problem CERTAIN-ANSWERS($\mathcal{M}$, Π) is PTIME-complete for some data exchange setting $\mathcal{M}$ and DATALOG$^{\mathbf{C}(\neq)}$ program Π. It is easy to see that this is the case given that the data complexity of the evaluation problem for DATALOG programs is PTIME-complete. But more interestingly, from Theorem 4.1 we have that this result is also a corollary of a stronger result for UCQ$^{\neq}$ queries, namely that there exist a data exchange setting $\mathcal{M}$ and a conjunctive query Q with one inequality such that the problem CERTAIN-ANSWERS($\mathcal{M}$, Q) is PTIME-complete.

Proposition 4.6 *There exist a* LAV *data exchange setting* $\mathcal{M}$ *and a Boolean conjunctive query* Q *with one inequality such that* CERTAIN-ANSWERS($\mathcal{M}$, Q) *is* PTIME*-complete, under* LOGSPACE *reductions.*

Proof Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a data exchange setting defined as follows. Source schema $\mathbf{S}$ consists of a unary relation V, a binary relation S, and a 4-ary relation P. Target schema $\mathbf{T}$ consists of a binary relation T and a 4-ary relation R. Set Σ_{st} consists of the following source-to-target dependencies:

$$V(x) \rightarrow \exists y T(x, y), \tag{18}$$

$$S(x, y) \rightarrow T(x, y), \tag{19}$$

$$P(x, y, w, z) \rightarrow R(x, y, w, z). \tag{20}$$

Furthermore, Boolean query Q over $\mathbf{T}$ is defined as:

$$\exists x \exists y \exists w \exists z \exists x' (R(x, y, w, z) \wedge T(x, x') \wedge T(y, y) \wedge T(w, w) \wedge T(z, z) \wedge x \neq x').$$

Next we show that CERTAIN-ANSWERS($\mathcal{M}$, Q) is PTIME-complete under LOGSPACE reductions.

Membership of CERTAIN-ANSWERS($\mathcal{M}$, Q) in PTIME follows from [9]. PTIME-hardness is established from a LOGSPACE reduction from Horn-3CNF to the complement of CERTAIN-ANSWERS($\mathcal{M}$, Q), where Horn-3CNF is the satisfiability problem for propositional formulas in CNF with at most 3 literals per clause, and with at most one positive literal per clause. This problem is known to be PTIME-complete (see, e.g., [12]). More precisely, for every Horn-3CNF formula ϕ, we construct

in logarithmic space an instance I_ϕ of $\mathbf{S}$ such that ϕ is satisfiable if and only if $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$.

Without loss of generality, assume that formula $\phi = C_1 \wedge \cdots \wedge C_k$, where each C_i ($i \in \{1, \ldots, k\}$) is a clause of the form either $p \vee \neg q \vee \neg r$ or p or $\neg p \vee \neg q \vee \neg r$, being p, q and r arbitrary propositional variables. Then instance I_ϕ is defined as follows:

- The interpretation of unary relation V in I_ϕ is the set of propositional variables mentioned in ϕ.
- The interpretation of binary relation S in I_ϕ is the set of tuples $\{(\texttt{b}, \texttt{b}), (\texttt{h}, \texttt{f})\}$, where b, h and f are fresh constants (not mentioned as propositional variables in ϕ).
- For every clause C_i in I_ϕ ($i \in \{1, \ldots, k\}$), the interpretation of 4-ary relation P in I_ϕ contains the following tuple:
  - $(p, q, r, \texttt{b})$ if $C_i = p \vee \neg q \vee \neg r$,
  - $(p, \texttt{b}, \texttt{b}, \texttt{b})$ if $C_i = p$, and
  - $(\texttt{h}, p, q, r)$ if $C_i = \neg p \vee \neg q \vee \neg r$.

Clearly, I_ϕ can be constructed in logarithmic space from ϕ.

Next, we show that $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$ if and only if ϕ is satisfiable.

($\Rightarrow$) Assume first that $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$.

In the setting $\mathcal{M}$, the canonical universal solution $\text{CAN}(I_\phi)$ for I_ϕ is as follows. Assume that $\perp_q$ is the null generated by applying rule (19) to each atom $V(q)$ in I_ϕ. Then the interpretation of R in $\text{CAN}(I)$ is equal to the interpretation of P in I, and the interpretation of T in $\text{CAN}(I)$ contains tuples $(\texttt{b}, \texttt{b})$, $(\texttt{h}, \texttt{f})$ and $(q, \perp_q)$ for every propositional variable q mentioned in ϕ.

Given that $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$, there exists a solution J for I such that $Q(J) = \texttt{false}$. Let $h : \text{CAN}(I) \to J$ be an homomorphism from $\text{CAN}(I)$ into J, and let σ be the following truth assignment for the propositional variables mentioned in ϕ: $\sigma(q) = 1$ iff $h(\perp_q) = q$. Next we show that σ satisfies ϕ. More precisely, we prove that $\sigma(C_i) = 1$, for every $i \in \{1, \ldots, k\}$. We consider three cases:

- Assume that $C_i = p$. Since $R(p, \texttt{b}, \texttt{b}, \texttt{b})$ belongs to J, and also $T(\texttt{b}, \texttt{b})$ belongs to J, it must be the case that $h(\perp_p) = p$ since $Q(J) = \texttt{false}$ and $(p, \perp_p)$ belongs to the interpretation of T in $\text{CAN}(I)$. We conclude that $\sigma(p) = 1$ and, hence, $\sigma(C_i) = 1$.
- Assume that $C_i = p \vee \neg q \vee \neg r$ and $\sigma(q) = \sigma(r) = 1$. Then by definition of h, we have that $h(\perp_q) = q$ and $h(\perp_r) = r$ and, therefore, (q, q) and (r, r) belong to the interpretation of T in J. Thus, given that $R(p, q, r, \texttt{b})$ and $T(\texttt{b}, \texttt{b})$ belong to J, it must be the case that $h(\perp_p) = p$ since $Q(J) = \texttt{false}$ and $(p, \perp_p)$ belongs to the interpretation of T in $\text{CAN}(I)$. We conclude that $\sigma(p) = 1$ and, hence, $\sigma(C_i) = 1$.
- Assume that $C_i = \neg p \vee \neg q \vee \neg r$. For the sake of contradiction, assume that $\sigma(p) = \sigma(q) = \sigma(r) = 1$. Then by definition of h, we have that $h(\perp_p) = p$, $h(\perp_q) = q$ and $h(\perp_r) = r$ and, therefore, (p, p), (q, q) and (r, r) belong to the interpretation of T in J. Thus, given that $R(\texttt{h}, p, q, r)$, $T(\texttt{h}, \texttt{f})$ belong to J and $\texttt{h} \neq \texttt{f}$ holds, we conclude that $Q(J) = \texttt{true}$, which contradicts our initial assumption. We conclude that $\sigma(p) = 0$ or $\sigma(q) = 0$ or $\sigma(r) = 0$, which implies that $\sigma(C_i) = 1$.

($\Leftarrow$) Assume that ϕ is satisfiable, and let σ be a truth assignment for the propositional variables in ϕ such that $\sigma(\phi) = 1$. Furthermore, assume that $\textsc{Can}(I_\phi)$ is constructed as above. From σ, define a function f from $\mathrm{dom}(\textsc{Can}(I_\phi))$ into $\mathrm{dom}(\textsc{Can}(I_\phi))$ as follows:

$$f(v) = \begin{cases} q & v = \bot_q \text{ and } \sigma(q) = 1, \\ v & \text{otherwise.} \end{cases}$$

Let $J^\star$ be a solution for I_ϕ under $\mathcal{M}$ obtained from $\textsc{Can}(I_\phi)$ by replacing each occurrence of an element v with $f(v)$. Next we show that $Q(J^\star) = \texttt{false}$ and, thus, we conclude that $\mathrm{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$.

Assume, for the sake of contradiction, that $Q(J^\star) = \texttt{true}$. Then, there exists a function $h : \{x, y, w, z, x'\} \to \mathrm{dom}(J^\star)$ such that $R(h(x), h(y), h(w), h(z))$, $T(h(x), h(x'))$, $T(h(y), h(y))$, $T(h(w), h(w))$ and $T(h(z), h(z))$ are all tuples in $J^\star$, and $h(x) \neq h(x')$. To prove that this leads to a contradiction, we consider three cases.

- Assume that $h(x) = p$, where p is a propositional variable, and $h(y) = h(w) = h(z) = \mathtt{b}$. Then by definition of $\mathcal{M}$ and I_ϕ, we have that p is a clause in ϕ. But given that $h(x) = p$, $h(x) \neq h(x')$ and $T(h(x), h(x'))$ is a tuple in $J^\star$, it is the case that $h(x') = \bot_p$. Thus, given that $\bot_p$ is an element of $J^\star$, it holds that $\sigma(p) = 0$ since $f(\bot_p) = \bot_p$. We conclude that $\sigma(\phi) = 0$ since $\sigma(p) = 0$, which contradicts our initial assumption.
- Assume that $h(x) = p$, $h(y) = q$ and $h(w) = r$, where p, q and r are propositional variables, and $h(z) = \mathtt{b}$. Then by definition of $\mathcal{M}$ and I_ϕ, we have that $p \vee \neg q \vee \neg r$ is a clause in ϕ. But given that $h(x) = p$, $h(x) \neq h(x')$ and $T(h(x), h(x'))$ is a tuple in $J^\star$, it is the case that $h(x') = \bot_p$. Thus, given that $\bot_p$ is an element of $J^\star$, it holds that $\sigma(p) = 0$ since $f(\bot_p) = \bot_p$. Moreover, given that $T(h(y), h(y))$ and $T(h(w), h(w))$ are tuples in $J^\star$, it holds that $T(q, q)$ and $T(r, r)$ are tuples in $J^\star$. Thus, $f(\bot_q) = q$ and $f(\bot_r) = r$ and, hence $\sigma(q) = \sigma(r) = 1$. We conclude that $\sigma(\phi) = 0$ since $\sigma(p) = 0$ and $\sigma(q) = \sigma(r) = 1$, which contradicts our initial assumption.
- Assume that $h(x) = \mathtt{h}$, $h(y) = p$, $h(w) = q$ and $h(z) = r$, where p, q and r are propositional variables. Then by definition of $\mathcal{M}$ and I_ϕ, we have that $\neg p \vee \neg q \vee \neg r$ is a clause in ϕ. But given that $T(h(y), h(y))$, $T(h(w), h(w))$ and $T(h(z), h(z))$ are all tuples in $J^\star$, it holds that $T(p, p)$, $T(q, q)$ and $T(r, r)$ are all tuples in $J^\star$. Thus, $f(\bot_p) = p$, $f(\bot_q) = q$ and $f(\bot_r) = r$ and, hence $\sigma(p) = \sigma(q) = \sigma(r) = 1$. We conclude that $\sigma(\phi) = 0$ since $\sigma(p) = \sigma(q) = \sigma(r) = 1$, which contradicts our initial assumption.

This concludes the proof of the proposition. □

It is worth mentioning that it follows from Proposition 3.1 in [14] that there exists a data exchange setting $\mathcal{M}$ containing some *target* dependencies and a conjunctive query Q with one inequality such that $\textsc{Certain-Answers}(\mathcal{M}, Q)$ is $\textsc{Ptime}$-complete. Proposition 4.6 shows that this result holds even when no target dependencies are provided.

5 Conjunctive Queries with Two Inequalities

As we mentioned before, computing certain answers to conjunctive queries with more than just one inequality is an intractable problem. Indeed, there is a LAV setting $\mathcal{M}$ and a Boolean conjunctive query Q with two inequalities such that the problem CERTAIN-ANSWERS($\mathcal{M}$, Q) is coNP-complete [20]. Therefore, unless PTIME = NP, Theorem 4.1 is no longer valid if we remove the restriction that every disjunct of Q must contain at most one inequality.

The intractability for conjunctive queries with two inequalities is tightly related with the use of null values when joining relations and checking inequalities. In this section, we investigate this relationship, and provide a syntactic condition on the type of joins and inequalities allowed in queries. This restriction leads to tractability of the problem of computing certain answers. Indeed, this tractability is a corollary of a stronger result, namely that for every conjunctive query Q with two inequalities, if Q satisfies the syntactic condition, then one can construct a DATALOG$^{\mathbf{C}(\neq)}$ program Π_Q such that certain$_{\mathcal{M}}(Q, I) =$ certain$_{\mathcal{M}}(\Pi_Q, I)$ for every source instance I. It should be noticed that in this case DATALOG$^{\mathbf{C}(\neq)}$ programs are used as a tool for finding a tractable class of queries for the problem of computing certain answers.

To define the syntactic restriction mentioned above, we need to introduce some terminology. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a data exchange setting. Then for every n-ary relation symbol T in $\mathbf{T}$, we say that the i-th attribute of T $(1 \leq i \leq n)$ *can be nullified* under $\mathcal{M}$, if there is an st-tgd α in Σ_{st} such that the i-th attribute of T is existentially quantified in the right hand side of α. Notice that for each setting $\mathcal{M}$ and source instance I, if the i-th attribute of T cannot be nullified under $\mathcal{M}$, then for every tuple $(c_1, \ldots, c_n)$ that belongs to T in the canonical universal solution for I, it holds that c_i is a constant. Moreover, if Q is a UCQ$^{\neq}$ query over $\mathbf{T}$ and x is a variable in Q, then we say that x *can be nullified* under Q and $\mathcal{M}$, if x appears in Q as the i-th attribute of a target relation T, and the i-th attribute of T can be nullified under $\mathcal{M}$.

Let $\mathcal{M}$ be a data exchange setting and Q a conjunctive query with two inequalities, and assume that if x appears as a variable in the inequalities of Q, then x cannot be nullified under Q and $\mathcal{M}$. In this case, it is straightforward to prove that CERTAIN-ANSWERS($\mathcal{M}$, Q) is tractable. Indeed, the previous condition implies that for every source instance I, if Q holds in CAN(I), then all the witnesses for Q in CAN(I) make comparisons of the form $c \neq c'$, where c and c' are constants. Thus, we have that certain$_{\mathcal{M}}(Q, I)$ can be computed by simply evaluating Q over CAN(I). Here we are interested in finding less obvious conditions that lead to tractability. In particular, we would like to find queries that do not restrict the use of null values in such a strict way.

Let Q be a conjunctive query with two inequalities over a target schema $\mathbf{T}$. Assume that the quantifier free part of Q is of the form $\phi(x_1, \ldots, x_m) \wedge u_1 \neq v_1 \wedge u_2 \neq v_2$, where ϕ is a conjunction of relational atoms over $\mathbf{T}$ and u_1, v_1, u_2 and v_2 are all mentioned in the set of variables $x_1, \ldots, x_m$ (Q is a safe query [2]). We are now ready to define the two components of the syntactic restriction that leads to tractability of the problem of computing certain answers. We say that Q has *almost constant inequalities* under $\mathcal{M}$, if u_1 or v_1 cannot be nullified under Q and $\mathcal{M}$, and u_2 or v_2 cannot be nullified under Q and $\mathcal{M}$. Intuitively, this means that to satisfy Q in the

canonical universal solution of a source instance, one can only make comparisons of the form $c \neq \perp$ and $c \neq c'$, where c, c' are constants and $\perp$ is a null value. Moreover, we say that Q has *constant joins* under $\mathcal{M}$, if for every variable x that appears at least twice in ϕ, x cannot be nullified under Q and $\mathcal{M}$. Intuitively, this means that to satisfy Q in the canonical universal solution of a source instance, one can only use constant values when joining relations.

Example 5.1 Let $\mathcal{M}$ be a data exchange setting specified by st-tgds:

$$\begin{aligned} P(x, y) &\rightarrow T(x, y), \\ P(x, y) &\rightarrow \exists z U(x, z). \end{aligned}$$

The first and second attribute of T, as well as the first attribute of U, cannot be nullified under $\mathcal{M}$. On the other hand, the second attribute of U can be nullified under $\mathcal{M}$.

Let $Q(x)$ be query $\exists y \exists z (T(y, x) \wedge U(z, x) \wedge x \neq y \wedge x \neq z)$. Then we have that Q has almost constant inequalities under $\mathcal{M}$ because variables y and z cannot be nullified under Q and $\mathcal{M}$, but Q does not have constant joins because variable x appears twice in $T(y, x) \wedge U(z, x)$ and it can be nullified under Q and $\mathcal{M}$. On the other hand, query $U(x, y) \wedge U(x, z) \wedge x \neq z \wedge y \neq z$ has constant joins but does not have almost constant inequalities, and query $U(x, y) \wedge T(x, z) \wedge x \neq z \wedge y \neq z$ has both constant joins and almost constant inequalities.

Although the notions of constant joins and almost constant inequalities were defined for $\mathrm{CQ}^{\neq}$ queries with two inequalities, they can be easily extended to the case of conjunctive queries with an arbitrary number of inequalities. In fact, the notion of constant joins does not change in the case of an arbitrary number of inequalities, while to define the notion of almost constant inequalities in the general case, one has to say that each inequality $x \neq y$ in a query satisfies the condition that x or y cannot be nullified. With this extension, we have all the necessary ingredients for the main result of this section.

Theorem 5.2 *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a data exchange setting and Q a $\mathrm{UCQ}^{\neq}$ query over $\mathbf{T}$ such that each disjunct of Q either* (1) *has at most one inequality and constant joins under $\mathcal{M}$, or* (2) *has two inequalities, constant joins and almost constant inequalities under $\mathcal{M}$. Then there exists a $\mathrm{DATALOG}^{\mathbf{C}(\neq)}$ program Π_Q over $\mathbf{T}$ such that for every instance I of $\mathbf{S}$, $\mathrm{certain}_{\mathcal{M}}(Q, I) = \mathrm{certain}_{\mathcal{M}}(\Pi_Q, I)$. Moreover, Π_Q can be effectively constructed from Q and $\mathcal{M}$ in polynomial time.*

Proof Let $\mathcal{M}$ and Q be as in the statement of the theorem. Assume that $Q(\bar{x})$ is of the form $Q_1(\bar{x}) \vee \cdots \vee Q_\ell(\bar{x})$, where $\bar{x} = (x_1, \ldots, x_m)$, $m \geq 0$. In order to prove the theorem we need to introduce some extra terminology and an intermediate result (Lemma 5.3).

Let I be an arbitrary source instance. In what follows, we use J instead of $\mathrm{CAN}(I)$ to denote the canonical universal solution for I under $\mathcal{M}$. Let then $\bar{t} = (t_1, \ldots, t_m)$ be a tuple of constants from I that also belong to J.

Although the proof of this theorem is rather long and technical, the intuitive idea that underlies it is simple to explain. Our goal is to construct an *implication* graph $H(Q, J, \bar{t})$—as in the standard algorithms for 2SAT—such that $\bar{t} \in \text{certain}_{\mathcal{M}}(Q, I)$ iff $H(Q, J, \bar{t})$ contains a *conflict*. We now show how to construct the graph $H(Q, J, \bar{t})$ from Q, J and $\bar{t}$. The set of nodes of $H(Q, J, \bar{t})$ consists of all pairs of distinct elements of J plus two fresh elements μ and ν that do not appear in J. The edges of $H(Q, J, \bar{t})$ are labeled over the alphabet {`blue`, `red`, `green`} as follows:

- There is an edge labeled `red` between two nodes in $H(Q, J, \bar{t})$ iff these two nodes share a null value;
- there is an edge labeled `blue` between nodes μ and ν in $H(Q, J, \bar{t})$ iff for some $1 \leq i \leq \ell$, $Q_i(\bar{x})$ is of the form $\exists \bar{y} \phi(\bar{x}, \bar{y})$, where $\phi(\bar{x}, \bar{y})$ is a conjunction of relational atoms over $\mathbf{T}$, and $J \models Q_i(\bar{t})$;
- there is an edge labeled `blue` between nodes (p_1, p_2) and (p_3, p_4) in $H(Q, J, \bar{t})$ iff for some $1 \leq i \leq \ell$, (1) $Q_i(\bar{x})$ is of the form $\exists \bar{y}(\phi(\bar{x}, \bar{y}) \wedge u_1 \neq u_2 \wedge v_1 \neq v_2)$, where $\phi(\bar{x}, \bar{y})$ is a conjunction of relational atoms over $\mathbf{T}$ and $u_1, u_2, v_1, v_2 \in \{\bar{x}, \bar{y}\}$, and (2) there is an assignment $\sigma : \{\bar{x}, \bar{y}\} \to \text{dom}(J)$, such that $\sigma(\bar{x}) = \bar{t}$, $(J, \sigma) \models \phi(\bar{x}, \bar{y}) \wedge u_1 \neq u_2 \wedge v_1 \neq v_2$, $\sigma(u_1) = p_1$, $\sigma(u_2) = p_2$, $\sigma(v_1) = p_3$, and $\sigma(v_2) = p_4$; and
- there is a loop labeled `green` on the node (p_1, p_2) iff for some $1 \leq i \leq \ell$, (1) $Q_i(\bar{x})$ is of the form $\exists \bar{y}(\phi(\bar{x}, \bar{y}) \wedge u_1 \neq u_2)$, where $\phi(\bar{x}, \bar{y})$ is a conjunction of relational atoms over $\mathbf{T}$ and $u_1, u_2 \in \{\bar{x}, \bar{y}\}$, and (2) there is an assignment $\sigma : \{\bar{x}, \bar{y}\} \to \text{dom}(J)$, such that $\sigma(\bar{x}) = \bar{t}$, $(J, \sigma) \models \phi(\bar{x}, \bar{y}) \wedge u_1 \neq u_2$, $\sigma(u_1) = p_1$, $\sigma(u_2) = p_2$.

A node q in $H(Q, J, \bar{t})$ is *open* if both of its components are nulls, and it is *semi-open* if one of its components is a constant and the other one is a null. The node q is *openly-reachable* from a node q' if there is a path $q' q_1 \cdots q_k q$ in $H(Q, J, \bar{t})$, $k \geq 0$, such that:

- Every node q_i is `red`-adjacent to q_{i+1}, $1 \leq i < k$, q' is `red`-adjacent to q_1 and q_k is `red`-adjacent to q;
- every node q_i, $1 \leq i \leq k$, is open; and
- every node q_i, $1 \leq i \leq k$, has a `green`-labeled loop.

Finally, we say that q has a *contradiction* path (c-path) in $H(Q, J, \bar{t})$ if either the components of q are distinct constants or there is a path $q = q_1 q_2 \cdots q_{2k+1}$ in $H(Q, J, \bar{t})$, $k \geq 0$, that satisfies the following:

- Every node q_i, $1 \leq i \leq 2k$, is semi-open;
- every node of the form q_{2i+2}, $0 \leq i \leq k-1$, is openly-reachable from q_{2i+1}, but the constant components in q_{2i+1} and q_{2i+2} are different;
- every node of the form q_{2i}, $0 < i \leq k$, is either `blue`-adjacent to q_{2i+1} or $q_{2i} = q_{2i+1}$ and q_{2i} has a `green`-labeled loop; and
- either q_{2k} has a `green`-labeled loop, or q_{2k+1} has two different constant components, or for some $1 \leq i \leq k$ it is the case that the node q_{2k+1} is openly-reachable from q_{2i-1} and the constant components of q_{2i-1} and q_{2k+1} are different.

The following lemma is a key component in the proof of the theorem. It confirms our intuitive idea that $H(Q, J, \bar{t})$ is an *implication* graph, over which one can check whether $\bar{t} \in \text{certain}_{\mathcal{M}}(Q, I)$ by simply looking for *conflicts*. Those conflicts can be identified by detecting the presence of specific paths in the graph. The proof of this lemma is rather technical and left to Appendix A.1.

Lemma 5.3 *Let Q and $\mathcal{M}$ be as defined above. For every source instance I with canonical universal solution J, and tuple $\bar{t}$ of constants from J, it is the case that $\bar{t} \in \text{certain}_{\mathcal{M}}(Q, I)$ iff μ and ν are* `blue`*-adjacent in $H(Q, J, \bar{t})$, or there are two nodes q and q' in $H(Q, J, \bar{t})$ such that q and q' are* `blue`*-adjacent and both q and q' have c-paths in $H(Q, J, \bar{t})$, or there is a node q in $H(Q, J, \bar{t})$ that has a* `green`*-labeled loop and a c-path in $H(Q, J, \bar{t})$.*

Before we continue with the proof of Theorem 5.2, we sketch the proof by means of an example. In what follows, given a data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ and a conjunctive query Q with two inequalities that satisfies the restrictions of Theorem 5.2, we construct a $\text{DATALOG}^{\mathbf{C}(\neq)}$ program Π_Q such that, when evaluated over the canonical universal solution J for some instance I, it computes all the tuples $\bar{t}$ for which the graph $H(Q, J, \bar{t})$ satisfies the conditions stated in Lemma 5.3.

Example 5.4 Let $\mathcal{M}$ be a data exchange setting such that $\mathbf{S} = \{D(\cdot, \cdot, \cdot), E(\cdot, \cdot, \cdot)\}$, $\mathbf{T} = \{P(\cdot, \cdot, \cdot), R(\cdot, \cdot, \cdot)\}$ and

$$\Sigma_{st} = \{D(x, y, z) \rightarrow \exists n(P(x, y, n) \wedge P(x, z, n)),$$
$$E(x, y, z) \rightarrow \exists n(R(x, y, n) \wedge R(x, z, n))\}.$$

Also, let $Q(x)$ be the following conjunctive query with two inequalities, constant joins and almost constant inequalities:

$$\exists y \exists z \exists w(P(x, y, z) \wedge R(x, y, w) \wedge y \neq z \wedge y \neq w).$$

We construct a $\text{DATALOG}^{\mathbf{C}(\neq)}$ program Π_Q such that $\text{certain}_{\mathcal{M}}(Q, I) = \text{certain}_{\mathcal{M}}(\Pi_Q, I)$, for every source instance I. The set of intensional predicates of Π_Q is $\{U_1(\cdot, \cdot, \cdot, \cdot), \{U_2(\cdot, \cdot, \cdot, \cdot), \text{DOM}(\cdot), \text{EQUAL}^{\mathbf{left},1}(\cdot, \cdot, \cdot, \cdot, \cdot, \cdot), \text{EQUAL}^{\mathbf{right},1}(\cdot, \cdot, \cdot, \cdot, \cdot, \cdot), \text{ANSWER}^{\mathbf{left},1}(\cdot, \cdot, \cdot, \cdot), \text{ANSWER}^{\mathbf{right},1}(\cdot, \cdot, \cdot, \cdot), \text{ANSWER}(\cdot)\}$. The program Π_Q over $\mathbf{T}$ is defined as follows.

- First, the program collects in $\text{dom}(x)$ all the elements that belong to the active domain of the instance of $\mathbf{T}$ where Π_Q is evaluated:

$$\text{DOM}(z) \leftarrow P(x, y, z), \quad (1)$$
$$\text{DOM}(z) \leftarrow P(x, z, y), \quad (2)$$
$$\text{DOM}(z) \leftarrow P(z, x, y), \quad (3)$$
$$\text{DOM}(z) \leftarrow R(x, y, z), \quad (4)$$
$$\text{DOM}(z) \leftarrow R(x, z, y), \quad (5)$$
$$\text{DOM}(z) \leftarrow R(z, x, y). \quad (6)$$

- We also add a rule that creates in U_1 and U_2 a copy of P and R respectively, but with an extra argument for keeping track of the element where Π_Q is being evaluated:

$$U_1(x, y, z, t) \leftarrow P(x, y, z), \textsc{Dom}(t), \tag{7}$$
$$U_2(x, y, z, t) \leftarrow R(x, y, z), \textsc{Dom}(t). \tag{8}$$

- If an element a does not belong to the set of certain answers to $Q(x)$, then for every tuple of the form (a, b, c, d) such that (a, b, c) belongs to the interpretation of P and (a, b, d) belongs to the interpretation of R, it is the case that $b = c$ or $b = d$. This is expressed by means of the following rules:

$$\textsc{Equal}^{\mathbf{left}}(y, z, y, z, w, x) \leftarrow U_1(x, y, z, x), U_2(x, y, w, x), \tag{9}$$
$$\textsc{Equal}^{\mathbf{right}}(y, w, y, z, w, x) \leftarrow U_1(x, y, z, x), U_2(x, y, w, x). \tag{10}$$

  Intuitively, predicate $\textsc{Equal}^{\mathbf{left}}$ (resp., $\textsc{Equal}^{\mathbf{right}}$) keeps track that it is the first (resp., second) inequality of Q that is falsified. For reasons that will become clear later, predicates $\textsc{Equal}^{\mathbf{left}}$ and $\textsc{Equal}^{\mathbf{right}}$ not only need to keep track of the element where the query is being evaluated, but also of the elements that witness the existential quantifiers of the query. This is handled by means of the last four arguments of the predicates $\textsc{Equal}^{\mathbf{left}}$ and $\textsc{Equal}^{\mathbf{right}}$.
- Since $\textsc{Equal}^{\mathbf{left}}(x, y, \cdot, \cdot, \cdot, \cdot)$ (resp. $\textsc{Equal}^{\mathbf{left}}(x, y, \cdot, \cdot, \cdot, \cdot)$) holds if x and y are the same elements, the program Π_Q must include the following rules,

$$\textsc{Equal}^{\mathbf{left}}(v, v, x, y, z, t) \leftarrow \textsc{Dom}(v), \textsc{Dom}(x), \textsc{Dom}(y), \textsc{Dom}(z), \textsc{Dom}(t), \tag{11}$$
$$\textsc{Equal}^{\mathbf{left}}(u, v, x, y, z, t) \leftarrow \textsc{Equal}^{\mathbf{left}}(v, u, x, y, z, t), \tag{12}$$
$$\textsc{Equal}^{\mathbf{left}}(v, u, x, y, z, t) \leftarrow \textsc{Equal}^{\mathbf{left}}(v, w, x, y, z, t), \textsc{Equal}^{\mathbf{left}}(w, u, x, y, z, t), \tag{13}$$
$$\textsc{Equal}^{\mathbf{right}}(v, v, x, y, z, t) \leftarrow \textsc{Dom}(v), \textsc{Dom}(x), \textsc{Dom}(y), \textsc{Dom}(z), \textsc{Dom}(t), \tag{14}$$
$$\textsc{Equal}^{\mathbf{right}}(u, v, x, y, z, t) \leftarrow \textsc{Equal}^{\mathbf{right}}(v, u, x, y, z, t), \tag{15}$$
$$\textsc{Equal}^{\mathbf{right}}(v, u, x, y, z, t) \leftarrow \textsc{Equal}^{\mathbf{right}}(v, w, x, y, z, t), \textsc{Equal}^{\mathbf{right}}(w, u, x, y, z, t). \tag{16}$$

- Lemma 5.3 shows that in order to check whether a belongs to the certain answers to $Q(x)$, it suffices to show that there exists a pair (q, q') of `blue`-adjacent nodes in $H(Q, J, a)$ that have c-paths. In order to guide the search for a c-path from a node that witnesses the first (i.e. **left**) inequality of Q, we use the following set of rules:

$$\textsc{Equal}^{\mathbf{left}}(y, w, t, u, v, x) \leftarrow U_1(x, y, z, x), U_2(x, y, w, x), \mathbf{C}(y), \textsc{Equal}^{\mathbf{left}}(z, s, t, u, v, x), \mathbf{C}(s), y \neq s, \tag{17}$$

$$\begin{aligned}
\textsc{Equal}^{\mathbf{left}}(y,w,t,u,v,x) \quad &\leftarrow \quad U_1(x,y,z,x), U_2(x,y,w,x), \mathbf{C}(z), \\
&\qquad \textsc{Equal}^{\mathbf{left}}(y,s,t,u,v,x), \mathbf{C}(s), z \neq s, \quad (18)\\
\textsc{Equal}^{\mathbf{left}}(y,z,t,u,v,x) \quad &\leftarrow \quad U_1(x,y,z,x), U_2(x,y,w,x), \mathbf{C}(y), \\
&\qquad \textsc{Equal}^{\mathbf{left}}(w,s,t,u,v,x), \mathbf{C}(s), y \neq s, \quad (19)\\
\textsc{Equal}^{\mathbf{left}}(y,z,t,u,v,x) \quad &\leftarrow \quad U_1(x,y,z,x), U_2(x,y,w,x), \mathbf{C}(w), \\
&\qquad \textsc{Equal}^{\mathbf{left}}(y,s,t,u,v,x), \mathbf{C}(s), w \neq s. \quad (20)
\end{aligned}$$

- Intuitively, the first of these rules expresses the following. If an element a does not belong to the set of certain answers to $Q(x)$, then for every tuple of the form (a,b,c,d) such that (a,b,c) belongs to the interpretation of P and (a,b,d) belongs to the interpretation of R, if b is a constant and c is set to be equal to the constant e such that $b \neq e$, then it must be the case that $b = d$. The intuition behind the rest of the rules is analogous.

  Equivalently, in order to guide the search for a c-path from a node that witnesses the second (i.e. **right**) inequality of Q, we use the following set of rules:

$$\begin{aligned}
\textsc{Equal}^{\mathbf{right}}(y,w,t,u,v,x) \quad &\leftarrow \quad U_1(x,y,z,x), U_2(x,y,w,x), \mathbf{C}(y), \\
&\qquad \textsc{Equal}^{\mathbf{right}}(z,s,t,u,v,x), \mathbf{C}(s), y \neq s, \quad (21)\\
\textsc{Equal}^{\mathbf{right}}(y,w,t,u,v,x) \quad &\leftarrow \quad U_1(x,y,z,x), U_2(x,y,w,x), \mathbf{C}(z), \\
&\qquad \textsc{Equal}^{\mathbf{right}}(y,s,t,u,v,x), \mathbf{C}(s), z \neq s, \quad (22)\\
\textsc{Equal}^{\mathbf{right}}(y,z,t,u,v,x) \quad &\leftarrow \quad U_1(x,y,z,x), U_2(x,y,w,x), \mathbf{C}(y), \\
&\qquad \textsc{Equal}^{\mathbf{right}}(w,s,t,u,v,x)), \mathbf{C}(s), y \neq s, \quad (23)\\
\textsc{Equal}^{\mathbf{right}}(y,z,t,u,v,x) \quad &\leftarrow \quad U_1(x,y,z,x), U_2(x,y,w,x), \mathbf{C}(w), \\
&\qquad \textsc{Equal}^{\mathbf{right}}(y,s,t,u,v,x), \mathbf{C}(s), w \neq s. \quad (24)
\end{aligned}$$

- The program Π_Q also includes the following rules:

$$\begin{aligned}
\textsc{Answer}^{\mathbf{left}}(y,z,w,x) \quad &\leftarrow \quad \textsc{Equal}^{\mathbf{left}}(u,v,y,z,w,x), \mathbf{C}(u), \\
&\qquad \mathbf{C}(v), u \neq v, \quad (25)\\
\textsc{Answer}^{\mathbf{right}}(y,z,w,x) \quad &\leftarrow \quad \textsc{Equal}^{\mathbf{right}}(u,v,y,z,w,x), \mathbf{C}(u), \\
&\qquad \mathbf{C}(v), u \neq v. \quad (26)
\end{aligned}$$

  Intuitively, these rules collect in $\textsc{Answer}^{\mathbf{left}}$ (resp. $\textsc{Answer}^{\mathbf{right}}$) all those nodes q that witness the first (resp. second) inequality of Q and that have a c-path.
- Finally, the program includes the following rule that collects certain answers:

$$\begin{aligned}
\textsc{Answer}(x) \quad &\leftarrow \quad \textsc{Answer}^{\mathbf{left}}(y,z,w,x), \\
&\qquad \textsc{Answer}^{\mathbf{right}}(y,z,w,x), \mathbf{C}(x). \quad (27)
\end{aligned}$$

Intuitively, this says that if there are blue-adjacent nodes q and q' in $H(Q, J, a)$ such that both q and q' have c-paths, then a belongs to the certain answers to $Q(x)$. Notice that it has been necessary to keep track until this last stage not only of the argument where the query is being evaluated, but also of the whole tuple that determines the blue-adjacency of q and q'. This is done by using the last four arguments in the predicates $\text{EQUAL}^{\textbf{left}}$ and $\text{EQUAL}^{\textbf{right}}$.

We now show the application of the program with an example. Let $I = \{D(a, a, b), D(a, c, d), E(a, b, c), E(a, b, d), E(a, a, c)\}$ be a source instance. Then the canonical universal solution J for I is as follows:

- The interpretation of the relation P in J is $\{(a, a, \perp_1), (a, b, \perp_1), (a, c, \perp_3), (a, d, \perp_3)\}$.
- The interpretation of the relation R in J is $\{(a, b, \perp_2), (a, c, \perp_2), (a, b, \perp_4), (a, d, \perp_4), (a, a, \perp_5), (a, c, \perp_5)\}$.

By applying rules (1) to (6) we first collect all the elements of J in DOM. From rules (7) and (8) we obtain that $U_1(a, a, \perp_1, a)$ and $U_2(a, a, \perp_5, a)$ hold. Then we use rule (9) to show that $\text{EQUAL}^{\textbf{left}}(a, \perp_1, a, \perp_1, \perp_5, a)$ holds. Now we use rule (12) to obtain that $\text{EQUAL}^{\textbf{left}}(\perp_1, a, a, \perp_1, \perp_5, a)$ holds.

Next, we apply rules (7) and (8) to obtain that $U_1(a, b, \perp_1, a)$ and $U_2(a, b, \perp_2, a)$ hold. Further, since $\mathbf{C}(a)$ and $\mathbf{C}(b)$ hold (a and b are constants) and $b \neq a$, we obtain from rule (17) that $\text{EQUAL}^{\textbf{left}}(b, \perp_2, a, \perp_1, \perp_5, a)$ holds, and by using rule (12) we show that $\text{EQUAL}^{\textbf{left}}(\perp_2, b, a, \perp_1, \perp_5, a)$ holds.

We use again rules (7) and (8) to obtain that $U_1(a, c, \perp_3, a)$ and $U_2(a, c, \perp_2, a)$ hold. Since c and b are different constants, we can apply rule (19) and obtain that $\text{EQUAL}^{\textbf{left}}(c, \perp_3, a, \perp_1, \perp_5, a)$ holds. We then use rule (12) to show that $\text{EQUAL}^{\textbf{left}}(\perp_3, c, a, \perp_1, \perp_5, a)$ also holds in J. Rules (7) and (8) are applied one more time to obtain that $U_1(a, d, \perp_3, a)$ and $U_2(a, d, \perp_4, a)$ hold. Then, since c and d are different constant values, we can use rule (17) for a second time to show that $\text{EQUAL}^{\textbf{left}}(d, \perp_4, a, \perp_1, \perp_5, a)$ holds, and next rule (12) is used to show that $\text{EQUAL}^{\textbf{left}}(\perp_4, d, a, \perp_1, \perp_5, a)$ holds.

Rule (8) is used for the last time to obtain that $U_2(a, b, \perp_4, a)$ holds. Further, we use rule (19) to show that $\text{EQUAL}^{\textbf{left}}(b, \perp_1, a, \perp_1, \perp_5, a)$ holds. Finally, by applying rule (13) we conclude that $\text{EQUAL}^{\textbf{left},1}(a, b, a, \perp_1, \perp_5, a)$ holds in J, and then we use rule (25) to show that $\text{ANSWER}^{\textbf{left},1}(a, \perp_1, \perp_5, a)$ belongs to J.

Using a procedure very similar to the preceding paragraphs, it can be shown that $\text{ANSWER}^{\textbf{right},1}(a, \perp_1, \perp_5, a)$ also holds in J, and then, since a is a constant, from rule (27) we obtain that a belongs to the certain answers of Q for I under $\mathcal{M}$.

We now continue with the proof of Theorem 5.2. Assume that $\mathbf{T} = \{T_1, \ldots, T_k\}$, where each T_i has arity $n_i > 0$, and that $Q(\bar{x}) = Q_1(\bar{x}) \vee \cdots \vee Q_\ell(\bar{x})$, where $\bar{x} = (x_1, \ldots, x_m)$ and each $Q_i(\bar{x})$ is either (1) a conjunctive query with at most one inequality and with constant joins, or (2) a conjunctive query with two inequalities but with constant joins and almost constant inequalities. Further, assume that $W \subseteq \{1, \ldots, \ell\}$ is the set of all indexes j such that $Q_j(\bar{x})$ contains two inequalities, and that p_j is the number of existentially quantified variables in Q_j. The set of

intensional predicates of the program Π_Q is

$$\{U_1, , \ldots, U_k, \textsc{Dom}, \textsc{Equal}, (\textsc{Equal}^{\mathbf{left},j})_{j\in W}, (\textsc{Equal}^{\mathbf{right},j})_{j\in W}, \textsc{Answer}, (\textsc{Answer}^{\mathbf{left},j})_{j\in W}, (\textsc{Answer}^{\mathbf{right},j})_{j\in W}\},$$

and the arity of each predicate is defined as follows:

- each U_i, for $i \in [1, k]$, has arity $n_i + m$;
- DOM has arity 1;
- EQUAL has arity $2 + m$;
- each predicate of the form $\textsc{Equal}^{\mathbf{left},j}$ or $\textsc{Equal}^{\mathbf{right},j}$, for $j \in W$, has arity $2 + p_j + m$;
- ANSWER has arity m; and
- each predicate of the form $\textsc{Answer}^{\mathbf{left},j}$ or $\textsc{Answer}^{\mathbf{right},j}$, for $j \in W$, has arity $p_j + m$.

The set of rules of Π_Q is defined as follows (if $\bar{y} = (y_1, \ldots, y_n)$, we use $\textsc{Dom}(\bar{y})$ as a shortening for $\textsc{Dom}(y_1), \ldots, \textsc{Dom}(y_n)$).

- For every predicate $T_i \in \mathbf{T}$, Π_Q includes the following n_i rules:

$$\begin{aligned}
\textsc{Dom}(x) &\leftarrow T_i(x, y_2, y_3, \ldots, y_{n_i-1}, y_{n_i}),\\
\textsc{Dom}(x) &\leftarrow T_i(y_1, x, y_3, \ldots, y_{n_i-1}, y_{n_i}),\\
&\ldots\\
\textsc{Dom}(x) &\leftarrow T_i(y_1, y_2, y_3, \ldots, y_{n_i-1}, x).
\end{aligned}$$

  Intuitively, predicate DOM collects the elements that belong to the domain of the extensional instance.
- Π_Q includes the following rules for predicate EQUAL:

$$\begin{aligned}
\textsc{Equal}(x, x, \bar{z}) &\leftarrow \textsc{Dom}(x), \textsc{Dom}(\bar{z}),\\
\textsc{Equal}(x, y, \bar{z}) &\leftarrow \textsc{Equal}(y, x, \bar{z}),\\
\textsc{Equal}(x, y, \bar{z}) &\leftarrow \textsc{Equal}(x, w, \bar{z}), \textsc{Equal}(w, y, \bar{z}).
\end{aligned}$$

- Π_Q includes the following rules for predicate $\textsc{Equal}^{\mathbf{left},j}$, for each $j \in W$, where $\bar{u}$ is a tuple of p_j fresh variables:

$$\begin{aligned}
\textsc{Equal}^{\mathbf{left},j}(x, x, \bar{u}, \bar{z}) &\leftarrow \textsc{Dom}(x), \textsc{Dom}(\bar{u}), \textsc{Dom}(\bar{z}),\\
\textsc{Equal}^{\mathbf{left},j}(x, y, \bar{u}, \bar{z}) &\leftarrow \textsc{Equal}^{\mathbf{left},j}(y, x, \bar{u}, \bar{z}),\\
\textsc{Equal}^{\mathbf{left},j}(x, y, \bar{u}, \bar{z}) &\leftarrow \textsc{Equal}^{\mathbf{left},j}(x, w, \bar{u}, \bar{z}), \textsc{Equal}^{\mathbf{left},j}(w, y, \bar{u}, \bar{z}).
\end{aligned}$$

- Π_Q includes the following rules for predicate $\textsc{Equal}^{\mathbf{right},j}$, for each $j \in W$, where $\bar{u}$ is a tuple of p_j fresh variables:

$$\begin{aligned}
\textsc{Equal}^{\mathbf{right},j}(x, x, \bar{u}, \bar{z}) &\leftarrow \textsc{Dom}(x), \textsc{Dom}(\bar{u}), \textsc{Dom}(\bar{z}),\\
\textsc{Equal}^{\mathbf{right},j}(x, y, \bar{u}, \bar{z}) &\leftarrow \textsc{Equal}^{\mathbf{right},j}(y, x, \bar{u}, \bar{z}),
\end{aligned}$$

$$\begin{aligned}\textsc{Equal}^{\mathbf{right},j}(x,y,\bar{u},\bar{z}) \quad &\leftarrow \quad \textsc{Equal}^{\mathbf{right},j}(x,w,\bar{u},\bar{z}),\\ &\qquad\ \textsc{Equal}^{\mathbf{right},j}(w,y,\bar{u},\bar{z}).\end{aligned}$$

- For every predicate U_i, $i \in [1,k]$, the program Π_Q includes the following rules, where $\bar{y} = (y_1,\ldots,y_{n_i})$, and $\bar{z} = (z_1,\ldots,z_m)$ are tuples of fresh variables:

$$U_i(\bar{y},\bar{z}) \quad \leftarrow \quad T_i(\bar{y}), \textsc{Dom}(\bar{z}).$$

- Let $i \in [1,\ell]$. First, assume that $Q_i(\bar{x})$ does not contain any inequality. Then $Q_i(\bar{x})$ is equal to $\exists\bar{u}(T_{s_1}(\bar{u}_1)\wedge\cdots\wedge T_{s_n}(\bar{u}_n))$, where $s_j \in [1,k]$ and every variable in $\bar{u}_j$ is mentioned in either $\bar{u}$ or $\bar{x}$, for every $j \in [1,n]$. In this case, program Π_Q includes the following rule:

$$\textsc{Answer}(\bar{x}) \quad \leftarrow \quad U_{s_1}(\bar{u}_1,\bar{x}),\ldots,U_{s_n}(\bar{u}_n,\bar{x}),\mathbf{C}(x_1),\ldots,\mathbf{C}(x_m).$$

  Notice that this rule is well defined since the set $\bar{x}$ is the set of free variables of $\exists\bar{u}(T_{s_1}(\bar{u}_1)\wedge\cdots\wedge T_{s_n}(\bar{u}_n))$. Second, assume that $Q_i(\bar{x})$ contains an inequality. Then $Q_i(\bar{x})$ is equal to the formula $\exists\bar{u}(T_{s_1}(\bar{u}_1)\wedge\cdots\wedge T_{s_n}(\bar{u}_n)\wedge v_1\neq v_2)$, where $s_i \in [1,k]$ and every variable in $\bar{u}_i$ is mentioned in either $\bar{u}$ or $\bar{x}$, for every $i \in [1,n]$, and v_1, v_2 are mentioned in $\bar{u}$ or $\bar{x}$. In this case, program Π_Q includes the following rules:

$$\begin{aligned}\textsc{Equal}(v_1,v_2,\bar{x}) \quad &\leftarrow \quad U_{s_1}(\bar{u}_1,\bar{x}),\ldots,U_{s_n}(\bar{u}_n,\bar{x}),\\ \textsc{Answer}(\bar{x}) \quad &\leftarrow \quad \textsc{Equal}(u,v,\bar{x}),\mathbf{C}(u),\mathbf{C}(v),u\neq v,\mathbf{C}(x_1),\ldots,\mathbf{C}(x_m).\end{aligned}$$

  We note that the first rule above is well defined since the query $\exists\bar{u}(T_{s_1}(\bar{u}_1)\wedge\cdots\wedge T_{s_n}(\bar{u}_n)\wedge v_1\neq v_2)$ is a safe query. Further, in this case Π_Q also contains the following rules for each $j \in W$, assuming $\bar{y}$ is a tuple of p_j fresh variables:

$$\begin{aligned}\textsc{Equal}^{\mathbf{left},j}(v_1,v_2,\bar{y},\bar{x}) \quad &\leftarrow \quad U_{s_1}(\bar{u}_1,\bar{x}),\ldots,U_{s_n}(\bar{u}_n,\bar{x}),\textsc{Dom}(\bar{y}),\\ \textsc{Equal}^{\mathbf{right},j}(v_1,v_2,\bar{y},\bar{x}) \quad &\leftarrow \quad U_{s_1}(\bar{u}_1,\bar{x}),\ldots,U_{s_n}(\bar{u}_n,\bar{x}),\textsc{Dom}(\bar{y}).\end{aligned}$$

  We note that the rules above are also well defined since $\exists\bar{u}(T_{s_1}(\bar{u}_1)\wedge\cdots\wedge T_{s_n}(\bar{u}_n)\wedge v_1\neq v_2)$ is a safe query. Finally, assume that $Q_i(\bar{x})$ contains two inequalities, and Q_i has constant joins and almost constant inequalities. Further, assume that $Q_i(\bar{x})$ is equal to the formula $\exists\bar{u}(T_{s_1}(\bar{u}_1)\wedge\cdots\wedge T_{s_n}(\bar{u}_n)\wedge v_1\neq v_2\wedge v_3\neq v_4)$, where each $s_j \in [1,k]$ and every variable in $\bar{u}_j$ is mentioned in either $\bar{u}$ or $\bar{x}$, for every $j \in [1,n]$, and v_1, v_2, v_3, and v_4 are mentioned in $\bar{u}$ or $\bar{x}$. In this case, program Π_Q includes the following rules:

$$\begin{aligned}\textsc{Equal}^{\mathbf{left},i}(v_1,v_2,\bar{u},\bar{x}) \quad &\leftarrow \quad U_{s_1}(\bar{u}_1,\bar{x}),\ldots,U_{s_n}(\bar{u}_n,\bar{x}),\\ \textsc{Equal}^{\mathbf{right},i}(v_3,v_4,\bar{u},\bar{x}) \quad &\leftarrow \quad U_{s_1}(\bar{u}_1,\bar{x}),\ldots,U_{s_n}(\bar{u}_n,\bar{x}).\end{aligned}$$

  Further, in this case Π_Q also contains the following rules for each $j \in W$, assuming $\bar{y}$ is a tuple of p_j fresh variables:

$$\begin{aligned}\textsc{Equal}^{\mathbf{left},j}(v_3,v_4,\bar{y},\bar{x}) \quad &\leftarrow \quad U_{p_1}(\bar{u}_1,\bar{x}),\ldots,U_{p_n}(\bar{u}_n,\bar{x}),\mathbf{C}(v_1),\\ &\qquad\ \textsc{Equal}^{\mathbf{left},j}(v_2,w,\bar{y},\bar{x}),\mathbf{C}(w),v_1\neq w,\end{aligned}$$

$$
\begin{aligned}
\textsc{Equal}^{\mathbf{left},j}(v_3, v_4, \bar{y}, \bar{x}) \quad &\leftarrow \quad U_{p_1}(\bar{u}_1, \bar{x}), \ldots, U_{p_n}(\bar{u}_n, \bar{x}), \mathbf{C}(v_2), \\
&\qquad \textsc{Equal}^{\mathbf{left},j}(v_1, w, \bar{y}, \bar{x}), \mathbf{C}(w), v_2 \neq w, \\
\textsc{Equal}^{\mathbf{left},j}(v_1, v_2, \bar{y}, \bar{x}) \quad &\leftarrow \quad U_{p_1}(\bar{u}_1, \bar{x}), \ldots, U_{p_n}(\bar{u}_n, \bar{x}), \mathbf{C}(v_3), \\
&\qquad \textsc{Equal}^{\mathbf{left},j}(v_4, w, \bar{y}, \bar{x}), \mathbf{C}(w), v_3 \neq w, \\
\textsc{Equal}^{\mathbf{left},j}(v_1, v_2, \bar{y}, \bar{x}) \quad &\leftarrow \quad U_{p_1}(\bar{u}_1, \bar{x}), \ldots, U_{p_n}(\bar{u}_n, \bar{x}), \mathbf{C}(v_4), \\
&\qquad \textsc{Equal}^{\mathbf{left},j}(v_3, w, \bar{y}, \bar{x}), \mathbf{C}(w), v_4 \neq w, \\
\textsc{Equal}^{\mathbf{right},j}(v_3, v_4, \bar{y}, \bar{x}) \quad &\leftarrow \quad U_{p_1}(\bar{u}_1, \bar{x}), \ldots, U_{p_n}(\bar{u}_n, \bar{x}), \mathbf{C}(v_1), \\
&\qquad \textsc{Equal}^{\mathbf{right},j}(v_2, w, \bar{y}, \bar{x}), \mathbf{C}(w), v_1 \neq w, \\
\textsc{Equal}^{\mathbf{right},j}(v_3, v_4, \bar{y}, \bar{x}) \quad &\leftarrow \quad U_{p_1}(\bar{u}_1, \bar{x}), \ldots, U_{p_n}(\bar{u}_n, \bar{x}), \mathbf{C}(v_2), \\
&\qquad \textsc{Equal}^{\mathbf{right},j}(v_1, w, \bar{y}, \bar{x}), \mathbf{C}(w), v_2 \neq w, \\
\textsc{Equal}^{\mathbf{right},j}(v_1, v_2, \bar{y}, \bar{x}) \quad &\leftarrow \quad U_{p_1}(\bar{u}_1, \bar{x}), \ldots, U_{p_n}(\bar{u}_n, \bar{x}), \mathbf{C}(v_3), \\
&\qquad \textsc{Equal}^{\mathbf{right},j}(v_4, w, \bar{y}, \bar{x}), \mathbf{C}(w), v_3 \neq w, \\
\textsc{Equal}^{\mathbf{right},j}(v_1, v_2, \bar{y}, \bar{x}) \quad &\leftarrow \quad U_{p_1}(\bar{u}_1, \bar{x}), \ldots, U_{p_n}(\bar{u}_n, \bar{x}), \mathbf{C}(v_4), \\
&\qquad \textsc{Equal}^{\mathbf{right},j}(v_3, w, \bar{y}, \bar{x}), \mathbf{C}(w), v_4 \neq w.
\end{aligned}
$$

Finally, the program Π_Q also includes the following rules for each $j \in W$, assuming $\bar{y}$ is a tuple of p_j fresh variables:

$$
\begin{aligned}
\textsc{Answer}^{\mathbf{left},j}(\bar{y}, \bar{x}) \quad &\leftarrow \quad \textsc{Equal}^{\mathbf{left},j}(u, v, \bar{y}, \bar{x}), \mathbf{C}(u), \mathbf{C}(v), u \neq v, \\
\textsc{Answer}^{\mathbf{right},j}(\bar{y}, \bar{x}) \quad &\leftarrow \quad \textsc{Equal}^{\mathbf{right},j}(u, v, \bar{y}, \bar{x}), \mathbf{C}(u), \mathbf{C}(v), u \neq v, \\
\textsc{Answer}(\bar{x}) \quad &\leftarrow \quad \textsc{Answer}^{\mathbf{left},j}(\bar{y}, \bar{x}), \\
&\qquad \textsc{Answer}^{\mathbf{right},j}(\bar{y}, \bar{x}), \mathbf{C}(x_1), \ldots, \mathbf{C}(x_m).
\end{aligned}
$$

Using Lemma 5.3, it is a tedious but not difficult task to prove that for every data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ and instance I of $\mathbf{S}$, $\text{certain}_{\mathcal{M}}(Q, I) = \text{certain}_{\mathcal{M}}(\Pi_Q, I)$. This can be done with the help of the intuition provided in Example 5.4. □

It immediately follows from Proposition 4.5 that if a data exchange setting $\mathcal{M}$ and a UCQ$^{\neq}$ query Q satisfy the conditions mentioned in Theorem 5.2, then CERTAIN-ANSWERS($\mathcal{M}$, Q) is in PTIME. Furthermore, it can also be shown that the properties of having constant joins and almost constant inequalities are helpful in reducing the complexity of computing certain answers to unions of conjunctive queries with at most one inequality per disjunct.

Proposition 5.5 *Let Q be a* UCQ$^{\neq}$ *query with at most one inequality per disjunct. Then* (1) *if every disjunct of Q has constant joins under a setting $\mathcal{M}$, then* CERTAIN-ANSWERS($\mathcal{M}$, Q) *is in* NLOGSPACE, *and* (2) *if in addition every disjunct of Q has almost constant inequalities under $\mathcal{M}$, then* CERTAIN-ANSWERS($\mathcal{M}$, Q) *is in* LOGSPACE.

Proof Before proving the proposition, we mention a couple of remarks that will be useful in the proof. First, it is immediate from the definition of canonical universal solution that $\text{CAN}(I)$ can be computed not only in polynomial time, but also in LOGSPACE for each source instance I. Second, if tuple $T(p_1, \ldots, p_n)$ belongs to $\text{CAN}(I)$ for an arbitrary source instance I under $\mathcal{M}$, and the i-th attribute of T ($1 \leq i \leq n$) is not existentially quantified in $\mathcal{M}$, then p_i has to be a constant.

We now prove the proposition, and start with part (1). Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a data exchange setting and Q a query that is the union of conjunctive queries, with at most one inequality per disjunct and without negated relational atoms, and such that each disjunct of Q has constant joins. We prove next that there exists a query Q', such that the data complexity of Q' is in NLOGSPACE and $\text{certain}_{\mathcal{M}}(Q, I) = Q'(\text{CAN}(I))$, for every source instance I. From this it immediately follows that CERTAIN-ANSWERS$(\mathcal{M}, Q)$ is in NLOGSPACE. Indeed an NLOGSPACE procedure can be constructed by composing two NLOGSPACE procedures; the first one that constructs $\text{CAN}(I)$ from I and the second one that evaluates Q' over $\text{CAN}(I)$. The result then follows from the fact that the class NLOGSPACE is closed under compositions (c.f. [21]).

The query Q' will be defined in *transitive closure logic* (for a precise definition of this logic, see e.g. Chap. 10.6 in [17]). In order to do so, need to introduce some terminology and an intermediate result (Lemma 5.6).

Assume that Q is $Q_1(\bar{x}) \vee \cdots \vee Q_\ell(\bar{x})$, where $\bar{x} = (x_1, \ldots, x_m)$, $m \geq 0$. Let I be an arbitrary source instance and $\bar{t} = (t_1, \ldots, t_m)$ a tuple of constants from I. We construct an undirected graph $G(Q, I, \bar{t})$ as follows:

- The nodes of $G(Q, I, \bar{t})$ are the elements in $\text{CAN}(I)$ plus two fresh elements μ and ν, i.e. neither μ nor ν belongs to $\text{CAN}(I)$;
- there exists an edge between elements p and p' in $G(Q, I, \bar{t})$ iff for some $i \in [1, \ell]$, (1) $Q_i(\bar{x})$ is of the form $\exists \bar{y}(\phi(\bar{x}, \bar{y}) \wedge u \neq v)$, where $\phi(\bar{x}, \bar{y})$ is a conjunction of relational atoms over $\mathbf{T}$, and $u, v \in \{\bar{x}, \bar{y}\}$, and (2) there is an assignment $\sigma : \{\bar{x}, \bar{y}\} \to \text{dom}(\text{CAN}(I))$, such that $\sigma(\bar{x}) = \bar{t}$, $(\text{CAN}(I), \sigma) \models \phi(\bar{x}, \bar{y}) \wedge u \neq v$, $\sigma(u) = p$ and $\sigma(v) = p'$; and
- there exists an edge between μ and ν in $G(Q, I, \bar{t})$ iff for some $i \in [1, \ell]$, $Q_i(\bar{x})$ is of the form $\exists \bar{y} \phi(\bar{x}, \bar{y})$, where $\phi(\bar{x}, \bar{y})$ is a conjunction of relational atoms over $\mathbf{T}$, and $\text{CAN}(I) \models Q_i(\bar{t})$.

We say that $G(Q, I, \bar{t})$ has a *contradiction* path (or c-path), if there is a path in $G(Q, I, \bar{t})$ from a constant $c \in \text{dom}(\text{CAN}(I))$ to a different constant $c' \in \text{dom}(\text{CAN}(I))$, or an edge between μ and ν. Notice that this construction is a simplified version of the graph used in the proof of Theorem 5.2. As for the case of Lemma 5.3 in Theorem 5.2, we present a key lemma that characterizes certain answers in terms of c-paths in the graph $G(Q, I, \bar{t})$. This can be proved using techniques along the lines of those used in the proof of Lemma 5.3.

Lemma 5.6 *Let Q be as defined above. For every source instance I and tuple $\bar{t}$ of constants from I, it is the case that*

$$\bar{t} \in \text{certain}_{\mathcal{M}}(Q, I) \quad \Leftrightarrow \quad G(Q, I, \bar{t}) \text{ has a c-path.}$$

We define the query Q' in three steps. Assume, without loss of generality, that for each $1 \leq i \leq s \leq \ell$, $Q_i(\bar{x})$ is of the form $\exists \bar{y}_i(\phi_i(\bar{x}, \bar{y}_i) \wedge u_i \neq v_i)$, where $\phi_i(\bar{x}, \bar{y}_i)$ is a conjunction of relational atoms over $\mathbf{T}$, and $u_i, v_i \in \{\bar{x}, \bar{y}_i\}$, and for each $s < j \leq \ell$, $Q_j(\bar{x})$ is of the form $\exists \bar{y}_j \phi_j(\bar{x}, \bar{y}_j)$, where $\phi_j(\bar{x}, \bar{y}_j)$ is a conjunction of relational atoms over $\mathbf{T}$. Then:

1. Define a formula $A(z_1, z_2, \bar{x})$ as follows, where z_1 and z_2 are fresh variables, i.e. z_1 and z_2 are not mentioned in $Q(\bar{x})$:

$$A(z_1, z_2, \bar{x}) \equiv \bigvee_{1 \leq i \leq s} \exists \bar{y}_i(\phi_i(\bar{x}, \bar{y}_i) \wedge z_1 \neq z_2 \wedge z_1 = u_i \wedge z_2 = v_i).$$

   Intuitively, the formula $A(z_1, z_2, \bar{x})$ defines the adjacency in the graph $G(Q, I, \bar{x})$, with respect to elements in $\text{CAN}(I)$;
2. define a formula $\alpha(\bar{x})$ as follows,

$$\alpha(\bar{x}) \equiv \bigvee_{s < j \leq \ell} \exists \bar{y}_j \phi_j(\bar{x}, \bar{y}_j).$$

   Intuitively $\alpha(\bar{x})$ checks whether there is an edge between μ and ν in $G(Q, I, \bar{x})$; and
3. finally, the query $Q'(\bar{x})$ is defined as:

$$(\alpha(\bar{x}) \vee \exists w_1 \exists w_2(\mathbf{C}(w_1) \wedge \mathbf{C}(w_2) \wedge w_1 \neq w_2 \wedge (w_1, w_2) \in \text{TrCl}.A(u, v, \bar{x})))$$
$$\wedge \mathbf{C}(x_1) \wedge \ldots \mathbf{C}(x_m)$$

   where $(w_1, w_2) \in \text{TrCl}.A(u, v, \bar{x})$ expresses that the pair (w_1, w_2) belongs to the transitive closure of the adjacency relation defined by the pairs (u, v) that satisfy A parameterized by $\bar{x}$.

It immediately follows from Lemma 5.6 that for every source instance I, $\text{certain}_{\mathcal{M}}(Q, I) = Q'(\text{CAN}(I))$. Further, it is well-known that the data complexity of any formula in transitive closure logic is in NLOGSPACE (see e.g. Chap. 10.6 in [17]). This concludes the first part of the proposition.

Now we prove part (2). Let Q be as in the first part of the proof, but with the addition that each disjunct of Q has almost constant inequalities. Lemma 5.6 continues being the case in this setting, but notice that now if there is a c-path in $G(Q, I, \bar{t})$ then there is a c-path of length at most 2. Thus, in this case $Q'(\bar{x})$ can be expressed as the FO formula that checks whether there is an edge between μ and ν in $G(Q, I, \bar{t})$, or a c-path of length at most 2 in $G(Q, I, \bar{t})$. Since the data complexity of any FO formula is in LOGSPACE (see e.g. Chap. 6 in [17]), we conclude that the problem of computing certain answers for this class of queries and settings is in LOGSPACE. □

An obvious question at this point is how natural the conditions used in Theorem 5.2 are. Although we cannot settle this subjective question, we are at least able to show that these conditions are optimal in the sense that removing any of them leads to intractability for the class of $\text{UCQ}^{\neq}$ queries with two inequalities.

Theorem 5.7

(1) *There exist a* LAV *data exchange setting* $\mathcal{M}$ *and a query* Q *such that* Q *is the union of a Boolean conjunctive query and a Boolean conjunctive query with two inequalities that has both constant joins and almost constant inequalities under* $\mathcal{M}$, *and such that* CERTAIN-ANSWERS($\mathcal{M}$, Q) *is* coNP*-complete.*

(2) *There exist a* LAV *data exchange setting* $\mathcal{M}$ *and a Boolean conjunctive query* Q *with two inequalities, such that* Q *has constant joins under* $\mathcal{M}$, Q *does not have almost constant inequalities under* $\mathcal{M}$ *and* CERTAIN-ANSWERS($\mathcal{M}$, Q) *is* coNP*-complete.*

(3) *There exist a* LAV *data exchange setting* $\mathcal{M}$ *and a Boolean conjunctive query* Q *with two inequalities, such that* Q *has almost constant inequalities under* $\mathcal{M}$, Q *does not have constant joins under* $\mathcal{M}$ *and* CERTAIN-ANSWERS($\mathcal{M}$, Q) *is* coNP*-complete.*

Proof We only present here the proof of the first part of the theorem. For the details of the second and third part see Appendix A.2. The proof for (1) is as follows: The LAV setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ is defined as follows. The source schema $\mathbf{S}$ consists of two relations: A binary relation P and a ternary relation R. The target schema $\mathbf{T}$ consists of three relations: Two binary relations T and S, and a ternary relation U. Further, Σ_{st} is the following set of source-to-target dependencies:

$$\begin{aligned} P(x, y) &\rightarrow \exists z(T(x, z) \wedge T(y, z) \wedge S(x, y)), \\ R(x, y, z) &\rightarrow U(x, y, z). \end{aligned}$$

Furthermore, Boolean query Q is defined as:

$$\begin{aligned} &\exists x \exists y \exists z(U(x, y, z) \wedge T(x, x) \wedge T(y, y) \wedge T(z, z)) \\ &\quad \vee \exists x \exists y \exists w \exists z(T(x, y) \wedge T(w, z) \wedge S(x, w) \wedge x \neq y \wedge w \neq z). \end{aligned}$$

We denote the first disjunct of Q by Q_1 and the second by Q_2. Clearly, Q_2 has constant joins and almost constant inequalities in $\mathcal{M}$. On the other hand, Q_1 does not have constant joins. Next we show that CERTAIN-ANSWERS($\mathcal{M}$, Q) is coNP-complete.

Membership of CERTAIN-ANSWERS($\mathcal{M}$, Q) in coNP follows from [9]. The coNP-hardness is established from a reduction from 3SAT to the complement of CERTAIN-ANSWERS($\mathcal{M}$, Q). More precisely, for every 3CNF propositional formula ϕ, we construct in polynomial time an instance I_ϕ of $\mathbf{S}$ such that ϕ is satisfiable iff $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$.

Given a propositional formula $\phi \equiv \bigwedge_{1 \leq j \leq m} C_j$ in 3CNF, where each C_j is a clause, let I_ϕ be the following source instance:

- The interpretation of P in I_ϕ contains the pair $(q, \neg q)$, for each propositional variable q mentioned in ϕ; and
- the interpretation of R in I_ϕ contains all tuples (α, β, γ) such that for some $1 \leq j \leq m$, $C_j = (\alpha \vee \beta \vee \gamma)$.

Clearly, I_ϕ can be constructed in polynomial time from ϕ. The canonical universal solution J for I_ϕ is as follows, where we denote by $\perp_q$ (or $\perp_{\neg q}$) the null generated by applying the std $P(x, y) \rightarrow \exists z(T(x, z) \wedge T(y, z) \wedge S(x, y))$ to $P(q, \neg q)$:

- The interpretation of the relation T in J contains the tuples $(q, \bot_q)$ and $(\neg q, \bot_q)$, for each propositional variable q mentioned in ϕ;
- the interpretation of the relation S in J is just a copy of the interpretation of the relation P in I_ϕ; and
- the interpretation of the relation U in J is just a copy of the interpretation of the relation R in I_ϕ.

We prove now that ϕ is satisfiable iff $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$.

($\Rightarrow$) Assume that ϕ is satisfiable, and let κ be a truth assignment for the propositional variables of ϕ such that $\kappa(\phi) = 1$. From κ, define a function f from J into J as follows:

$$f(v) = \begin{cases} \neg q & v = \bot_q \text{ and } \kappa(q) = 1, \\ q & v = \bot_q \text{ and } \kappa(q) = 0, \\ v & \text{otherwise.} \end{cases}$$

Let J^* be the solution for I_ϕ obtained from J by replacing each occurrence of an element v in J by $f(v)$. We show next that $Q(J^*) = \texttt{false}$, and, thus, that $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$.

Assume, for the sake of contradiction, that $Q(J^*) = \texttt{true}$. Then $Q_1(J^*) = \texttt{true}$ or $Q_2(J^*) = \texttt{true}$. Assume first that the latter holds. Then there is a function $h : \{x, y, z, w\} \to \text{dom}(J^*)$ such that $T(h(x), h(y))$, $T(h(z), h(w))$, and $S(h(x), h(z))$ are all tuples in J^*, and $h(x) \neq h(y)$ and $h(z) \neq h(w)$. Since $S(h(x), h(z))$ belongs to J^*, it follows that for some propositional variable q mentioned in ϕ, $h(x) = q$ and $h(z) = \neg q$. Further, since $T(h(x), h(y))$ and $T(h(z), h(w))$ belong to J^*, we have that $h(y) = f(\bot_q) = h(w)$. But then $f(\bot_q) \neq q$ and $f(\bot_q) \neq \neg q$, which contradicts the definition of J^*. Assume, on the other hand, that $Q_1(J^*) = \texttt{true}$. Then there is a function $h : \{x, y, z\} \to \text{dom}(J^*)$ such that the tuples $U(h(x), h(y), h(z))$, $T(h(x), h(x))$, $T(h(y), h(y))$, and $T(h(z), h(z))$ are all tuples in J^*. Then by definition of $\mathcal{M}$ and I_ϕ, there exists a clause $(\alpha \vee \beta \vee \gamma)$ in ϕ such that $h(x) = \alpha$, $h(y) = \beta$, and $h(z) = \gamma$. Since $L(h(x), h(x)) = L(\alpha, \alpha)$ belongs to J^*, it follows that $f(\bot_\alpha) = \alpha$, and thus, that $\kappa(\alpha) = 0$. Similarly, $\kappa(\beta) = 0$ and $\kappa(\gamma) = 0$. But this is a contradiction, since $\kappa(\phi) = 1$, and thus, $\kappa(\alpha) = 1$ or $\kappa(\beta) = 1$ or $\kappa(\gamma) = 1$.

($\Leftarrow$) Assume that $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$. Then there exists a solution J' such that $Q(J') = \texttt{false}$. Let $h : J \to J'$ be an homomorphism from J into J', and let κ be the following truth assignment for the propositional variables mentioned in ϕ: $\kappa(q) = 1$ iff $h(\bot_q) = \neg q$. We show next that $\kappa(C_j) = 1$, for each $1 \leq j \leq m$, and, thus, that ϕ is satisfiable.

Consider an arbitrary $j \in [1, m]$, and assume that $C_j = (\alpha \vee \beta \vee \gamma)$. Then, since $U(\alpha, \beta, \gamma)$, $T(\alpha, h(\bot_\alpha))$, $T(\beta, h(\bot_\beta))$, and $T(\gamma, h(\bot_\gamma))$ belong to J', it must be the case that $\alpha \neq h(\bot_\alpha)$ or $\beta \neq h(\bot_\beta)$ or $\gamma \neq h(\bot_\gamma)$. Further, since either $S(\alpha, \neg\alpha)$ or $S(\neg\alpha, \alpha)$ belongs to J', and both $T(\alpha, h(\bot_\alpha))$ and $T(\neg\alpha, h(\bot_\alpha))$ belong to J', we conclude from the fact that $Q_2(J') = \texttt{false}$ that $h(\bot_\alpha) = \alpha$ or $h(\bot_\alpha) = \neg\alpha$. Similarly, $h(\bot_\beta) = \beta$ or $h(\bot_\beta) = \neg\beta$, and $h(\bot_\gamma) = \gamma$ or $h(\bot_\gamma) = \neg\gamma$. Thus, $h(\bot_\alpha) = \neg\alpha$ or $h(\bot_\beta) = \neg\beta$ or $h(\bot_\gamma) = \neg\gamma$, and, therefore, that $\kappa(\alpha) = 1$ or $\kappa(\beta) = 1$ or $\kappa(\gamma) = 1$. We conclude that $\kappa(C_j) = 1$.

This concludes the proof of the first part of the theorem. □

It is important to notice that although the problem of computing certain answers to UCQ$^{\neq}$ queries has been considered in the literature, none of the results of Theorem 5.7 directly follows from any of the known results for this problem. In particular, Fagin et al. showed in [9] a similar result to (1), namely that the problem of computing certain answers is coNP-complete even for the union of two queries, the first of which is a conjunctive query and the second of which is a conjunctive query with two inequalities. The difficulty in our case is that the second query is restricted to have constant joins and almost constant inequalities, while Fagin et al. considered a query that does not satisfy any of these conditions. Moreover, Mądry proved in [20] a similar result to (2) and (3), namely that the problem of computing certain answers is coNP-complete for conjunctive queries with two inequalities. The difficulty in our case is that we consider a query that has constant joins in (2) and a query that has almost constant inequalities in (3), while Mądry considered a query that does not satisfy any of these conditions. In fact, we provide in (2) and (3) two new proofs of the fact that the problem of computing certain answer to a conjunctive query with two inequalities is coNP-complete.

We conclude this section with a remark about the possibility of using the conditions defined in this section to obtain tractability for UCQ$^{\neq}$. As we mentioned above, the notions of constant joins and almost constant inequalities can be extended to UCQ$^{\neq}$ queries with an arbitrary number of inequalities. Thus, one may wonder whether these conditions lead to tractability in this general scenario. Unfortunately, the following proposition shows that this is not the case, even for the class of UCQ$^{\neq}$ queries with three inequalities.

Proposition 5.8 *There exist a* LAV *data exchange setting* $\mathcal{M}$ *and a Boolean conjunctive query* Q *with three inequalities, such that* Q *has both constant joins and almost constant inequalities under* $\mathcal{M}$*, but the problem* CERTAIN-ANSWERS($\mathcal{M}$, Q) *is* coNP*-complete.*

Proof The LAV setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ is as follows. The source schema $\mathbf{S}$ consists of two relations: A binary relation P and a ternary relation R. The target schema $\mathbf{T}$ also consists of two relations: A binary relation T and a ternary relation S. Further, Σ_{st} is the following set of source-to-target dependencies:

$$\begin{aligned} P(x, y) &\rightarrow \exists z(T(x, z) \wedge T(y, z)), \\ R(x, y, z) &\rightarrow S(x, y, z). \end{aligned}$$

Furthermore, Boolean query Q is defined as:

$$\begin{aligned} &\exists x_1 \exists y_1 \exists x_2 \exists y_2 \exists x_3 \exists y_3 (S(x_1, x_2, x_3) \wedge T(x_1, y_1) \wedge T(x_2, y_2) \wedge T(x_3, y_3) \\ &\quad \wedge x_1 \neq y_1 \wedge x_2 \neq y_2 \wedge x_3 \neq y_3). \end{aligned}$$

Clearly, Q has almost constant inequalities and constant joins in $\mathcal{M}$. Next we show that the problem CERTAIN-ANSWERS($\mathcal{M}$, Q) is coNP-complete.

Membership of CERTAIN-ANSWERS($\mathcal{M}, Q$) in coNP follows from [9]. The coNP-hardness is established from a reduction from 3SAT to the complement of CERTAIN-ANSWERS($\mathcal{M}, Q$). More precisely, for every 3CNF propositional formula ϕ, we construct in polynomial time an instance I_ϕ of $\mathbf{S}$ such that ϕ is satisfiable iff $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$.

Given a propositional formula $\phi \equiv \bigwedge_{1 \le j \le m} C_j$ in 3CNF, where each C_j is a clause, let I_ϕ be the following source instance:

- The interpretation of P in I_ϕ contains the pair $(q, \neg q)$, for each propositional variable q mentioned in ϕ; and
- the interpretation of R in I_ϕ contains all tuples (α, β, γ) such that for some $1 \le j \le m$, $C_j = (\alpha \vee \beta \vee \gamma)$.

Clearly, I_ϕ can be constructed in polynomial time from ϕ.

The canonical universal solution J for I_ϕ is as follows, where we denote by $\perp_q$ (or $\perp_{\neg q}$) the null generated by applying the std $P(x, y) \rightarrow \exists z(T(x, z) \wedge T(y, z))$ to $P(q, \neg q)$:

- The interpretation of the relation T in J contains the tuples $(q, \perp_q)$ and $(\neg q, \perp_q)$, for each propositional variable q mentioned in ϕ; and
- the interpretation of the relation S in J is just a copy of the interpretation of the relation R in I_ϕ.

We prove now that ϕ is satisfiable iff $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$.

($\Rightarrow$) Assume that ϕ is satisfiable, and let κ be a truth assignment for the propositional variables of ϕ such that $\kappa(\phi) = 1$. From κ, define a function f from J into J as follows:

$$f(v) = \begin{cases} q & v = \perp_q \text{ and } \kappa(q) = 1, \\ \neg q & v = \perp_q \text{ and } \kappa(q) = 0, \\ v & \text{otherwise.} \end{cases}$$

Let J^* be the solution for I_ϕ obtained from J by replacing each occurrence of an element v in J by $f(v)$. We show next that $Q(J^*) = \texttt{false}$, and, thus, that $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$.

Assume, for the sake of contradiction, that $Q(J^*) = \texttt{true}$. Then there is a function $h : \{x_1, x_2, x_3, y_1, y_2, y_3\} \rightarrow \text{dom}(J^*)$ such that $S(h(x_1), h(x_2), h(x_3))$, $T(h(x_1), h(y_1))$, $T(h(x_2), h(y_2))$, $T(h(x_3), h(y_3))$ are all tuples in J^*, and $h(x_1) \neq h(y_1)$, $h(x_2) \neq h(y_2)$, and $h(x_3) \neq h(y_3)$. Then by definition of $\mathcal{M}$ and I_ϕ, there exists a clause $(\alpha \vee \beta \vee \gamma)$ in ϕ such that $h(x_1) = \alpha$, $h(x_2) = \beta$, and $h(x_3) = \gamma$. Since $L(h(x_1), h(y_1)) = L(\alpha, f(\perp_\alpha))$ belongs to J^*, and $\alpha = h(x_1) \neq h(y_1) = f(\perp_\alpha)$, it follows that $\kappa(\alpha) = 0$. Similarly, $\kappa(\beta) = 0$ and $\kappa(\gamma) = 0$. But this is a contradiction, since $\kappa(\phi) = 1$, and thus, $\kappa(\alpha) = 1$, $\kappa(\beta) = 1$, or $\kappa(\gamma) = 1$.

($\Leftarrow$) Assume that $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$. Then there exists a solution J' such that $Q(J') = \texttt{false}$. Let $h : J \rightarrow J'$ be an homomorphism from J into J', and let κ be the following truth assignment for the propositional variables mentioned in ϕ: $\kappa(q) = 1$ iff $h(\perp_q) = q$. We show next that $\kappa(C_j) = 1$, for each $1 \le j \le m$, and, thus, that ϕ is satisfiable.

Consider an arbitrary $j \in [1, m]$, and assume that $C_j = (\alpha \vee \beta \vee \gamma)$. Then, since $S(\alpha, \beta, \gamma)$, $T(\alpha, h(\bot_\alpha))$, $T(\beta, h(\bot_\beta))$, and $T(\gamma, h(\bot_\gamma))$ belong to J', it must be the case that $\alpha = h(\bot_\alpha)$ or $\beta = h(\bot_\beta)$ or $\gamma = h(\bot_\gamma)$. It follows that $\kappa(\alpha) = 1$ or $\kappa(\beta) = 1$ or $\kappa(\gamma) = 1$, and, thus, $\kappa(C_j) = 1$.

This concludes the proof of the proposition. □

6 The Combined Complexity of Query Answering

Beyond the usual data complexity analysis, it is natural to ask for the combined complexity of the problem of computing certain answers: What is the complexity if data exchange settings and queries are not considered to be fixed? To state this problem, we shall extend the notation defined in Sect. 2. Let $\mathcal{DE}$ be a class of data exchange settings and $\mathcal{C}$ a class of queries. In this section, we study the following problem:

PROBLEM:	CERTAIN-ANSWERS($\mathcal{DE}, \mathcal{C}$).
INPUT:	A data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}) \in \mathcal{DE}$, a source instance I, a query $Q \in \mathcal{C}$ and a tuple $\bar{t}$ of constants from I.
QUESTION:	Is $\bar{t} \in \text{certain}_{\mathcal{M}}(Q, I)$?

It is worth mentioning that a related study appeared in [14]. Even though the focus of that paper was the combined complexity of the existence of solutions problem, some of the results in [14] can be extended to the certain answers problem. In particular, some complexity bounds for unions of conjunctive queries with inequalities can be proved by using these results. Nevertheless, in this section we prove stronger lower bounds that consider single conjunctive queries with inequalities, and which cannot be directly proved by using the results of [14].

We start by stating the complexity for the case of DATALOG$^{\mathbf{C}(\neq)}$ queries. The study continues by considering some restrictions of DATALOG$^{\mathbf{C}(\neq)}$ that lead to lower combined complexity, and which are expressed in the form of conjunctive queries with single inequalities. We conclude this study by examining unrestricted CQ$^{\neq}$ queries, which are not rewritable in DATALOG$^{\mathbf{C}(\neq)}$ (under the assumption that PTIME $\neq$ NP). The results of this section are summarized in Table 1, where we let k-CQ$^{\neq}$ be the class of CQ$^{\neq}$ queries with at most k inequalities.

6.1 Combined Complexity of DATALOG$^{\mathbf{C}(\neq)}$ Queries

We showed in Proposition 3.3 that the certain answers of a DATALOG$^{\mathbf{C}(\neq)}$ program can be computed by directly posing the query over the canonical universal solution. It can be shown that such an approach can compute the certain answers to a DATALOG$^{\mathbf{C}(\neq)}$ program in exponential time, although canonical universal solutions can be of exponential size if data exchange settings are not considered to be fixed. And not only that, it can be proved that this is a tight bound.

Theorem 6.1 CERTAIN-ANSWERS(GLAV, DATALOG$^{\mathbf{C}(\neq)}$) *is* EXPTIME*-complete.*

Proof The EXPTIME-hardness follows directly from Theorems 4.1 and 6.3. More precisely, the problem of computing certain answers is shown in Theorem 6.3 to be EXPTIME-hard for the class of conjunctive queries with single inequalities, and it follows from Theorem 4.1 that for each query Q in this class, one can construct in polynomial time a DATALOG$^{\mathbf{C}(\neq)}$ program Π_Q such that $\text{certain}_{\mathcal{M}}(Q, I) = \text{certain}_{\mathcal{M}}(\Pi_Q, I)$, for every source instance I.

To show that CERTAIN-ANSWERS(GLAV, DATALOG$^{\mathbf{C}(\neq)}$) is in EXPTIME, assume that Π is a DATALOG$^{\mathbf{C}(\neq)}$ program, I is a source instance and Σ_{st} is a set of st-tgds. In Sect. 3, it is proved that to compute the certain answers of Π over I under Σ_{st}, it suffices to evaluate Π over CAN(I). Thus, given that CAN(I) can be computed in exponential time and is of size $O(\|I\|^{\|\Sigma_{st}\|})$, where $\|I\|$ and $\|\Sigma_{st}\|$ represent the size of I and Σ_{st}, respectively, we conclude that the certain answers to Π over I under Σ_{st} can be computed in time $O(\|I\|^{\|\Sigma_{st}\|\cdot\|\Pi\|})$, where $\|\Pi\|$ represents the size of Π, since Π can be evaluated over an instance D in time $O(\|D\|^{\|\Pi\|})$ [2, 22].

Note that the above problem has to deal with canonical universal solutions of exponential size. Then restricting these solutions to be of polynomial size would be a natural approach to reduce the complexity of the problem. There are at least two ways to do this. The obvious one would be to fix the data exchange settings, and leave only queries and source instances as input.[2] The less obvious but more interesting case is to restrict the class of data exchange settings to be LAV settings. However, for the case of DATALOG$^{\mathbf{C}(\neq)}$ programs, the combined complexity is inherently exponential, and thus reducing the size of canonical universal solutions does not help in improving the upper bound. □

Proposition 6.2 CERTAIN-ANSWERS(LAV, DATALOG$^{\mathbf{C}(\neq)}$) *is* EXPTIME*-complete.*

Proof The membership in EXPTIME follows from Theorem 6.1. For the EXPTIME-hardness, we show a reduction from the problem of checking whether a tuple $\bar{t}$ belongs to the evaluation of a DATALOG program Π over an instance I. This problem is well known to be EXPTIME-hard (see e.g. [2, 22]). Let Π be a DATALOG program defined over a schema $\mathbf{S}$, I an instance of $\mathbf{S}$ and $\bar{t}$ a tuple of elements from I. Next we show how to construct a LAV data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ and a DATALOG$^{\mathbf{C}(\neq)}$ program Π' such that $\bar{t} \in \Pi(I)$ if and only if $\bar{t} \in \text{certain}_{\mathcal{M}}(\Pi', I)$, which shows that there exists a polynomial time reduction from the problem mentioned above to our problem. Define $\mathbf{T}$ as a schema that contains a relation symbol $\widehat{R}$ of the same arity of R, for every relation symbol R in $\mathbf{S}$. Moreover, define Π' as a copy of Π where every predicate R from $\mathbf{S}$ is replaced by predicate $\widehat{R}$, and define Σ_{st} to include a dependency $R(\bar{x}) \rightarrow \widehat{R}(\bar{x})$, for every predicate symbol R in $\mathbf{S}$. Then we have that $\bar{t} \in \Pi(I)$ if and only if $\bar{t} \in \Pi'(\text{CAN}(I))$, which implies that $\bar{t} \in \Pi(I)$ if and only if $\bar{t} \in \text{certain}_{\mathcal{M}}(\Pi', I)$ by the results in Sect. 3.

It was shown in Theorem 4.1 that every conjunctive query with one inequality can be efficiently translated into a DATALOG$^{\mathbf{C}(\neq)}$ program. Hence, the class of 1-CQ$^{\neq}$

[2]Indeed, for obtaining a canonical solution of polynomial size it would be enough to fix the maximum arity of a relation symbol in the target schema.

Table 1 Combined complexity of computing certain answers

Query	GLAV setting	LAV setting
DATALOG$^{\mathbf{C}(\neq)}$	EXPTIME-complete	EXPTIME-complete
1-CQ$^{\neq}$	EXPTIME-complete	NP-complete
k-CQ$^{\neq}$, $k \geq 2$	coNEXPTIME-complete	Π_2^p-complete
CQ$^{\neq}$	coNEXPTIME-complete	Π_2^p-complete

queries form a subclass of the class of DATALOG$^{\mathbf{C}(\neq)}$ programs. Thus, it is natural to ask whether the EXPTIME lower bounds proved in this section carry over this class, and whether the LAV restriction could be useful in this case. These are the motivating questions for the next section. □

6.2 Combined Complexity of CQ$^{\neq}$

We leave the DATALOG$^{\mathbf{C}(\neq)}$ queries to concentrate on the analysis of CQ$^{\neq}$ queries in data exchange. We first study the class 1-CQ$^{\neq}$, that is, the class of conjunctive queries with only one inequality. It is worth mentioning that an EXPTIME lower bound can be obtained from [14] for the case of unions of 1-CQ$^{\neq}$ queries. We refine this result to the case of 1-CQ$^{\neq}$ queries, and therefore present a stronger lower bound.

Theorem 6.3 CERTAIN-ANSWERS(GLAV, 1-CQ$^{\neq}$) *is* EXPTIME-*complete.*

Proof Membership in EXPTIME is a corollary of Theorems 4.1 and 6.1. The proof of EXPTIME-hardness is a refinement of a proof given in [14], where it was essentially shown that the problem of computing certain answers is EXPTIME-hard for a union of two CQ$^{\neq}$ queries. The EXPTIME-hardness is established from a reduction from the Single Rule Datalog Problem [11], which is the following problem: given a DATALOG program Π consisting of only one rule and some of facts with only constants, is it the case that a tuple $\bar{t}$ belongs to the evaluation of Π over the empty instance? That is, we ask whether $\bar{t} \in \Pi(\emptyset)$. We shall call these programs Single Rule Datalog Programs (sirup). It is important to notice that each of these programs contains a single intensional predicate A, and it may include some facts with only constants about this predicate, that is, some facts of the form $A(\bar{c})$. These facts are needed when the only rule of the program is recursive, as otherwise the evaluation of the program would be empty. The combined complexity of the aforementioned problem was shown to be EXPTIME-complete by Gottlob and Papadimitriou [11].

As in [14], let Π be a program containing some facts with only constants and a single rule of the form $A(\bar{x}) \leftarrow Q_1(\bar{x}_1), \ldots, Q_n(\bar{x}_n)$, where each symbol Q_i $(1 \leq i \leq n)$ either represents an extensional database predicate or the only intensional predicate A. Furthermore, we assume that $\bar{t} = (c_1, \ldots, c_k)$ and we say that $\bar{t}$ belongs to the evaluation of Π over the empty instance if and only if $\bar{t} \in A^{\mathcal{T}_\Pi^\infty(\emptyset)}$.

The idea of the reduction in [14] is to precompute all the possible tuples that can be returned from the sirup rule into the canonical universal solution of the source

instance, and then simulate the sirup rule with a $\mathrm{CQ}^{\neq}$ query. A second query is used to check whether $\bar{t} \in \Pi(\emptyset)$. The difficulty in our case is to show that the bound remains the same even for a single $\mathrm{CQ}^{\neq}$ query with one inequality.

We now define a data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, a query Q and an instance I of $\mathbf{S}$ such that $\bar{t} \in \Pi(\emptyset)$ if and only if $\mathrm{certain}_{\mathcal{M}}(Q, I) = \mathtt{true}$.

- The source schema $\mathbf{S}$ consists of four unary relations T, V, F, S plus all the extensional predicate symbols $R_1, \ldots, R_m$ of Π, and two additional relation symbols A and W. The arity of the relations R_i $(1 \leq i \leq m)$ is the same as the corresponding arity in Π, denoted by l_i, the arity of the relation A is k and the arity of the relation W is $k+1$.
- The target schema $\mathbf{T}$ consists of relations $R'_1, \ldots, R'_m$, T' and A'. The arity of relation R'_i is $l_i + 3$ $(1 \leq i \leq m)$, T' is unary, and A' has arity $k+3$.
- Source instance I is defined as follows:
  - The interpretation of predicate R_i $(1 \leq 1 \leq m)$ in I is the same as in Π. Furthermore, the interpretation of predicate A in I consists of all the tuples $\bar{c}$ such that $A(\bar{c})$ is a fact in Π.
  - The relation W only contains one tuple, based on $\bar{t}$: $W(c_1, \ldots, c_k, d)$, where d is a fresh value not occurring elsewhere in I.
  - We create a single tuple for each relation T, F and S using constants c, 1, 2, also not used elsewhere in I: $T(c)$, $F(1)$ and $S(2)$.
  - Finally, we populate the unary relation V with all distinct values from Π and $\bar{t}$.

  Intuitively, the constants 1 and 2 will allow us to use the same query for both simulating the sirup rule and checking whether $\bar{t} \in \Pi(\emptyset)$; the relations containing value 1 will be used for the simulation of the sirup rule, while the relations containing value 2 will be used when checking whether $\bar{t} \in \Pi(\emptyset)$.
- The set Σ_{st} of dependencies is defined as follows. We create a copy of the relation T in T':

$$T(x) \quad \rightarrow \quad T'(x).$$

For every $i \in \{1, \ldots, m\}$, we create a copy of the facts about R_i in the program Π, so that they can be used when simulating the sirup rule:

$$F(z) \wedge T(y) \wedge R_i(x_1, \ldots, x_{l_i}) \quad \rightarrow \quad R'_i(x_1, \ldots, x_{l_i}, y, z, z).$$

Notice that we use the value z in F to indicate that these tuples will be used for the simulation of the sirup rule. Next, for every $i \in \{1, \ldots, m\}$, we populate R'_i in the target with a series of tuples built using every possible constant value in the source:

$$S(z) \wedge T(y) \wedge V(x_1) \wedge \cdots \wedge V(x_{l_i}) \quad \rightarrow \quad R'_i(x_1, \ldots, x_{l_i}, y, z, z).$$

It is important to notice that in this case, we use the value z in S to indicate that these tuples will be used when checking whether $\bar{t} \in \Pi(\emptyset)$. We then copy the relation A into A', to indicate that every fact in Π also holds in every solution for I under $\mathcal{M}$:

$$F(z) \wedge S(w) \wedge T(y) \wedge A(x_1, \ldots, x_k) \quad \rightarrow \quad A'(x_1, \ldots, x_k, y, z, w). \quad (28)$$

In this rule, the value c in the position $k+1$ of A' indicate that tuple $(x_1, \ldots, x_k)$ belongs to the interpretation of A in Π. Moreover, we also add to A' every possible tuple that could be generated with the values in Π and $\bar{t}$:

$$F(z) \wedge S(w) \wedge V(x_1) \wedge \cdots \wedge V(x_k) \quad \rightarrow \quad \exists n A'(x_1, \ldots, x_k, n, z, w). \quad (29)$$

Notice that in this rule, a null value is placed in the position $k+1$ of A' to indicate that tuple $(x_1, \ldots, x_k)$ has not yet been shown to belong to the interpretation of A in Π. As in the previous cases, relations F and S are used in the preceding two rules to ensure that the procedures are run in the correct order, that is, the query must first compute the tuples, and then check whether $\bar{t} \in \Pi(\emptyset)$. Finally, we need some extra tuples for the simulation of the sirup rule. We copy the relation W into A' and add again every possible tuple to the relation A', but considering a different order of predicates S, F and T:

$$S(z) \wedge F(y) \wedge W(x_1, \ldots, x_k, u) \quad \rightarrow \quad A'(x_1, \ldots, x_k, u, z, y), \quad (30)$$

$$S(z) \wedge T(y) \wedge V(x_1) \wedge \cdots \wedge V(x_k) \quad \rightarrow \quad A'(x_1, \ldots, x_k, y, z, y). \quad (31)$$

When showing that the reduction is correct, it will become clear why we need to use different orders of predicate F, S and T in the preceding rules.

- To define query Q, recall that we are considering a sirup rule of the form $A(\bar{x}) \leftarrow Q_1(\bar{x}_1), \ldots, Q_n(\bar{x}_n)$, where each Q_i can be either one of the extensional databases predicates R_j or the predicate A. Assume that $\bar{x}'$ is a tuple containing all the variables in the body of the sirup rule that are not mentioned in $\bar{x}$. Then query Q is defined as follows:

$$Q = \exists u \exists v \exists w \exists w_1 \cdots \exists w_n \exists x_1 \cdots \exists x_k \exists y \exists z \exists \bar{x}' \Big[A'(x_1, \ldots, x_k, z, u, w)$$

$$\wedge T'(y) \ \wedge A'(x_1, \ldots, x_k, y, w, v) \wedge z \neq y \wedge \bigwedge_{1 \leq i \leq n} Q_i'(\bar{x}_i, y, u, w_i) \Big].$$

Before proving that the reduction works properly, we describe the canonical universal solution for I, and give some intuition about the definitions of $\mathcal{M}$ and Q. For every $i \in \{1, \ldots, m\}$, relation R_i' contains tuples of the form $(\bar{a}, c, 1, 1)$, for every tuple $\bar{a}$ that belongs to the interpretation of R_i in I, and also the tuples of the form $(\bar{b}, c, 2, 2)$, for all the possible tuples $\bar{b}$ generated by using the elements in V^I (which correspond to all the tuples generated from the values mentioned in Π and $\bar{t}$). The relation T' is a copy of the relation T in I. The tuples in the relation A' result from the last four dependencies. First, due to the mapping (28), we copy every tuple in A from I into A', and add the constants $c, 1, 2$ in its last three positions. Second, for every possible tuple $\bar{b}$ generated by using the elements in V^I, mapping (29) includes in A' a tuple of the form $(\bar{b}, \perp, 1, 2)$, where $\perp$ is a fresh null value. We shall generically describe the null values added by (29) as $\perp$. Third, mapping (30) copies the relation W and adds the constants $2, 1$ to each of the tuples in W. Finally, for every possible tuple $\bar{b}$ generated by using the elements in V^I, mapping (31) includes in A' the tuple $(\bar{b}, c, 2, c)$.

Let us now give some intuition about the definitions of $\mathcal{M}$ and Q. To show that Q does not hold in every possible solution for I under $\mathcal{M}$, one can compute the canonical universal solution J for I under $\mathcal{M}$, and then try to replace some of the nulls of J in order to generate a solution for I where Q does not hold. Each one of these replacements represents either an application of the sirup rule to add a tuple to the predicate A or a test of whether $\bar{t} \in \Pi(\emptyset)$. More precisely, assume that one has found an assignment ρ for the variables of Q that satisfies the body of this query. By examining the set of possible tuples of A' in J, we know that $\rho(u)$ is either 1 or 2. The former case represents an application of the sirup rule, while the latter is used to test whether $\bar{t} \in \Pi(\emptyset)$. In particular, if $\rho(u) = 1$, then given that $\rho(y) = c$ (since $T'(y)$ is a conjunct of Q) and $\rho(y) \neq \rho(z)$ (since $y \neq z$ is a conjunct of Q), we conclude that $\rho(z)$ is a null value and $\rho(w) = 2$ by examining the set of possible tuples of A' in J having 1 as their $k+2$ argument. Thus, in this case we have to replace $\rho(z)$ by value c in order to construct a solution for I under $\mathcal{M}$ where Q does not hold. But in our reduction, the fact that tuple $(\rho(x_1), \ldots, \rho(x_k), c, 1, 2)$ is added to A' indicates that tuple $(\rho(x_1), \ldots, \rho(x_k))$ is included in A when computing $\Pi(\emptyset)$. It is important to notice that this represents a correct application of sirup rule $A(\bar{x}) \leftarrow Q_1(\bar{x}_1), \ldots, Q_n(\bar{x}_n)$, as from the fact that

$$\bigwedge_{1 \leq i \leq n} Q'_i(\rho(\bar{x}_i), c, 1, \rho(w_i))$$

holds and the definitions of I and $\mathcal{M}$, one can conclude that every atom $Q_i(\rho(\bar{x}_i))$ holds in $\Pi(\emptyset)$ (in particular, if $Q'_i = R'_j$, then by examining the tuples of R'_j in J having 1 in its penultimate argument, one concludes that $Q_i(\rho(\bar{x}_i))$ belongs to Π). On the other hand, if $\rho(u) = 2$, then by examining the set of possible tuples of A' in J and given that $\rho(y) = c$, we conclude that $\rho(w) = 1$ and $\rho(z) = d$. Thus, it is not possible to replace by $\rho(z)$ by $\rho(y)$ in this case, and one concludes that $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{true}$ (as formally shown below). But this corresponds with our intention of checking whether $\bar{t} \in \Pi(\emptyset)$. In fact, given that the only tuple in A' having d in its $k+1$ argument is generated from tuple $W(c_1, \ldots, c_k, d)$, we have that $\rho(x_i) = c_i$, for every $i \in \{1, \ldots, k\}$. Thus, by considering the conjunct $A'(x_1, \ldots, x_k, y, w, v)$ of Q, we conclude that $A'(c_1, \ldots, c_k, c, 1, \rho(v))$ holds, which means by the manner value 1 is used in our reduction (described above) that tuple $(c_1, \ldots, c_k)$ belongs to $\Pi(\emptyset)$.

Next, we formally show that $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{true}$ if and only if $\bar{t} \in \Pi(\emptyset)$.

($\Rightarrow$) If $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{true}$, then Q holds in all the possible solutions for I under $\mathcal{M}$. We use this condition to define the following sequence $J_0, \ldots, J_i, \ldots$ of solutions for I.

- J_0 is the canonical universal solution for I under $\mathcal{M}$.
- Assume that there exists a tuple $\bar{t}_i$ such that $\bar{t}_i$ witnesses the satisfaction of the body of Q in J_i and z is assigned a null value $\perp$ in $\bar{t}_i$. Then J_{i+1} is generated from J_i by replacing $\perp$ by the value assigned to y in $\bar{t}_i$.

We note that for every tuple $\bar{t}_i$ used to generate the sequence $J_0, \ldots, J_i, \ldots$, the value assigned to y in $\bar{t}_i$ is constant c. Thus, we have that the sequence $J_0, \ldots, J_i, \ldots$ is finite, and we let J_m be its last element. By definition of $\mathcal{M}$, and given that J_m is a

solution for I and $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{true}$, we have that there exists a tuple $\bar{t}_m$ such that $\bar{t}_m$ witnesses the satisfaction of the body of Q in J_m, z is assigned value d in $\bar{t}_m$ and y is assigned value c in $\bar{t}_m$. Furthermore, we also have that $A'(\bar{t}, d, 2, 1)$ and $A'(\bar{t}, c, 1, 2)$ are both tuples in J_m.

For every $i \in \{0, \ldots, m\}$ and tuple $\bar{t}_i$, let $\bar{a}_i$ be the restriction of $\bar{t}_i$ to the variables $x_1, \ldots, x_k$. In particular, we have that $\bar{a}_m = \bar{t}$. By induction on i, next we show that $A(\bar{a}_i) \in \mathcal{T}_{\Pi}^{i+1}(\emptyset)$.

- Base case: For every $j \in \{1, \ldots, n\}$, let $\bar{b}_j$ be the restriction of $\bar{a}_0$ to the tuple of variables $\bar{x}_j$. By definition of J_0 and $\bar{t}_0$, we have that for every $j \in \{1, \ldots, n\}$, if $Q'_j = R'_p$, for some $p \in \{1, \ldots, m\}$, then $R'_p(\bar{b}_j, c, 1, 1)$ holds in J_0, and if $Q'_j = A'$, then $A'(\bar{b}_j, c, 1, 2)$ holds in J_0. Thus, from the definition of J_0, we have that for every $j \in \{1, \ldots, n\}$, if $Q'_j = R'_p$, for some $p \in \{1, \ldots, m\}$, then $R_p(\bar{b}_j)$ is a fact in Π, and if $Q'_j = A'$, then $A(\bar{b}_j)$ is a fact in Π. Therefore, by definition of $\bar{t}_0$, we conclude that $A(\bar{a}_0)$ can be deduced from the facts of Π and, thus, $A(\bar{a}_0) \in \mathcal{T}_{\Pi}^{1}(\emptyset)$.
- Inductive step: Assume that for every $q < i$, it holds that $A(\bar{a}_q) \in \mathcal{T}_{\Pi}^{q+1}(\emptyset)$, and let $\bar{b}_j$ be the restriction of $\bar{a}_i$ to the tuple of variables $\bar{x}_j$, for every $j \in \{1, \ldots, n\}$. By definition of J_i and $\bar{t}_i$, we have that for every $j \in \{1, \ldots, n\}$, if $Q'_j = R'_p$, for some $p \in \{1, \ldots, m\}$, then $R'_p(\bar{b}_j, c, 1, 1)$ holds in J_i and, thus, $R_p(\bar{b}_j)$ is a fact in Π by definition of the sequence $J_0, \ldots, J_m$. On the other hand, if $Q'_j = A'$, then $A'(\bar{b}_j, c, 1, 2)$ holds in J_i. Let $q \leq i$ be the smallest index such that $A'(\bar{b}_j, c, 1, 2)$ holds in J_q. If $q = 0$, then $A(\bar{b}_j)$ is a fact in Π and, therefore, $A(\bar{b}_j) \in \mathcal{T}_{\Pi}^{q+1}(\emptyset)$. If $q > 0$, then $A'(\bar{b}_j, c, 1, 2)$ was included in J_q when replacing the z-value of $\bar{t}_{q-1}$ by the y-value of this tuple. Thus, by induction hypothesis, we have that $A(\bar{b}_j) \in \mathcal{T}_{\Pi}^{q+1}(\emptyset)$. Therefore, for every $j \in \{1, \ldots, n\}$, we have that $Q_j(\bar{b}_j) \in \mathcal{T}_{\Pi}^{i}(\emptyset)$, which implies that $A(\bar{a}_i) \in \mathcal{T}_{\Pi}^{i+1}(\emptyset)$ by the definition of $\bar{t}_i$ and $\bar{a}_i$.

Hence, we have that $A(\bar{a}_m) \in \mathcal{T}_{\Pi}^{m+1}(\emptyset)$ and, therefore, $\bar{t} \in \Pi(\emptyset)$ since $\bar{a}_m = \bar{t}$. This concludes the first part of the proof.

($\Leftarrow$) Assume that $\bar{t} \in \Pi(\emptyset)$ and, for the sake of contradiction, assume that $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$. Moreover, let $\hat{Q}(u, v, w, w_1, \ldots, w_n, x_1, \ldots, x_k, y, z, \bar{x}')$ be a query obtained by removing the existential quantifiers from Q:

$$A'(x_1, \ldots, x_k, z, u, w) \wedge T'(y) \wedge A'(x_1, \ldots, x_k, y, w, v) \wedge z \neq y \wedge \bigwedge_{1 \leq i \leq n} Q'_i(\bar{x}_i, y, u, w_i).$$

To obtain a contradiction, we define a sequence of solutions $J_0, \ldots, J_i, \ldots$ and a corresponding sequence of sets of tuples $T_0, \ldots, T_i, \ldots$ as follows.

- Let J_0 be the canonical universal solution for I under $\mathcal{M}$, and $T_0 = \hat{Q}(J_0)$, that is, the evaluation of $\hat{Q}$ over J_0.
- For every $i \geq 0$, let J_{i+1} be obtained from J_i by replacing every null value $\perp$ in a tuple of T_i by the constant c, if $\perp$ witnesses the inequality of $\hat{Q}$. Moreover, let $T_{i+1} = \hat{Q}(J_{i+1})$.

Given that J_0 has finite number of null values, we have that the sequence $J_0, \ldots, J_i, \ldots$ is finite, and we let J_m be its last element. Next we show that from the assumption that $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$, one can deduce that $Q(J_m) = \texttt{false}$. Let ψ be the following dependency:

$$\forall u \forall v \forall w \forall w_1 \cdots \forall w_n \forall x_1 \cdots \forall x_k \forall y \forall z \forall \bar{x}' \Big[(A'(x_1, \ldots, x_k, z, u, w)$$

$$\wedge\, T'(y) \wedge A'(x_1, \ldots, x_k, y, w, v) \wedge \bigwedge_{1 \leq i \leq n} Q'_i(\bar{x}_i, y, u, w_i)) \rightarrow (z = y) \Big].$$

It is easy to see that solution J_m can be obtained from J_0 as the result of repeatedly chasing J_0 with ψ [9]. Thus, it follows from [9] that $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$ if and only if $Q(J_m) = \texttt{false}$. Therefore, we conclude that $Q(J_m) = \texttt{false}$.

We now show that the fact that $Q(J_m) = \texttt{false}$ leads to a contradiction. Consider the program evaluation sequence $\mathcal{T}^0_\Pi(\emptyset), \ldots, \mathcal{T}^m_\Pi(\emptyset)$.

Claim 6.4 *For every $i \in \{0, \ldots, m\}$, if $A(\bar{a})$ holds in $\mathcal{T}^i_\Pi(\emptyset)$, then tuple $A'(\bar{a}, c, 1, 2)$ holds in J_i.*

Proof By induction on $i \in \{0, \ldots, m\}$.

- Base case: Assume that $A(\bar{a})$ holds in $\mathcal{T}^0_\Pi(\emptyset)$. Then $A(\bar{a})$ is a fact in Π and, thus, given that J_0 is the canonical universal solution for I under $\mathcal{M}$, we conclude that $A'(\bar{a}, c, 1, 2)$ holds in J_0.
- Inductive step: Assume that the property holds for every $j < i$ and that $A(\bar{a})$ holds in $\mathcal{T}^i_\Pi(\emptyset)$. If $A'(\bar{a}, c, 1, 2)$ holds in J_{i-1}, then by definition of the sequence $J_0, \ldots, J_m$, we have that $A'(\bar{a}, c, 1, 2)$ holds in J_i. Thus, assume that $A'(\bar{a}, c, 1, 2)$ does not hold in J_{i-1}, and notice this implies that $A'(\bar{a}, \perp, 1, 2)$ holds in J_{i-1}, where $\perp$ is a null value, and that $A(\bar{a})$ does not hold in $\mathcal{T}^{i-1}_\Pi(\emptyset)$ (otherwise by induction hypothesis we obtain that $A'(\bar{a}, c, 1, 2)$ holds in J_{i-1}). But then we have that $A(\bar{a})$ can be deduced from $\mathcal{T}^{i-1}_\Pi(\emptyset)$ by using the only rule in Π. Thus, there exists an instantiation $A(\bar{a}) \leftarrow Q_1(\bar{a}_1), \ldots, Q_n(\bar{a}_n)$ of the rule of Π such that $Q_1(\bar{a}_1), \ldots, Q_n(\bar{a}_n)$ belong to $\mathcal{T}^{i-1}_\Pi(\emptyset)$. Thus, by induction hypothesis and the definition of the sequence $J_0, \ldots, J_m$, we conclude that for every $p \in \{1, \ldots, n\}$, if $Q'_p = R_q$ for some $q \in \{1, \ldots, m\}$, then $R_q(\bar{a}_p, c, 1, 1)$ holds in J_{i-1}, and if $Q'_p = A'$, then $A'(\bar{a}_p, c, 1, 2)$ holds in J_{i-1}. Therefore, given that both $A'(\bar{a}, \perp, 1, 2)$ and $A'(\bar{a}, c, 2, c)$ hold in J_{i-1}, we conclude that one of the tuples of T_{i-1} has $\perp$ as a witness for the inequality of $\hat{Q}$. This implies that $A'(\bar{a}, c, 1, 2)$ holds in J_i since $\perp$ is replaced by c to obtain J_i from J_{i-1}. □

By Claim 6.4 and the definitions of sequence $J_0, \ldots, J_m$ and data exchange setting $\mathcal{M}$, we conclude that $\mathcal{T}^m_\Pi(\emptyset) = \mathcal{T}^{m+1}_\Pi(\emptyset)$. Thus, given that $\bar{t} \in \Pi(\emptyset)$, we have that $A(\bar{t})$ holds in $\mathcal{T}^m_\Pi(\emptyset)$. Therefore, by Claim 6.4, we have that $A'(\bar{t}, c, 1, 2)$ holds in J_m. But this implies that $Q(J_m) = \texttt{true}$ since (1) $A'(\bar{t}, d, 2, 1)$ holds in J_m, (2) $A'(\bar{b}, c, 2, c)$ holds in J_m for every k-tuple $\bar{b}$ of elements from V^I, and (3) $R_i(\bar{b}_i, c, 2, 2)$ holds in J_m for every l_i-tuple $\bar{b}_i$ of elements from V^I ($i \in \{1, \ldots, m\}$).

But this contradicts our initial assumption. This concludes the proof of the theorem. □

It is natural to ask what happens in the case of unrestricted queries and, more specifically, for queries with two inequalities. It was noted that the data complexity becomes higher when dealing with two inequalities, and a similar behavior should be expected for the combined complexity. Indeed, we have that:

Theorem 6.5 *For every* $k \geq 2$, CERTAIN-ANSWERS(GLAV, k-CQ$^{\neq}$) *is* coNEXPTIME*-complete.*

For the sake of readability, we just give here a brief sketch of the proof of Theorem 6.5, and we leave the rather technical proof of this theorem for Appendix A.3.

Proof sketch First, we prove the membership in coNEXPTIME. In [9], it was proved that given a UCQ$^{\neq}$ query Q and a data exchange setting $\mathcal{M}$, the problem CERTAIN-ANSWERS($\mathcal{M}$, Q) is in coNP. An inspection of this proof reveals that if $\mathcal{M}$ and Q are not assumed to be fixed, then the same proof shows that CERTAIN-ANSWERS(GLAV, UCQ$^{\neq}$) is in coNEXPTIME. Thus, we conclude that CERTAIN-ANSWERS(GLAV, k-CQ$^{\neq}$) is in coNEXPTIME.

The coNEXPTIME-hardness is established by a reduction from the satisfiability problem for the Bernays-Schönfinkel class of FO sentences to the complement of CERTAIN-ANSWERS(GLAV, 2-CQ$^{\neq}$). Formally, the Bernays-Schönfinkel class of FO sentences is defined as the class of all FO formulas of the form $\exists\bar{x}\forall\bar{y}\psi(\bar{x},\bar{y})$, where $\psi(\bar{x},\bar{y})$ is quantifier-free and mentions neither any function symbol nor the equality symbol. Then the satisfiability problem for the Bernays-Schönfinkel class is the problem of verifying, given a formula $\exists\bar{x}\forall\bar{y}\psi(\bar{x},\bar{y})$ in this class, whether there exists a structure that satisfies $\exists\bar{x}\forall\bar{y}\psi(\bar{x},\bar{y})$. This problem is known to be NEXPTIME-complete (see, e.g., [6]). □

As we mentioned in the previous section, if data exchange settings are not considered to be fixed, then one has to deal with canonical universal solutions of exponential size when computing certain answers. A natural way to avoid this problem is by restricting the class of data exchange settings to be LAV settings. For the case of DATALOG$^{\mathbf{C}(\neq)}$ programs, this restriction does not help in reducing the complexity of computing certain answers. However, the evaluation of CQ$^{\neq}$ queries is not inherently exponential and, thus, we are able to considerably reduce the complexity by considering LAV settings, as we show in the following proposition.

Proposition 6.6 CERTAIN-ANSWERS(LAV, 1-CQ$^{\neq}$) *is* NP*-complete, and* CERTAIN-ANSWERS(LAV, k-CQ$^{\neq}$) *is* Π_2^p*-complete for every* $k \geq 2$.

Proof That the problem CERTAIN-ANSWERS(LAV, 1-CQ$^{\neq}$) is NP-complete can be proved using techniques in [9] for membership, and in [14] for hardness. Furthermore, the membership of CERTAIN-ANSWERS(LAV, k-CQ$^{\neq}$) in Π_2^p follows from [1]. Thus, we only need show that CERTAIN-ANSWERS(LAV, k-CQ$^{\neq}$) is Π_2^p-hard.

The Π_2^p-hardness is established by a reduction from $\forall\exists$ 3-SAT, which is the problem of verifying, given a Boolean formula ψ in 3-CNF with variables partitioned into sets $\bar{x}$ and $\bar{z}$, whether it is true that for every truth assignment of the variables in $\bar{x}$, there exists a truth assignment of the variables in $\bar{z}$ so that ψ is satisfied with the overall assignment. This problem is known to be Π_2^p-complete.

Let ϕ be a formula of the form $\forall\bar{x}\exists\bar{z}\bigwedge_{1\leq k\leq \ell}\psi_k$, where each ψ_k $(1 \leq k \leq \ell)$ is a clause containing exactly three literals. Let $\bar{x} = (x_1, \ldots, x_n)$ and $\bar{z} = (z_1, \ldots, z_m)$. Based on ϕ, we show how to construct in polynomial time a LAV data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, a query Q and an instance I such that $\text{certain}_{\mathcal{M}}(Q, I) =$ `true` if and only if ϕ is satisfiable. More precisely, the LAV setting $(\mathbf{S},\mathbf{T},\Sigma_{st})$ and the source instance I are defined as follows:

- The source schema $\mathbf{S}$ consists of ℓ ternary relations $C_1, \ldots, C_\ell$, and a relation A of arity four.
- The target schema $\mathbf{T}$ consists of two unary relations U', O', n unary relations $X'_1, \ldots, X'_n$, m unary relations $Z'_1, \ldots, Z'_m$, two binary relations R' and T' and ℓ ternary relations $C'_1, \ldots, C'_\ell$, that are intended to be copies of the relations $C_1, \ldots, C_\ell$.
- The elements of the source instance I are the constants $1, 0, a, d$. The interpretation of the relation A in I contains the single tuple $(0, 1, a, d)$. For each $k \in \{1, \ldots, \ell\}$, the interpretation of the relation C_k in I contains the tuple (d, d, d), plus seven tuples of the form (u, v, w), where u, v, w represent the values of the satisfying assignments for ψ_k. For example, if $\psi_k \equiv (x_1 \vee \neg x_2 \vee z_1)$, then C_k^I consists of the following tuples: (d, d, d), $(0, 0, 0)$, $(0, 0, 1)$, $(0, 1, 1)$, $(1, 0, 0)$, $(1, 0, 1)$, $(1, 1, 0)$ and $(1, 1, 1)$. Notice that tuple $(0, 1, 0)$ is not included in C_k^I as it does not represent a satisfying assignment for ψ_k.
- The set Σ_{st} of source-to-target dependencies is defined as follows:
  - First, we create the tuples $O'(0)$ and $U'(1)$ in $\text{CAN}(I)$:

$$A(x, y, z, w) \quad \rightarrow \quad O'(x), \tag{32}$$

$$A(x, y, z, w) \quad \rightarrow \quad U'(y). \tag{33}$$

  - Next, for every $i \in \{1, \ldots, n\}$, we add a rule that is intended to create the following tuples in $\text{CAN}(I)$ (where $\perp_i$ is a fresh null value): $X'_i(\perp_i)$, $X'_i(d)$, $T'(\perp_i)$, $R'(a, \perp_i)$ and $R'(\perp_i, d)$:

$$A(x, y, z, w) \quad \rightarrow \quad \exists n\big(X'_i(n) \wedge X'_i(w) \wedge T'(n) \wedge R'(z, n) \wedge R'(n, w)\big). \tag{34}$$

  - For every $j \in \{1, \ldots, m\}$, we also add the following st-tgd:

$$A(x, y, z, w) \quad \rightarrow \quad Z'_j(x) \wedge Z'_j(y) \wedge Z'_j(w). \tag{35}$$

    The purpose of this set of dependencies is to add the following tuples to $\text{CAN}(I)$: $Z'_j(0)$, $Z'_j(1)$ and $Z'_j(d)$, for every $j \in \{1, \ldots, m\}$.
  - Next, we add to $\text{CAN}(I)$ the tuples $R'(a, 0)$, $R'(a, 1)$ and $T'(a)$:

$$A(x, y, z, w) \quad \rightarrow \quad T'(z) \wedge R'(z, x) \wedge R'(z, y). \tag{36}$$

– Finally, for every $k \in \{1, \ldots, n\}$, we add a rule that creates a copy of the relation C_k in $\text{CAN}(I)$:

$$C_k(x, y, z) \quad \rightarrow \quad C'_k(x, y, z). \tag{37}$$

Furthermore, the query Q is defined as follows:

$$Q = \exists b \exists e \exists g \exists v_1 \cdots \exists v_n \exists w_1 \cdots \exists w_m \Bigg[\Bigg(\bigwedge_{1 \leq i \leq n} X'_i(v_i) \wedge R'(g, v_i) \Bigg) \wedge \Bigg(\bigwedge_{1 \leq j \leq m} Z'_j(w_j) \wedge R'(g, w_j) \Bigg) \wedge \alpha(v_1, \ldots, v_n, w_1, \ldots, w_m) \wedge U'(e) \wedge O'(b) \wedge T'(g) \wedge g \neq b \wedge g \neq e \Bigg],$$

where $\alpha(v_1, \ldots, v_n, w_1, \ldots, w_m)$ is defined as follows. For every $k \in [1, \ell]$, let u_1^k, u_2^k, u_3^k be the propositional variables of ψ_k. Further, let χ be a function such that $\chi(x_i) = v_i$ and $\chi(z_j) = w_j$, for every $i \in [1, n]$ and $j \in [1, m]$. Then

$$\alpha(v_1, \ldots, v_n, w_1, \ldots, w_m) = \bigwedge_{1 \leq k \leq \ell} C'_k(\chi(u_1^k), \chi(u_2^k), \chi(u_3^k)).$$

For example, if ϕ is the formula $\forall x_1 \forall x_2 \exists z_1 \exists z_2 \exists z_3((x_1 \vee x_2 \vee z_1) \wedge (\neg x_1 \vee z_2 \vee \neg z_3))$, then α is defined as $\alpha(v_1, v_2, w_1, w_2, w_3) = C'_1(v_1, v_2, w_1) \wedge C'_2(v_1, w_2, w_3)$.

Before we continue with the proof, we give some intuition about the reduction. For every $i \in [1, n]$, the relation X'_i is intended to store the truth value of the variable x_i in ϕ, and for every $j \in [1, m]$, the relation Z'_j is intended to store the truth value for the variable z_j in ϕ. As previously shown, for each variable in $z_j \in \bar{z}$, the tuples $Z'_j(0)$ and $Z'_j(1)$ belong to $\text{CAN}(I)$, while for each variable $x_i \in \bar{x}$, only the tuple $X'_i(\perp_i)$ belongs to the canonical universal solution for I. We are interested in those solutions in which every null $\perp_i$ in $\text{CAN}(I)$ $(1 \leq i \leq n)$ has been replaced with an element 0 or 1. Each one of these solutions represents a particular valuation for the variables in $\bar{x}$: For every $i \in \{1, \ldots, n\}$, the valuation assigns the value 1 to the variable x_i if and only if the null $\perp_i$ in the tuple $X'_i(\perp_i)$ in $\text{CAN}(I)$ is replaced with the element 1.

Intuitively, the first task of the query Q is to select only those solutions in which every null has been replaced with the element either 0 or 1. Roughly speaking, if J is an arbitrary solution in which there is a null that has not been replaced with either 0 or 1, then the evaluation of Q over J must be true. This is done with the help of the relation T'. Let $\perp$ be the aforementioned null of J. Then one can always construct a satisfying assignment ρ for the variables of Q as follows: ρ assigns the element d to every variable except for g, b and e, that are assigned the values $\perp$, 0 and 1, respectively. The second task of Q is to verify whether for every valuation of the universally quantified variables of ϕ, there exists a valuation of the existentially quantified variables that satisfy ϕ. Recall that every solution J in which the nulls of $\text{CAN}(I)$ have been replaced with the element 0 or 1 represents a particular valuation $\sigma_{\bar{x}}$ for the variables in $\bar{x}$. Further, notice that, since for every $j \in [1, m]$, $\text{CAN}(I)$ contains the

tuples $Z'_j(1)$ and $Z'_j(0)$, every solution J for I contains essentially every possible valuation for the variables in $\bar{z}$. Then the query Q will choose a specific valuation $\sigma_{\bar{z}}$ for the existentially quantified variables such that ϕ is satisfied by the valuations $(\sigma_{\bar{x}}, \sigma_{\bar{z}})$. Intuitively, the satisfying valuation for the existentially quantified variables comes from the tuples in J that witness the predicates $Z'_1(w_1), \ldots, Z'_m(w_m)$ of the body of Q. More precisely, it can be shown that the evaluation of Q over one of the selected solutions J is true if and only if there exists a valuation $\sigma_{\bar{z}}$ for the variables in $\bar{z}$ such that ϕ is satisfied by the valuation $(\sigma_{\bar{x}}, \sigma_{\bar{z}})$. Finally, to compute the certain answers of Q for I one must check if Q holds in every solution for I under $\mathcal{M}$. Intuitively, by doing this we are verifying whether for every possible valuation of the variables in $\bar{x}$, there exists a valuation for the variables of $\bar{z}$ such that ϕ holds under those valuations.

We now prove that $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{true}$ if and only if ϕ is satisfiable. In fact, it is more convenient to show that $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$ if and only if ϕ is not satisfiable.

($\Leftarrow$) Assume that ϕ is not satisfiable, that is, there exists a valuation $\sigma_{\bar{x}}$ of the universally quantified variables such that ϕ does not hold under any possible valuation of the existentially quantified variables. Define a function h from $\text{CAN}(I)$ to $\text{CAN}(I)$ as follows:

- $h(y) = 1$ if $y = \perp_i$, there is a tuple $X'_i(\perp_i)$ in $\text{CAN}(I)$ and $\sigma_{\bar{x}}(x_i) = 1$;
- $h(y) = 0$ if $y = \perp_i$, there is a tuple $X'_i(\perp_i)$ in $\text{CAN}(I)$, and $\sigma_{\bar{x}}(x_i) = 0$; and
- $h(y) = y$ otherwise.

Let $J^\star$ be the solution obtained from the canonical solution $\text{CAN}(I)$ by replacing each element y in $\text{CAN}(I)$ with $h(y)$. We now show that $Q(J^\star) = \texttt{false}$ and, thus, $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$. Assume, for the sake of contradiction, that $J^\star$ satisfies Q. Then there must exist an assignment ρ of the variables of Q that satisfy the body of the query. Depending on the value of $\rho(g)$, we have two cases:

- Assume first that $\rho(g) = h(\perp_i)$ for some $i \in [1, n]$, where $\perp_i$ is the null value in the tuple $X'_i(\perp_i)$ in the canonical solution for I. Notice that the only tuples in the interpretation of the relations O' and U' in $J^\star$ are $O'(0)$ and $U'(1)$, respectively. Thus, $\rho(b) = 0$ and $\rho(e) = 1$. From the definition of Q, it must be the case that $h(\perp_i) \neq 0$ and $h(\perp_i) \neq 1$, but this contradicts the definition of h.
- Assume now that $\rho(g) = a$. Then, for every v_i, it must be the case that $\rho(v_i) = h(\perp_i)$, or, in other words, $\rho(v_i) = \sigma_{\bar{x}}(x_i)$. Choose a valuation $\sigma_{\bar{z}}$ such that for every $j \in \{1, \ldots, m\}$, $\sigma_{\bar{z}}(z_j) = 1$ if ρ assigns the element 1 to the variable w_j in Q, and $\sigma_{\bar{z}}(z_j) = 0$ if ρ assigns the element 0 to w_j. Then given that for every $k \in [1, \ell]$, the interpretation of the relation C'_k in $J^\star$ contains all the tuples corresponding to the satisfying assignments of ψ_k, it is easy to see that the valuation $(\sigma_{\bar{x}}, \sigma_{\bar{z}})$ satisfies all the clauses in ϕ. More precisely, for every $k \in \{1, \ldots, \ell\}$, truth assignment $(\sigma_{\bar{x}}, \sigma_{\bar{z}})$ assigns the values $\rho(\chi(u^k_1))$, $\rho(\chi(u^k_2))$ and $\rho(\chi(u^k_3))$ to the propositional variables u^k_1, u^k_2 and u^k_3, respectively. Since $(\rho(\chi(u^k_1)), \rho(\chi(u^k_2)), \rho(\chi(u^k_3)))$ is a satisfying assignment for ψ_k, it must be the case that ψ_k holds under $(\sigma_{\bar{x}}, \sigma_{\bar{z}})$. This also leads to a contradiction, since we assumed that ϕ is not satisfiable.

($\Rightarrow$) Assume that $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$. Then there is a solution $J^\star$ where Q does not hold. Let h be an homomorphism from $\text{CAN}(I)$ to $J^\star$. We first claim that

for every null $\perp$ in $\text{CAN}(I)$, it must be the case that $h(\perp) = 0$ or $h(\perp) = 1$. Assume, for the sake of contradiction, that for some $i \in [1, n]$, the tuple $X'_i(h(\perp_i))$ in $J^\star$ is such that $h(\perp_i) \neq 0$ and $h(\perp_i) \neq 1$. But then, given that all the tuples $T'(h(\perp_i))$, $R'(h(\perp_i), d)$, $X'_i(d)$ $(1 \leq i \leq n)$, $Z'_j(d)$ $(1 \leq j \leq m)$ and $C'_k(d, d, d)$ $(1 \leq k \leq \ell)$ belong to $J^\star$, we obtain that $Q(J^\star) = \texttt{true}$, which contradicts our initial assumption.

Next, to prove that ϕ is not satisfiable, we provide a truth assignment $\sigma_{\bar{x}}$ for the universally quantified variables of ϕ, and then prove that under this assignment, the evaluation of ϕ is false under every valuation of the existentially quantified variables of ϕ. The valuation $\sigma_{\bar{x}}$ is defined as follows:

- $\sigma_{\bar{x}}(x_i) = 1$ if $\perp_i$ is a null value such that the tuple $X'_i(\perp_i)$ belongs to $\text{CAN}(I)$ and $h(\perp_i) = 1$; and
- $\sigma_{\bar{x}}(x_i) = 0$ if $\perp_i$ is a null value such that the tuple $X'_i(\perp_i)$ belongs to $\text{CAN}(I)$ and $h(\perp_i) = 0$.

Notice that this valuation is well defined since, as shown above, h assigns value either 0 or 1 to every null in $\text{CAN}(I)$. Assume, for the sake of contradiction, that ϕ is satisfiable. In particular, there must exist a valuation $\sigma_{\bar{z}}$ such that the valuation $\sigma = (\sigma_{\bar{x}}, \sigma_{\bar{z}})$ satisfies ϕ. We know that for each $k \in [1, \ell]$, the interpretation of the relation C'_k in $J^\star$ contains the seven tuples that represent a satisfying valuation for the k-th clause of ϕ. Then it is clear that for every $k \in [1, \ell]$, it holds that $J^\star$ contains the tuples $C'_k(\sigma(u^k_1), \sigma(u^k_2), \sigma(u^k_3))$, where u^k_1, u^k_2, u^k_3 are the propositional variables of ψ_k. We also know that $J^\star$ contains the tuples $Z'_j(\sigma(z_j))$ and $R'(a, \sigma(z_j))$, for every $j \in [1, m]$. Moreover, by the definition of $\sigma_{\bar{x}}$, we have that the tuples $X'_i(\sigma(x_i))$ and $R'(a, \sigma(x_i))$ also belong to $J^\star$, for every $i \in [1, n]$. It follows that $Q(J^\star) = \texttt{true}$, which is again a contradiction. This proves that ϕ is not satisfiable, and concludes the proof of the theorem. □

A natural question at this point is what happens with the complexity of the certain answers problem if one considers the entire class $\text{CQ}^{\neq}$. In the following theorem, we show that the same complexity bounds as in Theorem 6.5 and Proposition 6.6 hold in this case. Notice that the lower bounds in the following theorem follow from the lower bounds in these results.

Theorem 6.7 CERTAIN-ANSWERS(GLAV, $\text{CQ}^{\neq}$) *is* coNEXPTIME*-complete and* CERTAIN-ANSWERS(LAV, $\text{CQ}^{\neq}$) *is* Π^p_2*-complete.*

We conclude this section by pointing out that all the complexity bounds presented in this section remain the same if one allows unions of conjunctive queries with inequalities; if k-$\text{UCQ}^{\neq}$ is the class of unions of k-$\text{CQ}^{\neq}$ queries, then

Proposition 6.8

(1) CERTAIN-ANSWERS(GLAV, 1-$\text{UCQ}^{\neq}$) *is* EXPTIME*-complete*, CERTAIN-ANSWERS(LAV, 1-$\text{UCQ}^{\neq}$) *is* NP*-complete.*
(2) CERTAIN-ANSWERS(GLAV, k-$\text{UCQ}^{\neq}$) *is* coNEXPTIME*-complete, and* CERTAIN-ANSWERS(LAV, k-$\text{UCQ}^{\neq}$) *is* Π^p_2*-complete for every* $k \geq 2$.

(3) CERTAIN-ANSWERS(GLAV, UCQ$^{\neq}$) *is* coNEXPTIME*-complete, and* CERTAIN-ANSWERS(LAV, UCQ$^{\neq}$) *is* Π_2^p*-complete.*

7 Concluding Remarks

In this paper, we proposed the language DATALOG$^{\mathbf{C}(\neq)}$ that extends DATALOG with a restricted form of negation, and studied some of its fundamental properties. In particular, we showed that the certain answers to a DATALOG$^{\mathbf{C}(\neq)}$ program can be computed in polynomial time (in terms of data complexity), and we used this property to find tractable fragments of the class of unions of conjunctive queries with inequalities. In the paper, we also studied the combined complexity of computing certain answers to DATALOG$^{\mathbf{C}(\neq)}$ programs and other related query languages.

Many problems related to DATALOG$^{\mathbf{C}(\neq)}$ programs remain open. In particular, it would be interesting to know if it is decidable whether the certain answers to a query Q in UCQ$^{\neq}$ can be computed as the certain answers to a DATALOG$^{\mathbf{C}(\neq)}$ program Π_Q, and whether there exist a setting $\mathcal{M}$ and a query Q in UCQ$^{\neq}$ such that the problem CERTAIN-ANSWERS($\mathcal{M}$, Q) is in PTIME, but the certain answers to Q cannot be computed as the certain answers to a DATALOG$^{\mathbf{C}(\neq)}$ program Π_Q.

Acknowledgements We are very grateful to Jorge Pérez for many helpful discussions. The authors were supported by: Arenas—FONDECYT grants 1070732 and 1090565; Barceló—FONDECYT grant 11080011; Arenas and Barceló—grant P04-067-F from the Millennium Nucleus Centre for Web Research; Reutter—EPSRC grant G049165. Most of this work was done when Reutter was a Master's student at Pontificia Universidad Católica de Chile.

Appendix: Proofs and Intermediate Results

A.1 Proof of Lemma 5.3

We first prove a technical result that relates query answering and open-reachability.

Claim A.1 *Let Q and $\mathcal{M}$ be as defined above. Let I be an arbitrary source instance with canonical universal solution J, and let $\bar{t} = (t_1, \ldots, t_m)$ be a tuple of constants from I that also belong to J. Let $q = (p_1, p_2)$ and $q' = (p'_1, p'_2)$ be two semi-open nodes in $H(Q, J, \bar{t})$ such that the constant components of q and q' are different, and that q' is openly-reachable from q. Then, for every solution J^* of I and homomorphism h from J to J^*, if $h(p_1) = h(p_2)$ and $h(p'_1) = h(p'_2)$, then $J^* \models Q(\bar{t})$.*

Proof Assume that $Q(\bar{x}) = Q_1(\bar{x}) \vee \cdots \vee Q_l(\bar{x})$, where $\bar{x} = x_1, \ldots, x_m$. Let $q = (p_1, p_2)$ and $q' = (p'_1, p'_2)$ be two semi-open nodes in $H(Q, J, \bar{t})$ such that q and q' have different constant components, and that q' is openly-reachable from q. Let us fix a solution J^* for I and assume there is a homomorphism h from J to J^* such that $h(p_1) = h(p_2)$ and $h(p'_1) = h(p'_2)$. Since q and q' are openly-reachable, there exists a path $qq_1 \cdots q_k q'$ in $H(Q, J, \bar{t})$ such that every node q_i, $1 \leq i \leq k$, is open and has a `green`-labeled loop. Notice that in this particular case there must be at least one

extra node in the path, that is, $k \geq 1$. Assume otherwise. Then q and q' are red-adjacent (share its null value) and have different constant components. It follows that either $h(p_1) \neq h(p_2)$ or $h(p_1') \neq h(p_2')$, which is a contradiction.

Notice also that there must be at least one node $q_j = (p_1^j, p_2^j)$, $1 \leq j \leq k$, such that $h(p_1^j) \neq h(p_2^j)$. This can be proved using a similar argument to the one in the previous paragraph. From the definition, q_j has a green-labeled loop, that is, there exists some $1 \leq i \leq l$ such that $Q_i(\bar{x})$ is a conjunctive query with one inequality of the form $\exists \bar{y}(\phi(\bar{x}, \bar{y}) \wedge u_1 \neq u_2)$, and an assignment $\sigma : \{\bar{x}, \bar{y}\} \to \mathrm{dom}(J)$, such that $\sigma(\bar{x}) = \bar{t}$, $(J, \sigma) \models \phi(\bar{x}, \bar{y}) \wedge u_1 \neq u_2$, $\sigma(u_1) = p_1^j$, $\sigma(u_2) = p_2^j$. We obtain that $(J^*, \sigma) \models \phi(\bar{x}, \bar{y}) \wedge u_1 \neq u_2$ from the fact that conjunctive queries are preserved under homomorphisms, and that $h(p_1^j) \neq h(p_2^j)$. Thus, we obtain that $J \models Q_i(\bar{t})$, and therefore $J^* \models Q(\bar{t})$. □

We now continue with the proof of Lemma 5.3. We first prove the 'if' direction. Let J^* be an arbitrary solution for I, and h a homomorphism from J to J^*. In particular, $h(\bar{t}) = \bar{t}$. Assume first that μ and ν are blue-adjacent in $H(Q, J, \bar{t})$. Then for some $i \in [1, \ell]$, $Q_i(\bar{x})$ is of the form $\exists \bar{y} \phi(\bar{x}, \bar{y})$, where $\phi(\bar{x}, \bar{y})$ is a conjunction of relational atoms over $\mathbf{T}$, and $J \models Q_i(\bar{t})$. It follows that $J^* \models Q_i(h(\bar{t}))$, since conjunctive queries are preserved under homomorphisms, and, thus, that $J^* \models Q_i(\bar{t})$, because $h(\bar{t}) = \bar{t}$. Therefore, $J^* \models Q(\bar{t})$, and $\bar{t} \in \mathrm{certain}_{\mathcal{M}}(Q, I)$, because J^* was arbitrarily chosen.

Assume now that μ and ν are not blue-adjacent in $H(Q, J, \bar{t})$, but that there are two nodes $q = (p_1, p_2)$ and $q' = (p_3, p_4)$ in $H(Q, J, \bar{t})$ such that q and q' are blue-adjacent and both q and q' have c-paths in $H(Q, J, \bar{t})$. Since q and q' are blue adjacent, for some $1 \leq i \leq \ell$, (1) $Q_i(\bar{x})$ is of the form $\exists \bar{y}(\phi(\bar{x}, \bar{y}) \wedge u_1 \neq u_2 \wedge v_1 \neq v_2)$, where $\phi(\bar{x}, \bar{y})$ is a conjunction of relational atoms over $\mathbf{T}$, and $u_1, u_2, v_1, v_2 \in \{\bar{x}, \bar{y}\}$, and (2) there is an assignment $\sigma : \{\bar{x}, \bar{y}\} \to \mathrm{dom}(J)$, such that $\sigma(\bar{x}) = \bar{t}$, $(J, \sigma) \models \phi(\bar{x}, \bar{y}) \wedge u_1 \neq u_2 \wedge v_1 \neq v_2$, $\sigma(u_1) = p_1$, $\sigma(u_2) = p_2$, $\sigma(v_1) = p_3$, and $\sigma(v_2) = p_4$.

Notice that if $h(\sigma(u_1)) \neq h(\sigma(u_2))$ and $h(\sigma(v_1)) \neq h(\sigma(v_2))$, then $J^* \models Q_i(\bar{t})$ (because conjunctive queries are preserved under homomorphisms), and, therefore, $J^* \models Q(\bar{t})$ and $\bar{t} \in \mathrm{certain}_{\mathcal{M}}(Q, I)$ (because J^* was arbitrarily chosen). So, assume otherwise that $h(\sigma(u_1)) = h(\sigma(u_2))$ (the case when $h(\sigma(v_1)) = h(\sigma(v_2))$ is completely symmetrical). Let $q = q_1 q_2 \cdots q_{2k+1}$ be a c-path in $H(Q, J, \bar{t})$, $k \geq 0$. There are two cases to consider:

- For some $0 < j \leq k - 1$, $q_{2j+1} = (p_1^{2j+1}, p_2^{2j+1})$ is such that $h(p_1^{2j+1}) \neq h(p_2^{2j+1})$: Assume without loss of generality that for every $0 \leq j' < j$, $q_{2j'+1} = (p_1^{2j'+1}, p_2^{2j'+1})$ is such that $h(p_1^{2j'+1}) = h(p_2^{2j'+1})$. Since for every $0 < s \leq j$ it is the case that $q_{2s} = (p_1^{2s}, p_2^{2s})$ is openly-reachable from $q_{2s-1} = (p_1^{2s-1}, p_2^{2s-1})$, and the constant components of q_{2s} and q_{2s-1} are different, if for some $0 < s \leq j$ it holds that $h(p_1^{2s}) = h(p_2^{2s})$, then from Claim A.1 we obtain that $J^* \models Q(\bar{t})$ (and thus $\bar{t} \in \mathrm{certain}_{\mathcal{M}}(Q, I)$ because J^* was arbitrarily chosen). Thus, assume that $h(p_1^{2s}) \neq h(p_2^{2s})$ for every $0 < s \leq j$. In particular, $h(p_1^{2j}) \neq h(p_2^{2j})$.

  Assume first that q_{2j} has a green-labeled loop (the case when q_{2j+1} has a green-labeled loop is completely symmetrical). Then, it must be the case that for

some $1 \leq i' \leq \ell$, (1) $Q_i(\bar{x})$ is of the form $\exists \bar{z}(\psi(\bar{x}, \bar{z}) \wedge w_1 \neq w_2)$, where $\psi(\bar{x}, \bar{z})$ is a conjunction of relational atoms over $\mathbf{T}$, and $w_1, w_2 \in \{\bar{x}, \bar{z}\}$, and (2) there is an assignment $\sigma' : \{\bar{x}, \bar{z}\} \to \text{dom}(J)$, such that $\sigma'(\bar{x}) = \bar{t}$, $(J, \sigma) \models \phi(\bar{x}, \bar{z}) \wedge w_1 \neq w_2$, $\sigma(w_1) = p_1^{2j}$, $\sigma(w_2) = p_2^{2j}$. Because (1) conjunctive queries are preserved under homomorphisms, (2) $h(\sigma'(w_1)) = h(p_1^{2j}) \neq h(p_2^{2j}) = h(\sigma'(w_2))$, it is the case $J^* \models \psi(h(\bar{t}), h(\sigma'(\bar{z}))) \wedge h(\sigma'(w_1)) \neq h(\sigma'(w_2))$. It follows that $J^* \models \exists \bar{z}(\psi(\bar{t}, \bar{z}) \wedge w_1 \neq w_2)$, because $h(\bar{t}) = \bar{t}$. We conclude that $J^* \models Q(\bar{t})$, and thus that $\bar{t} \in \text{certain}_{\mathcal{M}}(Q, I)$, since J^* was arbitrarily chosen.

Assume now that q_{2j} does not have a `green`-labeled loop. Thus, q_{2j} and q_{2j+1} are `blue`-adjacent. Therefore, it must be the case that for some $i' \in [1, \ell]$, (1) $Q_{i'}(\bar{x})$ is of the form $\exists \bar{z}(\psi(\bar{x}, \bar{z}) \wedge w_1 \neq w_2 \wedge w_3 \neq w_4)$, where $\psi(\bar{x}, \bar{z})$ is a conjunction of relational atoms over $\mathbf{T}$, and $w_1, w_2, w_3, w_4 \in \{\bar{x}, \bar{z}\}$, and (2) there is an assignment $\sigma' : \{\bar{x}, \bar{z}\} \to \text{dom}(J)$, such that $\sigma'(\bar{x}) = \bar{t}$, $(J, \sigma') \models \psi(\bar{x}, \bar{z}) \wedge w_1 \neq w_2 \wedge w_3 \neq w_4, \sigma'(w_1) = p_1^{2j}, \sigma'(w_2) = p_2^{2j}, \sigma'(w_3) = p_1^{2j+1}$, and $\sigma'(w_4) = p_2^{2j+1}$. Because (1) conjunctive queries are preserved under homomorphisms, (2) $h(\sigma'(w_1)) = h(p_1^{2j}) \neq h(p_2^{2j}) = h(\sigma'(w_2))$, and (3) $h(\sigma'(w_3)) = h(p_1^{2j+1}) \neq h(p_2^{2j+1}) = h(\sigma'(w_4))$, it is the case $J^* \models \psi(h(\bar{t}), h(\sigma'(\bar{z}))) \wedge h(\sigma'(w_1)) \neq h(\sigma'(w_2)) \wedge h(\sigma'(w_3)) \neq h(\sigma'(w_4))$. It follows that $J^* \models \exists \bar{z}(\psi(\bar{t}, \bar{z}) \wedge w_1 \neq w_2 \wedge w_3 \neq w_4)$, because $h(\bar{t}) = \bar{t}$. We conclude that $J \models Q(\bar{t})$, and, thus, that $\bar{t} \in \text{certain}_{\mathcal{M}}(Q, I)$, since J^* was arbitrarily chosen.

- For every $0 \leq j \leq k - 1$, $q_{2j+1} = (p_1^{2j+1}, p_2^{2j+1})$ is such that $h(p_1^{2j+1}) = h(p_2^{2j+1})$: Since for every $0 < s \leq k$, it is the case that $q_{2s} = (p_1^{2s}, p_2^{2s})$ is openly-reachable from $q_{2s-1} = (p_1^{2s-1}, p_2^{2s-1})$, and the constant components of q_{2s} and q_{2s-1} are different, from Claim A.1 if for some $0 < s \leq k$ it holds that $h(p_1^{2s}) = h(p_2^{2s})$, then $J^* \models Q(\bar{t})$, and, since J^* was arbitrarily chosen, we prove that $\bar{t} \in \text{certain}_{\mathcal{M}}(Q, I)$. Thus, assume that $h(p_1^{2k}) \neq h(p_2^{2k})$.

  Suppose first that q_{2k+1} has two constant components. Then by definition of c-path it contains two different constants, and thus, it must be the case that $h(p_1^{2k+1}) \neq h(p_2^{2k+1})$. Since either $q_{2k} = q_{2k+1}$ and q_{2k} has a `green`-labeled loop or q_{2k+1} and q_{2k} are `blue`-adjacent in $H(Q, J, \bar{t})$, $h(p_1^{2k+1}) \neq h(p_2^{2k+1})$. One can then follow the same reasoning than in the previous item, and show that $J^* \models Q(\bar{t})$, and, thus, that $\bar{t} \in \text{certain}_{\mathcal{M}}(Q, I)$.

  Suppose, on the other hand, that $q_{2k+1} = (p_1^{2k+1}, p_1^{2k+2})$ is semi-open. By definition of c-path, q_{2k+1} is openly-reachable from $q_{2j-1} = (p_1^{2j-1}, p_2^{2j-1})$, for some $1 \leq j \leq k$. Since $h(p_1^{2j-1}) = h(p_2^{2j-1})$ and the constant components of q_{2k+1} and q_{2j-1} are different, we assume again that $h(p_1^{2k+1}) \neq h(p_2^{2k+1})$. If not, by Claim A.1, we obtain that $J^* \models Q(\bar{t})$, and thus $\bar{t} \in \text{certain}_{\mathcal{M}}(Q, I)$, because J^* was arbitrarily chosen. The rest of the proof follows using the same kind of reasoning than in the previous item.

Finally, assume that there is a node $q = (p_1, p_2)$ in $H(Q, J, \bar{t})$ such that q has a `green`-labeled loop and a c-path in $H(Q, J, \bar{t})$. Thus, for some $1 \leq i \leq \ell$, (1) $Q_i(\bar{x})$ is of the form $\exists \bar{y}(\phi(\bar{x}, \bar{y}) \wedge u_1 \neq u_2)$, where $\phi(\bar{x}, \bar{y})$ is a conjunction of relational atoms over $\mathbf{T}$, and $u_1, u_2 \in \{\bar{x}, \bar{y}\}$, and (2) there is an assignment

$\sigma : \{\bar{x}, \bar{y}\} \to \text{dom}(J)$, such that $\sigma(\bar{x}) = \bar{t}$, $(J, \sigma) \models \phi(\bar{x}, \bar{y}) \wedge u_1 \neq u_2$, $\sigma(u_1) = p_1$, $\sigma(u_2) = p_2$.

Notice that if $h(\sigma(u_1)) \neq h(\sigma(u_2))$, then $J^* \models Q_i(\bar{t})$ (because conjunctive queries are preserved under homomorphisms), and, since J^* was arbitrarily chosen, it follows that $\bar{t} \in \text{certain}_{\mathcal{M}}(Q, I)$. So we assume that $h(\sigma(u_1)) \neq h(\sigma(u_2))$. Since q is marked and has a c-path, using the same argument that in the previous paragraphs it is possible to show that $J^* \models Q(\bar{t})$, and, thus, that $\bar{t} \in \text{certain}_{\mathcal{M}}(Q, I)$.

Now we prove the 'only if' direction. Let I be an arbitrary source instance and J its canonical universal solution. We prove that if there is no `blue` edge between μ and ν in $H(Q, J, \bar{t})$, there is no node q with a `green`-labeled loop that has a c-path, and there are no `blue`-adjacent nodes q and q' such that both q and q' have c-paths in $H(Q, J, \bar{t})$, then $\bar{t} \notin \text{certain}_{\mathcal{M}}(Q, I)$. We prove this by building a solution J' of I such that $J' \not\models Q(\bar{t})$.

Let us say that a node has a *red-blue*-path in $H(Q, J, \bar{t})$, if it has a c-path but without the restriction on the last element of the path. That is, we say that the node q has a *red-blue* path (rb-path) in $H(Q, J, \bar{t})$ if there is a path $q = q_1 q_2 \cdots q_{2k+1}$ in $H(Q, J, \bar{t})$, $k \geq 0$, that satisfies the following:

- Every node q_i, $1 \leq i \leq 2k$, is semi-open;
- every node of the form q_{2i+1}, $0 \leq i \leq k-1$, is openly-reachable from q_{2i+2}, but the constant components in q_{2i+1} and q_{2i+2} are different; and
- every node of the form q_{2i}, $0 < i \leq k$, either is `blue`-adjacent to q_{2i+1} or $q_{2i} = q_{2i+1}$ and q_{2i} has a `green`-labeled loop.

Further, we say that a node q has an *open-null* path (*on*-path), if q is openly-reachable from an open node q' in $H(Q, J, \bar{t})$.

We define a procedure that does the following. Let

$$T = \{(p_0^1, p_0^2, p_0^3, p_0^4), (p_1^1, p_1^2, p_1^3, p_1^4), \ldots, (p_n^1, p_n^2, p_n^3, p_n^4)\}$$

be the maximal set of tuples (of length 4) of elements in J that satisfies the following: For each tuple $(p_j^1, p_j^2, p_j^3, p_j^4) \in T$, $0 \leq j \leq n$,

- there exists $i \in [1, \ell]$ such that (1) $Q_i(\bar{x})$ is of the form $\exists \bar{y}(\phi(\bar{x}, \bar{y}) \wedge u_1 \neq u_2 \wedge v_1 \neq v_2)$, where $\phi(\bar{x}, \bar{y})$ is a conjunction of relational atoms over $\mathbf{T}$, and $u_1, u_2, v_1, v_2 \in \{\bar{x}, \bar{y}\}$, and (2) there is an assignment $\sigma : \{\bar{x}, \bar{y}\} \to \text{dom}(J)$, such that $\sigma(\bar{x}) = \bar{t}$, $\sigma(u_1) = p_j^1$, $\sigma(u_2) = p_j^2$, $\sigma(v_1) = p_j^3$, $\sigma(v_2) = p_j^4$, and

$$(J, \sigma) \models \phi(\bar{x}, \bar{y}) \wedge u_1 \neq u_2 \wedge v_1 \neq v_2,$$

  or
- it is the case that $t_j^1 = t_j^3$, and $t_j^2 = t_j^4$, and there exists $i \in [1, \ell]$ such that (1) $Q_i(\bar{x})$ is of the form $\exists \bar{y}(\phi(\bar{x}, \bar{y}) \wedge u_1 \neq u_2)$, where $\phi(\bar{x}, \bar{y})$ is a conjunction of relational atoms over $\mathbf{T}$, and $u_1, u_2 \in \{\bar{x}, \bar{y}\}$, and (2) there is an assignment $\sigma : \{\bar{x}, \bar{y}\} \to \text{dom}(J)$, such that $\sigma(\bar{x}) = \bar{t}$, $\sigma(u_1) = p_j^1$, $\sigma(u_2) = p_j^2$, and

$$(J, \sigma) \models \phi(\bar{x}, \bar{y}) \wedge u_1 \neq u_2,$$

  and (3) t_j^1 or t_j^2 is a constant.

In any of these cases, we say that the tuple $(p_j^1, p_j^2, p_j^3, p_j^4)$, $0 \leq j \leq n$, *witnesses* the adjacency of the nodes (p_j^1, p_j^2) and (p_j^3, p_j^4) in $H(Q, J, \bar{t})$. Notice that it is possible that two different tuples in T witness the adjacency of the same pair of nodes.

The procedure first defines an arbitrary linear order over T, and then repeat the following step until all tuples in T have been marked: It first takes the least tuple in the order that has not yet been marked, let us say $(p_j^1, p_j^2, p_j^3, p_j^4)$, $0 \leq j \leq n$. Then chooses (nondeterministically) a node $q \in H(Q, J, \bar{t})$ that does not have a c-path, and whose adjacency to some node $q' \in H(Q, J, \bar{t})$ is witnessed by $(p_j^1, p_j^2, p_j^3, p_j^4)$. This node exists because, by hypothesis, it cannot be the case that both (p_j^1, p_j^2) and (p_j^3, p_j^4) have c-paths in $H(Q, J, \bar{t})$, since either (p_j^1, p_j^2) has a `green`-labeled loop or (p_j^1, p_j^2) and (p_j^3, p_j^4) are `blue`-adjacent in $H(Q, J, \bar{t})$. Further, notice that by definition, q has to be semi-open. The procedure then marks $(p_j^1, p_j^2, p_j^3, p_j^4)$ and does the following for each rb-path $q = q_1 q_2 \cdots q_{2k+1}$ in $H(Q, J, \bar{t})$:

(i) For every $i \in [0, k]$, if the node q_{2i+1} contains the null $\perp_{2i+1}$ and the constant c_{2i+1}, it assigns to $\perp_{2i+1}$ the value c_{2i+1}. It also assigns the value c_{2i+1} to every component of every node that belongs to an *on*-path starting from q_{2i+1}; and
(ii) for every $i \in [1, k]$, the procedure marks each tuple $(p_r^1, p_r^2, p_r^3, p_r^4)$, $1 \leq r \leq n$, that witnesses the adjacency of the nodes q_{2i} and q_{2i+1} in $H(Q, J, \bar{t})$.

In this case we say that the tuple $(p_j^1, p_j^2, p_j^3, p_j^4)$ and the node q *initialize* this step of the procedure.

Claim A.2 *The procedure described above assigns at most one constant to each null $\perp$ in J.*

Proof First, it cannot be the case that a null $\perp$ is assigned different constants in steps i and j of the algorithm with $i < j$. Assume otherwise, and let q be the node that initializes step i of the procedure, and let $(p_r^1, p_r^2, p_r^3, p_r^4)$, $0 \leq r \leq n$, and q' be the tuple and node, respectively, that initialize step j of the procedure. Then either:

a. There are rb-paths $q \cdots q_t$ and $q' \cdots q'_t$ such that both q_t and q'_t have null $\perp$ as a component, but q_t and q'_t have different constant components. Notice that q' cannot have a `green`-labeled loop; otherwise, $q \cdots q_t q'_t \cdots q' q'$ would be a c-path, and therefore q could not have initialized step i of the algorithm. Assume, thus, that there is a node q'' in $H(Q, J, \bar{t})$, that is `blue`-adjacent to q', and that the `blue`-adjacency of q' to q'' is witnessed by $(p_r^1, p_r^2, p_r^3, p_r^4)$. But then $q \cdots q_t q'_t \cdots q' q''$ is also an rb-path in $H(Q, J, \bar{t})$, and, therefore, the tuple $(p_r^1, p_r^2, p_r^3, p_r^4)$ would have been marked in step i of the procedure, which is a contradiction; or
b. there are rb-paths $q \cdots q_{2k+1}$ and $q' \cdots q'_{2k'+1}$, and *on*-paths $q_{2j+1} \cdots q_s$ and $q'_{2j'+1} \cdots q_t$, $0 \leq j \leq k$ and $0 \leq j' \leq k'$, such that both q_s and q_t have null $\perp$ as a component and the constant components of q_{2j+1} and $q'_{2j'+1}$ are different. But then q_{2j+1} is openly-reachable from $q'_{2j'+1}$, and, thus, either q' has a `green`-labeled loop and $q \cdots q_{2j+1} q'_{2j'+1} \cdots q' q'$ is a c-path, or there is a node q'' that is `blue` adjacent to q' and such that $q \cdots q_{2j+1} q'_{2j'+1} \cdots q' q''$ is an rb-path. Following the same reasoning that in the last item one can see that this is a contradiction.

We now prove that each step i of the algorithm makes at most one assignment to each null $\perp$ in J. Assume otherwise, and let q be the node of $H(Q, J, \bar{t})$ that initializes step i. Then either:

a. There are rb-paths $q \cdots q_t$ and $qq_1 \cdots q_t'' q_t'$, where both q_t and q_t' have null $\perp$ as a component, but q_t and q_t' have different constant components. But then $q \cdots q_t q_t' q_t'' \cdots q_1$ is also an rb-path in $H(Q, J, \bar{t})$, and both q and q_1 have the same null component but a different constant component. This shows that $q \cdots q_t q_t' q_t'' \cdots q_1$ is a c-path in $H(Q, J, \bar{t})$, which is a contradiction; or
b. there are rb-paths $q \cdots q_{2k+1}$ and $qq_1' \cdots q_{2k'+1}'$, and on-paths $q_{2j+1} \cdots q_s$ and $q_{2j'+1}' \cdots q_t$, $0 \leq j \leq k$ and $0 \leq j' \leq k'$, such that both q_s and q_t have null $\perp$ as a component and the constant components of q_{2j+1} and $q_{2j'+1}'$ are different. But then q_{2j+1} is openly-reachable from $q_{2j'+1}'$, and, thus, $q \cdots q_{2j+1} q_{2j'+1}' \cdots q_1'$ is also an rb-path in $H(q, J, \bar{t})$, and both q and q_1' have the same null component but a different constant component. We conclude that $q \cdots q_{2j+1} q_{2j'+1}' \cdots q_1'$ is a c-path in $H(Q, J, \bar{t})$, which is a contradiction.

This finishes the proof of the claim. □

Clearly, the procedure finishes after a finite number of steps and marks every tuple in T. Further, for every tuple $(p_r^1, p_r^2, p_r^3, p_r^4)$ in T, $0 \leq r \leq n$, it is the case that at least one of the nodes (p_r^1, p_r^2) and (p_r^3, p_r^4) is semi-open and the procedure assigns the constant component of such node to the null component.

We construct a solution J' from J as follows: Take the canonical solution J. For each null $\perp$ of J, if the procedure assigns the constant c to $\perp$, then replace all appearances of $\perp$ in J by c. Next, choose a fresh constant value c' that has not been used in J, and replace every null value $\perp'$ that is not assigned a constant by the procedure by the fresh constant c'. In the following we show that $J' \not\models Q(\bar{t})$, and, thus, that $\bar{t} \notin \text{certain}_{\mathcal{M}}(Q, I)$.

Assume otherwise. Then there exists $i \in [1, \ell]$ such that $J' \models Q_i(\bar{t})$. We analyze three cases.

(1) Assume first that Q_i is of the form $\exists \bar{y}_1, \ldots, \bar{y}_s (T_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge T_s(\bar{x}_s, \bar{y}_s) \wedge u_1 \neq u_2 \wedge v_1 \neq v_2)$, where $\{T_1, \ldots, T_s\} \subseteq \mathbf{T}$, $\bar{x} = \{\bar{x}_1\} \cup \cdots \cup \{\bar{x}_s\}$, and $u_1, u_2, v_1, v_2 \in \{\bar{x}, \bar{y}_1, \ldots, \bar{y}_s\}$. Thus, there exist tuples $\bar{p}_1, \ldots, \bar{p}_s$ of elements in J' and an assignment $\sigma : \{\bar{x}\} \cup \{\bar{y}_1\} \cup \cdots \cup \{\bar{y}_s\} \to \text{dom}(J')$ defined by $\sigma(\bar{x}) = \bar{t}$ and $\sigma(\bar{y}_j) = \bar{p}_j$, for every $1 \leq j \leq s$, such that

$$(J', \sigma) \models T_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge T_s(\bar{x}_s, \bar{y}_s) \wedge u_1 \neq u_2 \wedge v_1 \neq v_2.$$

In particular, $\sigma(u_1) \neq \sigma(u_2)$ and $\sigma(v_1) \neq \sigma(v_2)$.

For every $j \in [1, s]$, let us denote by $\bar{t}_j$ the value of $\sigma(\bar{x}_j)$. By definition of J', every tuple $(\bar{t}_j, \bar{p}_j) \in T_j^{J'}$ $(1 \leq j \leq s)$ is obtained from a tuple $(\bar{t}_j, \bar{r}_j) \in T_j^J$ by replacing each null value $\perp \in \bar{r}_j$ with the constant c, if the procedure assigned c to $\perp$, and every other null value $\perp'$ with the fresh constant c'. Let us define an assignment $\sigma' : \{\bar{x}\} \cup \{\bar{y}_1\} \cup \cdots \cup \{\bar{y}_s\} \to \text{dom}(J)$ as follows: $\sigma'(\bar{x}) = \bar{t}$ and $\sigma'(\bar{y}_j) = \bar{r}_j$, for each $1 \leq j \leq s$. We show that σ' is well-defined. Assume that z is a variable that appears in at least two different positions in $(\bar{y}_1, \ldots, \bar{y}_s)$. We show that σ' assigns

the same value to each appearance of z. Indeed, that z appears in two different positions in $(\bar{y}_1, \ldots, \bar{y}_s)$ implies that z is a join variable in $T_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge T_s(\bar{x}_s, \bar{y}_s)$. By hypothesis, z cannot be nullified under Q_i and $\mathcal{M}$. Thus, $\sigma(z)$ is a constant and $\sigma(z) = \sigma'(z)$ (because all the witnesses for z in J must be constants). It immediately follows that every appearance of z in $(\bar{y}_1, \ldots, \bar{y}_s)$ is assigned the same value by σ'.

Therefore, $(J, \sigma') \models T_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge T_s(\bar{x}_s, \bar{y}_s)$. If $\sigma'(u_1) = \sigma'(u_2)$ or $\sigma'(v_1) = \sigma'(v_2)$, then $\sigma(u_1) = \sigma(u_2)$ or $\sigma(v_1) = \sigma(v_2)$, which is a contradiction. Assume then that $\sigma'(u_1) \neq \sigma'(u_2)$ and $\sigma'(v_1) \neq \sigma'(v_2)$. Therefore,

$$(J, \sigma') \models T_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge T_s(\bar{x}_s, \bar{y}_s) \wedge u_1 \neq u_2 \wedge v_1 \neq v_2.$$

Then the tuple $(\sigma'(u_1), \sigma'(u_2), \sigma'(v_1), \sigma'(v_2))$ belongs to T, and since at least one of the nodes $(\sigma'(u_1), \sigma'(u_2))$ and $(\sigma'(v_1), \sigma'(v_2))$ is semi-open and the procedure assigns the constant component of such node to the null component, it must be the case that $\sigma'(u_1) = \sigma'(u_2)$ or $\sigma'(v_1) = \sigma'(v_2)$. This is a contradiction.

(2) Assume second that Q_i is of the form $\exists \bar{y}_1, \ldots, \bar{y}_s (T_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge T_s(\bar{x}_s, \bar{y}_s) \wedge u_1 \neq u_2)$, where $\{T_1, \ldots, T_s\} \subseteq \mathbf{T}$, $\bar{x} = \{\bar{x}_1\} \cup \cdots \cup \{\bar{x}_s\}$, and $u_1, u_2 \in \{\bar{x}, \bar{y}_1, \ldots, \bar{y}_s\}$. Thus, there exist tuples $\bar{p}_1, \ldots, \bar{p}_s$ of elements in J' and an assignment $\sigma : \{\bar{x}\} \cup \{\bar{y}_1\} \cup \cdots \cup \{\bar{y}_s\} \to \mathrm{dom}(J')$ defined by $\sigma(\bar{x}) = \bar{t}$ and $\sigma(\bar{y}_j) = \bar{p}_j$, for every $1 \leq j \leq s$, such that

$$(J', \sigma) \models T_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge T_s(\bar{x}_s, \bar{y}_s) \wedge u_1 \neq u_2.$$

In particular, $\sigma(u_1) \neq \sigma(u_2)$.

Following the same reasoning it is possible to prove that $(J, \sigma') \models T_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge T_s(\bar{x}_s, \bar{y}_s)$. If $\sigma'(u_1) = \sigma'(u_2)$, then $\sigma(u_1) = \sigma(u_2)$, which is a contradiction. Assume then that $\sigma'(u_1) \neq \sigma'(u_2)$. As for the previous case, we obtain that

$$(J, \sigma') \models T_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge T_s(\bar{x}_s, \bar{y}_s) \wedge u_1 \neq u_2.$$

From the construction of $H(Q, J, \bar{t})$, it must then be the case that the node $(\sigma'(u_1), \sigma'(u_2))$ has a `green`-labeled loop. Further, the node $(\sigma'(u_1), \sigma'(u_2))$ must be semi-open. Assume otherwise. Clearly $(\sigma'(u_1), \sigma'(u_2))$ cannot consists of two distinct constants, as otherwise there would exist a c-path starting on this node. Thus, it must consist of two null values. If the procedure assigns a constant neither to $\sigma'(u_1)$ nor to $\sigma'(u_2)$, then $\sigma(u_1) = \sigma(u_2)$, which is a contradiction. Then, it must be the case that the procedure assigns the constant c to at least one of $\sigma'(u_1)$ or $\sigma'(u_2)$. But then the node $(\sigma'(u_1), \sigma'(u_2))$ is part of an *on*-path, and therefore both $\sigma'(u_1)$ and $\sigma'(u_2)$ must have been assigned the same constant, and thus $\sigma(u_1) = \sigma(u_2)$, which is a contradiction. We conclude that $(\sigma'(u_1), \sigma'(u_2), \sigma'(u_1), \sigma'(u_2))$ belongs to T, and, thus, the procedure assigns the constant component of such node to the null component. Therefore, it must be the case that $\sigma'(u_1) = \sigma'(u_2)$, which is a contradiction.

(3) Assume finally that Q_i is of the form $\exists \bar{y}_1, \ldots, \bar{y}_n (T_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge T_n(\bar{x}_n, \bar{y}_n))$, where $\{T_1, \ldots, T_n\} \subseteq \mathbf{T}$ and $\bar{x} = \{\bar{x}_1\} \cup \cdots \cup \{\bar{x}_n\}$. Following the same reasoning it is possible to show that $(J, \sigma') \models T_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge T_n(\bar{x}_n, \bar{y}_n)$. Thus, $J \models Q_i(\bar{t})$, which implies that there is an edge between μ and ν in $G(Q, J, \bar{t})$. This is again a contradiction.

A.2 Proof of Theorem 5.7

We now present the proof for the second and third assertions of Theorem 5.7.

We first prove part (2). That is, we prove that there is a LAV data exchange setting $\mathcal{M}$ and a conjunctive query Q with two inequalities, such that Q has constant joins but does not have almost constant inequalities under $\mathcal{M}$, and CERTAIN-ANSWERS($\mathcal{M}$, Q) is coNP-complete.

The LAV setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ is as follows. The source schema $\mathbf{S}$ consists of one ternary relation symbol M, one binary relation symbol N, and one unary relation symbol U. The target schema $\mathbf{T}$ consists of three relation symbols: One ternary relation P, and two binary relations R and S. Further, Σ_{st} is the following set of source-to-target dependencies:

$$\begin{aligned} M(x, y, z) &\rightarrow P(x, y, z), \\ N(x, y) &\rightarrow \exists z \exists u (R(x, z) \wedge R(y, u) \wedge S(x, u)), \\ U(x) &\rightarrow S(x, x). \end{aligned}$$

The Boolean query Q is as follows:

$$\exists x_1 \exists y_1 \exists x_2 \exists y_2 \exists x_3 \exists y_3 (P(x_1, x_2, x_3) \wedge R(x_1, y_1) \wedge S(x_2, y_2) \wedge R(x_3, y_3) \wedge y_1 \neq y_2 \wedge y_2 \neq y_3).$$

Clearly, Q has constant joins, but does not have constant inequalities in $\mathcal{M}$. We prove next that the problem CERTAIN-ANSWERS($\mathcal{M}$, Q) is coNP-complete.

Membership in coNP follows from [9]. The coNP-hardness is established from a reduction from POSITIVE-NOT-ALL-EQUAL-3SAT, which is the following decision problem: Given a propositional formula ϕ in 3CNF consisting entirely of positive clauses $(p \vee q \vee r)$, is there a valuation to the propositional variables of ϕ such that for every clause of ϕ at least one variable is assigned value 1 and at least one variable is assigned value 0? This problem is known to be NP-hard (see e.g. the proof of Theorem 5.11 in [9]). More precisely, for every 3CNF propositional formula ϕ consisting entirely of positive clauses, we construct in polynomial time an instance I_ϕ of $\mathbf{S}$ such that ϕ is NOT-ALL-EQUAL-satisfiable iff $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$.

Given a propositional formula $\phi \equiv \bigwedge_{1 \leq j \leq m} C_j$ in 3CNF, where each C_j is a clause consisting entirely of positive literals, let I_ϕ be the following source instance, where 1 and 0 are constants not mentioned in ϕ:

- The interpretation of M in I_ϕ contains the tuples $(q, 1, \hat{q})$ and $(q, 0, \hat{q})$, for each propositional variable q mentioned in ϕ, and contains the tuple (p, q, r) if for some $j \in [1, m]$, $C_j = (p \vee q \vee r)$;
- the interpretation of N in I_ϕ contains the tuple $(q, \hat{q})$, for each propositional variable mentioned in ϕ; and
- the interpretation of U in I_ϕ contains the elements 0 and 1.

Clearly, I_ϕ can be constructed in polynomial from ϕ.

The canonical universal solution J of I_ϕ is as follows, where we denote by $\perp_q$ and $\#_q$ the nulls that are generated in order to witness variables z and u, respectively, when applying the std $N(x, y) \rightarrow \exists z \exists u (R(x, z) \wedge R(y, u) \wedge S(x, u))$ to $N(q, \hat{q})$:

- The interpretation of the relation P in J is just a copy of the interpretation of the relation M in I_ϕ;
- the interpretation of the relation R in J contains the pairs $(q, \bot_q)$ and $(\hat{q}, \#_q)$, for each propositional variable q mentioned in ϕ; and
- the interpretation of the relation S in J contains the pair $(q, \#_q)$, for each propositional variable q mentioned in ϕ, and also contains the pairs $(1, 1)$ and $(0, 0)$.

We prove next that ϕ is NOT-ALL-EQUAL satisfiable iff $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$.

($\Rightarrow$) Assume first that ϕ is NOT-ALL-EQUAL-satisfiable, and let κ be a truth assignment for the propositional variables mentioned in ϕ, such that for every clause $(p \vee q \vee r)$ in ϕ, it is the case that $\kappa(p) = 1$ or $\kappa(q) = 1$ or $\kappa(r) = 1$, and $\kappa(p) = 0$ or $\kappa(q) = 0$ or $\kappa(r) = 0$. From κ we construct a function f from $\text{dom}(J)$ into $\text{dom}(J)$ as follows:

$$f(v) = \begin{cases} 1 & v = \bot_q \text{ and } \kappa(q) = 1, \\ 0 & v = \bot_q \text{ and } \kappa(q) = 0, \\ 1 & v = \#_q \text{ and } \kappa(q) = 0, \\ 0 & v = \#_q \text{ and } \kappa(q) = 1, \\ v & \text{otherwise.} \end{cases}$$

Let J^* be the solution for I_ϕ obtained from J by replacing each occurrence of an element v in J by $f(v)$. We show next that $Q(J^*) = \texttt{false}$, and, thus, that $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$.

Assume, for the sake of contradiction, that $Q(J^*) = \texttt{true}$. Then there is a function $h : \{x_1, x_2, x_3, y_1, y_2, y_3\} \to \text{dom}(J^*)$ such that $P(h(x_1), h(x_2), h(x_3))$, $R(h(x_1), h(y_1))$, $S(h(x_2), h(y_2))$, as well as $R(h(x_3), h(y_3))$ belong to J^*, and $h(y_1) \neq h(y_2)$ and $h(y_2) \neq h(y_3)$. Since $P(h(x_1), h(x_2), h(x_3)$ belongs to J^*, we only have to consider three cases for the value of $h(x_2)$:

1. First, $h(x_2) = 1$. Then it must be the case that $h(x_1) = q$ and $h(x_3) = \hat{q}$, for some propositional variable q mentioned in ϕ. Further, $h(y_1) = f(\bot_q)$, $h(y_2) = 1$, and $h(y_3) = f(\#_q)$. It follows that $f(\bot_q) \neq 1$ and $f(\#_q) \neq 1$, which contradicts the definition of the function f.
2. Second, $h(x_2) = 0$. This case is similar to the previous one.
3. Finally, $h(x_2) = q$, for some propositional variable q mentioned in ϕ. Then there is a clause $(p \vee q \vee r)$ in ϕ such that $h(x_1) = p$ and $h(x_3) = r$. Further, $h(y_1) = f(\bot_p)$, $h(y_2) = f(\#_q)$, and $h(y_3) = f(\bot_r)$, and $f(\bot_p) \neq f(\#_q)$ and $f(\#_q) \neq f(\bot_r)$. It follows from the definition of f that $f(\bot_p) = f(\bot_q) = f(\bot_r)$, and, thus, that $\kappa(p) = \kappa(q) = \kappa(r)$. This is a contradiction because κ is NOT-ALL-EQUAL.

($\Leftarrow$) Assume, on the other hand, that $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$. That is, there exists a solution J' such that $Q(J') = \texttt{false}$. Let $h : J \to J'$ be a homomorphism from J to J'. Let us define a valuation κ for the propositional variables in ϕ as follows: $\kappa(q) = 1$ iff $h(\bot_q) = 1$.

We show next that for each $1 \leq j \leq m$, if $C_j = (p \vee q \vee r)$ then $\kappa(C_j) = 1$, but it is not the case that $\kappa(p) = \kappa(q) = \kappa(r) = 1$. This will show that ϕ is NOT-ALL-EQUAL satisfiable. In order to do so, we first show that $h(\bot_q) = 1$ or $h(\#_q) = 1$, and that $h(\bot_q) = 0$ or $h(\#_q) = 0$, for every propositional variable q mentioned in ϕ.

Assume first, for the sake of contradiction, that $h(\bot_q) = 0$ and $h(\#_q) = 0$, for some propositional variable q mentioned in ϕ. Consider the function $f : \{x_1, y_1, x_2, y_2, x_3, y_3\} \to \text{dom}(J')$, such that $f(x_1) = q$, $f(y_1) = h(\bot_q)$, $f(x_2) = f(y_2) = 1$, $f(x_3) = \hat{q}$, and $f(y_3) = h(\#_q)$. Then $P(f(x_1), f(x_2), f(x_3))$, $R(f(x_1), f(y_1))$, $S(f(x_2), f(y_2))$, as well as $Q(f(x_3), f(y_3))$ belong to J', and $f(y_1) \neq f(y_2)$ and $f(y_2) \neq f(y_3)$. Then $Q(J') = \texttt{true}$, which is a contradiction.

In the same way we can prove that $h(\bot_q) = 0$ or $h(\#_q) = 0$, for every propositional variable q mentioned in ϕ.

Consider now an arbitrary $j \in [1, m]$, and assume that $C_j = (p \vee q \vee r)$. Consider the function $f : \{x_1, y_1, x_2, y_2, x_3, y_3\} \to \text{dom}(J')$, such that $f(x_1) = p$, $f(y_1) = h(\bot_p)$, $f(x_2) = q$, $f(y_2) = h(\#_q)$, $f(x_3) = r$, and $f(y_3) = h(\bot_r)$. Then $P(f(x_1), f(x_2), f(x_3))$, $R(f(x_1), f(y_1))$, $S(f(x_2), f(y_2))$, as well as $R(f(x_3), f(y_3))$ belong to J'. Therefore, since $Q(J') = \texttt{false}$, it must be the case that $h(\bot_p) = h(\#_q)$ or $h(\#_q) = h(\bot_r)$. From the previous remark, either $\kappa(p) = 1 - \kappa(q)$ or $\kappa(q) = 1 - \kappa(r)$. In any case, $\kappa(C_j) = 1$, and it is not the case that $\kappa(p) = \kappa(q) = \kappa(r) = 1$.

This concludes the proof of the second part of the theorem.

We now prove part (3). That is, that there is a LAV data exchange setting $\mathcal{M}$ and a conjunctive query Q with two inequalities, such that Q has almost constant inequalities but does not have constant joins under $\mathcal{M}$, and CERTAIN-ANSWERS($\mathcal{M}$, Q) is coNP-complete.

The LAV setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ is as follows. The source schema $\mathbf{S}$ consists of two binary relations M and N, one ternary relation P, and one 4-ary relation R. The target schema $\mathbf{T}$ consists of two binary relations S and T, one ternary relation U, and one 4-ary relation V. The set Σ_{st} of source-to-target dependencies is:

$$\begin{aligned}
M(x, y) &\rightarrow \exists z(S(x, y) \wedge S(y, x) \wedge V(x, y, z, z) \wedge U(z, z, z) \wedge S(z, z)),\\
R(x, y, v, w) &\rightarrow \exists z(T(x, z) \wedge T(y, z) \wedge V(v, w, x, z) \wedge V(v, w, y, z)),\\
P(x, y, z) &\rightarrow U(x, y, z),\\
N(x, y) &\rightarrow T(x, y).
\end{aligned}$$

The Boolean query Q over $\mathbf{T}$ is as follows:

$$\begin{aligned}
&\exists x \exists x' \exists y \exists y' \exists z \exists z' \exists x_1 \exists y_1 \exists x_2 \exists y_2 (T(x_1, y_1) \wedge T(x_2, y_2)\\
&\quad \wedge U(x, y, z) \wedge S(x, x') \wedge S(y, y') \wedge S(z, z')\\
&\quad \wedge V(x_1, x_2, x', x') \wedge V(x_1, x_2, y', y') \wedge V(x_1, x_2, z', z') \wedge x_1 \neq y_1 \wedge x_2 \neq y_2).
\end{aligned}$$

Clearly, Q has almost constant inequalities in $\mathcal{M}$, but does not have constant joins in $\mathcal{M}$. We prove next that CERTAIN-ANSWERS($\mathcal{M}$, Q) is coNP-complete.

Membership in coNPfollows from [9]. The coNP-hardness is established from a reduction from 3SAT to the complement of the problem studied, namely CERTAIN-ANSWERS($\mathcal{M}$, Q). More precisely, for every 3CNF propositional formula ϕ, we construct in polynomial time an instance I_ϕ of $\mathbf{S}$ such that ϕ is satisfiable iff certain$_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$.

Given a propositional formula $\phi \equiv \bigwedge_{1 \le j \le m} C_j$ in 3CNF, where each C_j is a clause, let I_ϕ be the following source instance:

- The interpretation of the binary relation M in I_ϕ contains the pair $(q, \neg q)$, for each propositional variable q mentioned in ϕ;
- the interpretation of the binary relation N in I_ϕ contains the pairs $(\texttt{a}, \texttt{b})$ and $(\texttt{c}, \texttt{d})$, where a, b, c and d are fresh constants (not mentioned as propositional variables in ϕ);
- the interpretation of the ternary relation P in I_ϕ contains all triples (α, β, γ) such that for some $1 \le j \le m$, $(\alpha \vee \beta \vee \gamma) = C_j$; and
- the interpretation of the 4-ary relation R in I_ϕ contains the tuple $(q, \neg q, \texttt{a}, \texttt{c})$, for each propositional variable q mentioned in ϕ.

Clearly, I_ϕ can be constructed in polynomial time from ϕ.

Let $\#_q$ be the null obtained from the application of the std $M(x, y) \to \exists z(S(x, y) \wedge S(y, x) \wedge V(x, y, z, z) \wedge U(z, z, z) \wedge S(z, z))$ to the tuple $M(q, \neg q)$, and let $\bot_q$ (or $\bot_{\neg q}$) be the null obtained from the application of the std $R(x, y, v, w) \to \exists z(T(x, z) \wedge T(y, z) \wedge R(v, w, x, z) \wedge R(v, w, y, z))$ to the tuple $(q, \neg q, \texttt{a}, \texttt{c})$. The canonical universal solution J for I_ϕ is as follows:

- The interpretation of S in J contains the pairs $(q, \neg q)$, $(\neg q, q)$, and $(\#_q, \#_q)$, for each propositional variable q mentioned in ϕ;
- the interpretation of T in J contains a copy of the interpretation of N in I_ϕ and the pairs $(q, \bot_q)$, $(\neg q, \bot_q)$, for each propositional variable q mentioned in ϕ;
- the interpretation of U in J contains a copy of the interpretation of P in I_ϕ and the tuple $(\#_q, \#_q, \#_q)$, for each propositional variable q mentioned in ϕ; and
- the interpretation of V in J contains the tuples $(q, \neg q, \#_q, \#_q)$, $(a, c, q, \bot_q)$, and $(a, c, \neg q, \bot q)$, for each propositional variable q mentioned in ϕ.

We prove next that certain$_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$ iff ϕ is satisfiable.

($\Leftarrow$) Assume that ϕ is satisfiable, and let κ be a truth assignment for the propositional variables mentioned in ϕ such that $\kappa(\phi) = 1$. Define a function f from dom(J) into dom(J) as follows:

$$f(v) = \begin{cases} q & v = \bot_q \text{ and } \kappa(q) = 1, \\ \neg q & v = \bot_q \text{ and } \kappa(q) = 0, \\ v & \text{otherwise.} \end{cases}$$

Let J^* be the solution for I_ϕ obtained from J by replacing each occurrence of an element v in J by $f(v)$. We show next that $Q(J^*) = \texttt{false}$, and, thus, that certain$_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$.

Assume, for the sake of contradiction, that $Q(J^*) = \texttt{true}$. Then there is a function $h : \{x, x', y, y', z, z', x_1, y_1, x_2, y_2\} \to \text{dom}(J^*)$, such that $T(h(x_1),$

$h(y_1))$, $T(h(x_2), h(y_2))$, $U(h(x), h(y), h(z))$, $S(h(x), h(x'))$, $S(h(y), h(y'))$, $S(h(z), h(z'))$, $V(h(x_1), h(x_2), h(x'), h(x'))$, $V(h(x_1), h(x_2), h(y'), h(y'))$, and $V(h(x_1), h(x_2), h(z'), h(z'))$ belong to J^*, and, furthermore, $h(x_1) \neq h(y_1)$ and $h(x_2) \neq h(y_2)$. Since $V(h(x_1), h(x_2), h(x'), h(x'))$ belongs to J^*, there are only two cases to consider with respect to the values $h(x_1)$ and $h(x_2)$:

1. The first case is that $h(x_1) = q$ and $h(x_2) = \neg q$, for some propositional variable q mentioned in ϕ. But then $h(y_1) = h(y_2) = f(\bot_q)$, because $T(h(x_1), h(y_1)$ and $T(h(x_2), h(y_2))$ belong to J^* . It follows that $f(\bot_q) \neq q$ and $f(\bot_q) \neq \neg q$, which is in contradiction with the definition of the function f.
2. The second case is that $h(x_1) = \mathtt{a}$ and $h(x_2) = \mathtt{c}$. But then $h(x') = q$ or $h(x') = \neg q$, for some propositional variable q mentioned in ϕ. Since $S(h(x), h(x'))$ belongs to J^*, it must be the case that for some clause $(\alpha \vee \beta \vee \gamma)$ in ϕ, $h(x) = \alpha$, $h(y) = \beta$ and $h(z) = \gamma$. Furthermore, $h(x') = \neg\alpha$. Since $\kappa(\phi) = 1$, it must be the case that $\kappa(\alpha) = 1$ or $\kappa(\beta) = 1$ or $\kappa(\gamma) = 1$. Assume that $\kappa(\alpha) = 1$. Since $V(h(x_1), h(x_2), h(x'), h(x')) = V(a, c, \neg\alpha, \neg\alpha)$ belongs to J^*, it follows that $f(\bot_\alpha) = \neg\alpha$. But then $\kappa(\alpha) = 0$, which contradicts our previous assumption. The cases $\kappa(\beta) = 1$ or $\kappa(\gamma) = 1$ are identical.

($\Rightarrow$) Assume, on the other hand, that $\text{certain}_{\mathcal{M}}(Q, I) = \mathtt{false}$. Then there exists a solution J' for I_ϕ such that $Q(J') = \mathtt{false}$. Let h be a homomorphism from J to J'. Let us define a truth assignment κ for the propositional variables mentioned in ϕ as follows: $\kappa(q) = 1$ iff $h(\bot_q) = q$. We prove next that for each $1 \leq j \leq m$, $\kappa(C_j) = 1$, and, therefore, that ϕ is satisfiable.

Let clause C_j be $(\alpha \vee \beta \vee \gamma)$ ($j \in [1, m]$). We prove first that $h(\bot_\alpha) \neq \neg\alpha$ or $h(\bot_\beta) \neq \neg\beta$ or $h(\bot_\gamma) \neq \neg\gamma$. Assume otherwise. Then the function $f : \{x, x', y, y', z, z', x_1, y_1, x_2, y_2\} \to \text{dom}(J')$ defined as $f(x_1) = \mathtt{a}$, $f(y_1) = \mathtt{b}$, $f(x_2) = \mathtt{c}$, $f(y_2) = \mathtt{d}$, $f(x) = \alpha$, $f(x') = \neg\alpha$, $f(y) = \beta$, $f(y') = \neg\beta$, $f(z) = \gamma$, $f(z') = \neg\gamma$ satisfies that $T(f(x_1), f(y_1))$, $T(f(x_2), f(y_2))$, $U(f(x), f(y), f(z))$, $S(f(x), f(x'))$, $S(f(y), f(y'))$, $S(f(z), f(z'))$, $V(f(x_1), f(x_2), f(x'), f(x'))$, $V(f(x_1), f(x_2), f(y'), f(y'))$, $V(f(x_1), f(x_2), f(z'), f(z'))$ belong to J'. Further, $f(x_1) \neq f(y_1)$ and $f(x_2) \neq f(y_2)$. Then $Q(J') = \mathtt{true}$, which is a contradiction.

We prove second that for each propositional variable q mentioned in ϕ, $h(\bot_q) = q$ or $h(\bot_q) = \neg q$. Assume otherwise. Then the function $f : \{x, x', y, y', z, z', x_1, y_1, x_2, y_2\} \to \text{dom}(J')$ defined as $f(x_1) = q$, $f(y_1) = h(\bot_q)$, $f(x_2) = \neg q$, $f(y_2) = h(\bot_q)$, $f(x) = f(x') = f(y) = f(y') = f(z) = f(z') = \#_q$, satisfies that $T(f(x_1), f(y_1))$, $T(f(x_2), f(y_2))$, $U(f(x), f(y), f(z))$, $S(f(x), f(x'))$, $S(f(y), f(y'))$, $S(f(z), f(z'))$, $V(f(x_1), f(x_2), f(x'), f(x'))$, and also satisfies $V(f(x_1), f(x_2), f(y'), f(y'))$ and $V(f(x_1), f(x_2), f(z'), f(z'))$ belong to J'. Further, $f(x_1) \neq f(y_1)$ and $f(x_2) \neq f(y_2)$. Then $Q(J') = \mathtt{true}$, which is a contradiction.

We finally prove that $\kappa(C_j) = 1$. Assume first that $h(\bot_\alpha) \neq \neg\alpha$. Then $h(\bot_\alpha) = \alpha$, and, thus, $\kappa(\alpha) = \kappa(C_j) = 1$. The cases when $h(\bot_\beta) \neq \neg\beta$ and $h(\bot_\gamma) \neq \neg\gamma$ are identical.

This concludes the proof of the theorem.

A.3 Proof of Theorem 6.5

Fix an FO sentence $\phi \equiv \exists x_1 \cdots \exists x_p \forall y_1 \cdots \forall y_m \psi$ in the Bernays-Schönfinkel class. Thus, we have that the vocabulary of ϕ is constant-free, and that ψ mentions neither any function symbol nor the equality symbol. Also, let $\{R_1, \ldots, R_n\}$ be the set of all relation symbols mentioned in ψ. For each relation R_i, $1 \leq i \leq n$, we let r_i denote the arity of R_i. Along the proof we heavily use the following property of ϕ: Either ϕ is unsatisfiable, or it has a model with at most p elements (see, e.g., [6]). Finally, let $S_1, \ldots, S_\ell$ be an enumeration of all the subformulas of ψ, and assume, without loss of generality that $S_1 = \psi$.

To give the intuition behind our reduction, we start by showing a weaker result, namely that CERTAIN-ANSWERS(GLAV, 2-UCQ$^{\neq}$) is coNEXPTIME-hard, where 2-UCQ$^{\neq}$ is the class of unions of conjunctive queries with at most two inequalities per disjunct. This result is proved by a polynomial-time reduction from the satisfiability problem for the Bernays-Schönfinkel class, that is, we start by showing how to construct in polynomial time from ϕ a GLAV data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, a 2-UCQ$^{\neq}$ query Q, and an instance I of $\mathbf{S}$, such that ϕ is satisfiable iff $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$. Although this is still not sufficient to prove the theorem, because Q belongs to 2-UCQ$^{\neq}$, the construction helps obtaining intuition for the second part of the proof, which is technically more involved. Second, using a refinement of the techniques in the first part of the proof, we show how to construct in polynomial time from ϕ another GLAV data exchange setting $\mathcal{M}' = (\mathbf{S}', \mathbf{T}', \Sigma'_{st})$, a conjunctive query Q' with two inequalities, and an instance I' of $\mathbf{S}'$, such that ϕ is satisfiable iff $\text{certain}_{\mathcal{M}'}(Q', I') = \texttt{false}$, and, thus, we conclude that CERTAIN-ANSWERS(GLAV, 2-CQ$^{\neq}$) is coNEXPTIME-hard.

The intuition of the first part of the reduction is the following. We construct a source instance I such that $\text{dom}(I)$ includes p elements $a_1, \ldots, a_p$. This is justified by the fact, mentioned above, that if ϕ is satisfiable then it has a model of size at most p. We then construct a set Σ_{st} of st-tgds such that for each tuple $\bar{a}$ of r_i elements of $\text{dom}(I)$ ($i \in [1, n]$), $R'_i(\bar{a}, \perp)$ belongs to $\text{CAN}(I)$, where $\perp$ is a fresh null value. The target schema $\mathbf{T}$ will also contain one relation F_j of arity $m + 1$ for each subformula S_j of ψ ($j \in [1, \ell]$), such that for each tuple $\bar{b} = (b_1, \ldots, b_m)$ of m elements of $\text{dom}(I)$, $F_j(\bar{b}, \perp)$ belongs to $\text{CAN}(I)$. We are interested in those solutions for I in which each of these null values is replaced by the element either 0 or 1. With each such solution J, we naturally identify a structure $\mathcal{A}_J$ over the vocabulary $\{R_1, \ldots, R_k\}$ as follows: $\bar{a}$ belongs to the interpretation of the symbol R_i in $\mathcal{A}_J$ if and only if $R'_i(\bar{a}, 1) \in J$. Moreover, from those solutions J that define a structure, we are interested in the ones that assign truth values to each subformula of ψ in a consistent way. That is, we are interested in those solutions J such that for every $j \in [1, \ell]$, it holds that $F_j(\bar{b}, 1) \in J$, where $\bar{b} = (b_1, \ldots, b_m)$, if and only if $\mathcal{A}_J$ satisfies subformula S_j with each variable y_i replaced by b_i and each variable x_i replaced by a_i. Then the 2-UCQ$^{\neq}$ query Q constructed in this first reduction is used to verify whether there exists an assignment for the variables $y_1, \ldots, y_m$ such that F_1 does not hold. Thus, given that subformula $S_1 = \psi$, if Q is evaluated over a solution J that represents a structure $\mathcal{A}_J$, then Q holds in J if $\mathcal{A}_J$ does not satisfy the formula $\forall y_1 \cdots \forall y_m \psi$ with each variable x_i replaced by value a_i ($1 \leq i \leq p$). Hence, given

that for every structure $\mathcal{A}$ with at most p elements, there exists a solution J for I under $\mathcal{M}$ such that $\mathcal{A}_J$ is isomorphic to $\mathcal{A}$ (notice that we are no assuming that a_i and a_j represent distinct elements if $i \neq j$, although they are distinct constant symbols in our reduction), we have that ϕ is satisfiable if and only if there exists a solution J for I under $\mathcal{M}$ where Q does not hold, that is, if and only if $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$.

We now present the first reduction.

- The source schema $\mathbf{S}$ consists of three unary relations B, O and U, a set of unary relations $\{V_1, \ldots, V_p\}$ (recall that p is the number of existentially quantified variables in ϕ), two ternary relations C and D, and one binary relation E.
- The target schema $\mathbf{T}$ consists of a relation R'_i of arity $r_i + 1$, for each $i \in [1, n]$, a set $\{V'_1, \ldots, V'_p\}$ of unary relations, two other unary relations O' and U', two ternary relations C' and D', a binary relation E', and an extra set of relations $\{F_1, \ldots, F_\ell\}$ each with arity $m + 1$ (recall that ℓ is the number of subformulas of ψ, and that m is the number of universally quantified variables of ϕ).
- The instance I is as follows. The domain of I contains the elements $a_1, \ldots, a_p$, plus two different constants not used elsewhere in the instance, 1 and 0. The interpretation in I of each symbol of $\mathbf{S}$ is as follows:

$$
\begin{aligned}
B^I &= \{a_1, \ldots, a_p\},\\
O^I &= \{0\},\\
U^I &= \{1\},\\
C^I &= \{(1,1,1), (1,0,0), (0,1,0), (0,0,0)\},\\
D^I &= \{(1,1,1), (1,0,1), (0,1,1), (0,0,0)\},\\
E^I &= \{(0,1), (1,0)\},\\
V_i^I &= \{a_i\}, \quad \text{for each } i \in [1, p].
\end{aligned}
$$

- The set Σ_{st} of source-to-target dependencies is as follows:
  - For each $i \in \{1, \ldots, n\}$ we create a copy of every relation V_i into V'_i:

$$V_i(x) \quad \rightarrow \quad V'_i(x).$$

    We also create a copy of O, U, C, D and E into O', U', C', D' and E', respectively:

$$
\begin{aligned}
O(x) &\rightarrow O'(x),\\
U(x) &\rightarrow U'(x),\\
C(x,y,z) &\rightarrow C'(x,y,z),\\
D(x,y,z) &\rightarrow D'(x,y,z),\\
E(x,y) &\rightarrow E'(x,y).
\end{aligned}
$$

  - For each $i \in \{1, \ldots, n\}$, we populate each R'_i (of arity $r_i + 1$) with every tuple of arity r_i that can be constructed from the constants in B, and create a new null

value associated with each such tuple:

$$B(x_1) \wedge \cdots \wedge B(x_{r_i}) \quad \rightarrow \quad \exists z R'_i(x_1, \ldots, x_{r_i}, z).$$

As we mentioned before, we are interested in those solutions for I that replace each such null value with either 0 or 1, as with each such solution J, we associate a structure $\mathcal{A}_J$ over vocabulary $\{R_1, \ldots, R_n\}$ as follows: $\bar{a}$ belongs to the interpretation of R_i in $\mathcal{A}_J$ iff $R'_i(\bar{a}, 1) \in J$.

– We do the same for each symbol F_j. That is, for every $j \in \{1, \ldots, \ell\}$, we populate each F_j (of arity $m + 1$) with every tuple of arity m that can be constructed from the constants in B, and create a new null value associated with each such tuple:

$$B(x_1) \wedge \cdots \wedge B(x_m) \quad \rightarrow \quad \exists z F_j(x_1, \ldots, x_m, z).$$

We are interested in those solutions that replace each such null value with 0 or 1. Informally, $F_j(a_{i_1}, \ldots, a_{i_m}, 1)$ belongs to one of these solutions J iff the subformula S_j of ψ holds in $\mathcal{A}_J$, whenever we assign to the universally quantified variables $y_1, \ldots, y_m$ the elements $a_{i_1}, \ldots, a_{i_m}$ and to the existentially quantified variables $x_1, \ldots, x_p$ the elements $a_1, \ldots, a_p$.

It is clear at this point what the canonical universal solution $\textsc{Can}(I)$ for I is. Before presenting the query Q, we give an intuition of what Q does: In order to verify that the formula ϕ is satisfiable, one must show a structure $\mathcal{A}$ such that $\mathcal{A} \models \phi$. Intuitively, the query Q will first nondeterministically choose a structure $\mathcal{A}$ from the set of all possible structures that can be built using p elements. Once the structure is chosen, Q will verify that such structure indeed satisfies the formula ϕ. To that extent, first, Q has to nondeterministically guess an interpretation of each relation in $\{R'_1, \ldots, R'_k\}$. It does so by assigning either a value 1 or a value 0 to every null $\perp$ that belongs to a tuple of the form $R'_i(\bar{a}, \perp)$ in $\textsc{Can}(I)$. Intuitively, if the value 1 is assigned to the null $\perp$ in the tuple $R'_i(\bar{a}, \perp)$, then the interpretation of the relation R_i in $\mathcal{A}$ will contain the tuple $\bar{a}$. Second, Q will proceed in the same way for each relation in $\{F_1, \ldots, F_\ell\}$, where the assignment of the value 1 to a null $\perp$ that belongs to the tuple $F_j(\bar{b}, \perp)$ in $\textsc{Can}(I)$, where $\bar{b} = (b_1, \ldots, b_m)$, represents that the j-th subformula of ψ (denoted by S_j) holds in $\mathcal{A}$ when we assign the elements $b_1, \ldots, b_m$ to the variables $y_1, \ldots, y_m$ and the elements $a_1, \ldots, a_p$ to the variables $x_1, \ldots, x_p$, respectively. Afterwards, Q must verify that the assigned null values represent a consistent valuation of the subformulas in $\mathcal{A}$.

Finally, the query will ask if for some $\bar{c}$ there is a tuple $F_1(\bar{c}, \perp)$ in $\textsc{Can}(I)$ such that the null $\perp$ has been assigned the value 0, which intuitively means that $\mathcal{A}$ does not satisfy $\forall y_1 \cdots \forall y_m \psi$ with each variable x_i replaced by value a_i $(1 \leq i \leq p)$.

Formally, the query Q is defined as $Q_\alpha \vee Q_\beta \vee Q_\gamma \vee Q_\delta$, where

– Q_α is $(\bigvee_{i \in [1,n]} Q^i_{\alpha_1}) \vee (\bigvee_{j \in [1,\ell]} Q^j_{\alpha_2})$, where each $Q^i_{\alpha_1}$ is defined as follows:

$$\exists z_1 \cdots \exists z_{r_i} \exists n \exists v \exists w (R'_i(z_1, \ldots, z_{r_i}, n) \wedge O'(v) \wedge U'(w) \wedge n \neq v \wedge n \neq w).$$

and each $Q^j_{\alpha_2}$ is defined as:

$$\exists z_1 \cdots \exists z_m \exists n \exists v \exists w (F_i(z_1, \ldots, z_m, n) \wedge O'(v) \wedge U'(w) \wedge n \neq v \wedge n \neq w).$$

Consider an arbitrary solution J for I. Notice that if for some $1 \leq i \leq n$ the evaluation of $Q^i_{\alpha_1}$ over J is false, then all tuples in the interpretation of the relation R'_i over J must be of form $R'_i(\bar{a}, 0)$ or $R'_i(\bar{a}, 1)$. Likewise, if for some $1 \leq j \leq \ell$ the evaluation of $Q^j_{\alpha_2}$ over J is false, then all tuples in the interpretation of the relation F_j over J must be of form $F_j(\bar{a}, 0)$ or $F_j(\bar{a}, 1)$. Hence, if a solution J is such that $q_\alpha(J) = \texttt{false}$, then all the tuples in the relations $R'_1, \ldots, R'_n$ and $F_1, \ldots, F_\ell$ in J must contain a 0 or a 1 in its last argument.

– Let $\Theta \subseteq \{1, \ldots, \ell\}$ be the set of all indexes j such that S_j is an atomic formula. The query Q_β is defined as $\bigvee_{j \in \Theta} Q^j_\beta$, where for each j such that $S_j = R_i(\bar{x}, \bar{y})$, $\bar{x}$ is a tuple of variables in $\{x_1, \ldots, x_p\}$ and $\bar{y}$ is a tuple of variables in $\{y_1, \ldots, y_m\}$, the query Q^j_β is as follows:

$$\exists y_1 \cdots \exists y_m \exists n \exists v \exists \bar{x} \Bigg(F_j(y_1, \ldots, y_m, n) \wedge R'_i(\bar{x}, \bar{y}, w) \wedge \bigwedge_{x_k \in \bar{x}} V'_k(x_k) \wedge n \neq w \Bigg). \tag{38}$$

Consider now a solution J that does not satisfy Q_α, and consider the associated structure $\mathcal{A}_J$. Assume that S_j holds (resp., does not hold) in $\mathcal{A}_J$ when we assign elements $a_{i_1}, \ldots, a_{i_m}$ to variables $y_1, \ldots, y_m$ and elements $a_1, \ldots, a_p$ to variables $x_1, \ldots, x_p$. Since the tuple $F_j(a_{i_1}, \ldots, a_{i_m}, \bot)$ belongs to $\textsc{Can}(I)$, then J does not satisfy Q^j_β only if J contains the tuple $F_j(a_{i_1}, \ldots, a_{i_m}, 1)$ (resp. $F_j(a_{i_1}, \ldots, a_{i_m}, 0)$), and J contains no other tuple of the form $F_j(a_{i_1}, \ldots, a_{i_m}, v)$, with $v \neq 1$ (resp. $v \neq 0$). Intuitively, every solution J that satisfies neither Q_α nor Q^j_β is such that a tuple $\bar{a}$ is in the interpretation of the relation R_i in $\mathcal{A}_J$ iff the subformula S_j holds under an assignment g of the variables in ϕ, such that g assigns the elements $a_1, \ldots, a_p$ to the existentially quantified variables $x_1, \ldots, x_p$, and $g(\bar{x}, \bar{y}) = \bar{a}$.

It is important to notice that predicate V'_k is included in (38) to ensure that variable x_k in $\bar{x}$ is assigned value a_k, as V'_k in $\textsc{Can}(I)$ is a copy of the interpretation of V_k in I, and $V^I_k = \{a_k\}$.

– Let Θ be as above. Then Q_γ is defined as $\bigvee_{k \in (\{1,\ldots,\ell\} \setminus \Theta)} Q^k_\gamma$, where each query Q^k_γ is defined as follows:
  – If $S_k = (S_g \vee S_h)$, then

$$Q^k_\gamma = \exists y_1 \cdots \exists y_m \exists n \exists v \exists w \exists z (F_k(y_1, \ldots, y_m, n) \wedge F_g(y_1, \ldots, y_m, v) \wedge F_h(y_1, \ldots, y_m, w) \wedge D'(v, w, z) \wedge n \neq z).$$

  – If $S_k = (S_g \wedge S_h)$, then

$$Q^k_\gamma = \exists y_1 \cdots \exists y_m \exists n \exists v \exists w \exists z (F_k(y_1, \ldots, y_m, n) \wedge F_g(y_1, \ldots, y_m, v) \wedge F_h(y_1, \ldots, y_m, w) \wedge C'(v, w, z) \wedge n \neq z).$$

– If $S_k = (\neg S_g)$, then

$$Q_\gamma^k = \exists y_1 \cdots \exists y_m \exists n \exists v \exists z (F_k(y_1, \ldots, y_m, n) \wedge F_g(y_1, \ldots, y_m, v) \wedge E'(v, z) \wedge n \neq z).$$

The purpose of this query is similar to Q_β, but here we ensure the correct interpretation of the subformulas of ψ that are Boolean combinations of other subformulas. For this reason, the tuples in relations C', D' and E' encode the truth tables of $\wedge$, $\vee$ and $\neg$, respectively. For example, if $S_k = (S_g \wedge S_h)$, and a solution J that satisfies neither Q_α nor Q_β is such that it contains tuples $F_g(\bar{a}, 1)$ and $F_h(\bar{a}, 1)$, then J does not satisfy Q_γ^k only if $F_k(\bar{a}, 1)$ is the only tuple of the form $F_k(\bar{a}, v)$ in J.

– Finally, Q_δ is defined to be $\exists y_1 \cdots \exists y_m \exists v (F_1(y_1, \ldots, y_m, v) \wedge O'(v))$. This query asks for a tuple of the form $F_1(\bar{b}, 0)$. That is, this query will not hold in a solution J if and only if none of the tuples in the interpretation of F_1 in J contains a 0 in its last argument.

At this point, it is instructive to show an example of the reduction, to get the idea of the construction.

Example A.3 Let ϕ be the formula $\exists x_1 \exists x_2 \forall y_1 (R_1(x_1, y_1) \vee (\neg R_1(x_2, y_1)))$. Recall that the source schema $\mathbf{S}$ consists of relations B, O, U, C, D, E as described above, plus extra relations V_1 and V_2. The target schema $\mathbf{T}$ consists of relations O', U', C', D', E', R_1', F_1, F_2, F_3 and F_4 (because $(R_1(x_1, y_1) \vee (\neg R_1(x_2, y_1)))$ has 4 subformulas). The enumeration of the subformulas of $(R_1(x_1, y_1) \vee (\neg R_1(x_2, y_1)))$ is as follows: $S_1 = (R_1(x_1, y_1) \vee (\neg R_1(x_2, y_1)))$, $S_2 = (\neg R_1(x_2, y_2))$, $S_3 = R_1(x_1, y_1)$ and $S_4 = R_1(x_2, y_1)$. Then the source-to-target dependencies are:

$$\begin{aligned}
V_1(x) &\rightarrow V_1'(x),\\
V_2(x) &\rightarrow V_2'(x),\\
O(x) &\rightarrow O'(x),\\
U(x) &\rightarrow U'(x),\\
C(x, y, z) &\rightarrow C'(x, y, z),\\
D(x, y, z) &\rightarrow D'(x, y, z),\\
E(x, y) &\rightarrow E'(x, y),\\
B(x_1) \wedge B(x_2) &\rightarrow \exists z R_1'(x_1, x_2, z),\\
B(x_1) &\rightarrow \exists z F_1(x_1, z),\\
B(x_1) &\rightarrow \exists z F_2(x_1, z),\\
B(x_1) &\rightarrow \exists z F_3(x_1, z),\\
B(x_1) &\rightarrow \exists z F_4(x_1, z).
\end{aligned}$$

The instance I of $\mathbf{S}$ is constructed as follows: $B^I = \{a_1, a_2\}$, $0^I = \{0\}$ and $U^I = \{1\}$. Furthermore, $C^I = \{(1,1,1), (1,0,0), (0,1,0), (0,0,0)\}$, $D^I = \{(1,1,1), (1,0,1), (0,1,1), (0,0,0)\}$, and $E^I = \{(0,1), (1,0)\}$. Finally, $V_1^I = \{a_1\}$ and $V_2^I = \{a_2\}$. In this case, $\text{CAN}(I)$ contains the following interpretations of the symbols R_1', F_1, F_2, F_3 and F_4 (all the other relations are simple copies of the respective relations in I). The interpretation of R_1' in $\text{CAN}(I)$ contains the tuples $(a_1, a_1, \perp_1)$, $(a_2, a_2, \perp_2)$, $(a_1, a_2, \perp_3)$, and $(a_2, a_1, \perp_4)$. The interpretation of the relations F_1 in $\text{CAN}(I)$ contains the tuples $(a_1, \perp_5)$ and $(a_2, \perp_6)$; the interpretation of the relations F_2 in $\text{CAN}(I)$ contains the tuples $(a_1, \perp_7)$ and $(a_2, \perp_8)$; the interpretation of the relations F_3 in $\text{CAN}(I)$ contains the tuples $(a_1, \perp_9)$ and $(a_2, \perp_{10})$; and interpretation of the relations F_4 in $\text{CAN}(I)$ contains the tuples $(a_1, \perp_{11})$ and $(a_2, \perp_{12})$. Finally, the queries Q_α, Q_β, Q_γ and Q_δ are as follows in this case:

- Q_α is the union of the following queries:

$$Q_{\alpha_1}^1 = \exists x_1 \exists x_2 \exists n \exists v \exists w (R_1'(x_1, x_2, n) \wedge O'(v) \wedge U'(w) \wedge n \neq v \wedge n \neq w),$$
$$Q_{\alpha_2}^1 = \exists x_1 \exists n \exists v \exists w (F_1(x_1, n) \wedge O'(v) \wedge U'(w) \wedge n \neq v \wedge n \neq w),$$
$$Q_{\alpha_2}^2 = \exists x_1 \exists n \exists v \exists w (F_2(x_1, n) \wedge O'(v) \wedge U'(w) \wedge n \neq v \wedge n \neq w),$$
$$Q_{\alpha_2}^3 = \exists x_1 \exists n \exists v \exists w (F_3(x_1, n) \wedge O'(v) \wedge U'(w) \wedge n \neq v \wedge n \neq w),$$
$$Q_{\alpha_2}^4 = \exists x_1 \exists n \exists v \exists w (F_4(x_1, n) \wedge O'(v) \wedge U'(w) \wedge n \neq v \wedge n \neq w).$$

- Q_β is the union of Q_β^3 and Q_β^4, where:

$$Q_\beta^3 = \exists y_1 \exists n \exists v \exists x_1 (F_3(y_1, n) \wedge R_1'(x_1, y_1, w) \wedge V_1'(x_1) \wedge n \neq w),$$
$$Q_\beta^4 = \exists y_1 \exists n \exists v \exists x_2 (F_4(y_1, n) \wedge R_1'(x_2, y_1, w) \wedge V_2'(x_2) \wedge n \neq w).$$

- Q_γ is the union of Q_γ^1 and Q_γ^2, where:

$$Q_\gamma^1 = \exists y_1 \exists n \exists v \exists w \exists z (F_1(y_1, n) \wedge F_3(y_1, v) \wedge F_2(y_1, w) \wedge D'(v, w, z) \wedge n \neq z),$$
$$Q_\gamma^2 = \exists y_1 \exists n \exists v \exists z (F_2(y_1, n) \wedge F_4(y_1, v) \wedge E'(v, z) \wedge n \neq z).$$

- $Q_\delta = \exists y_1 \exists v (F_1(y_1, v) \wedge O'(v))$.

This concludes the example.

From the definitions of data exchange setting $\mathcal{M}$, source instance I and query Q, it is straightforward but lengthy to prove that ϕ is satisfiable if and only if $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$, which concludes the proof of the fact that CERTAIN-ANSWERS(GLAV, 2-UCQ$^{\neq}$) is coNEXPTIME-hard.

We now continue with the second part of the reduction. As we mentioned before, the problem with the previous query Q is that it is a *union* of conjunctive queries with at most two inequalities per disjunct. Fix a FO formula ϕ that belongs to the Bernays-Schönfinkel class. Next, based on the previous reduction, we construct for ϕ a second

data exchange setting $\mathcal{M}' = (\mathbf{S}', \mathbf{T}', \Sigma'_{st})$, an instance I' of $\mathbf{S}'$ and a conjunctive query Q' with two inequalities, and then provide a complete proof that ϕ is satisfiable iff certain$_{\mathcal{M}'}(Q', I') = \texttt{false}$.

First, let us explain some of the techniques used in the second reduction. Recall that in the previous reduction the target schema contained the relation symbols $R_1, \ldots, R_n$ and $F_1, \ldots, F_\ell$. The idea of the second reduction is to use a single relation symbol R' to code the same information stored in the relation symbols $R_1, \ldots, R_n$ of the first reduction. In order to do this, we use the first position in R' to store the particular relation R_i, $1 \leq i \leq n$ of ϕ that is being represented. Notice that we do not assume that the relations are of the same arity; instead we choose the arity of R' depending on the maximum arity of all relations in the vocabulary of ϕ. We will also code the information that was previously stored in the relation symbols $F_1, \ldots, F_\ell$ by using again a single relation symbol F' and an extra element to store which subformula S_j, $1 \leq j \leq \ell$ is being represented.

We need some additional notation. Let again ℓ be the number of subformulas of ϕ, and $\Theta \subseteq \{1, \ldots, \ell\}$ be the set of all indexes j such that S_j is an atomic formula. Let $|\Theta|$ be the size of Θ, that is, the number of atomic subformulas of ϕ. We assume that Θ is ordered, and we use a function $\tau : \Theta \to \{1, \ldots, |\Theta|\}$ such that $\tau(j) = m$ if j is the m-th element of Θ.

Now we show the data exchange setting $\mathcal{M}'$:

- The source schema $\mathbf{S}'$ consists of eight unary relations E_a, E_b, E_f, D, B, C, O and U, a set $\{Q_1, \ldots, Q_n\}$ of unary relations (one for each relation R_i), another set of unary relations $\{V_1, \ldots, V_p\}$ (recall that p is the number of existentially quantified variables in ϕ), a relation Z of arity 4, and a relation A of arity equal to $|\Theta| + 5$.
- The target schema $\mathbf{T}'$ consists of a relation R' with arity $\max_{i \in [1,n]} r_i + 2$, a relation Z' of arity 4, a relation A' with the same arity than A, a relation F' of arity $m + 2$ (recall m is the number of universally quantified variables in ϕ), and a set of binary relations $\{V'_1, \ldots, V'_p\}$.
- The instance I' is as follows. The domain of I' contains the elements $a_1, \ldots, a_p$, $s_1, \ldots, s_\ell$, $c_1, \ldots, c_n$, plus the elements s_f, s_a, s_b, 1, 0, and d. The interpretation of each symbol in $\mathbf{S}$ in I' is as follows:
  - $B^{I'} = \{a_1, \ldots, a_p\}$.
  - $O^{I'} = \{0\}$, $U^{I'} = \{1\}$ and $D^{I'} = \{d\}$.
  - $E_a^{I'} = \{s_a\}$, $E_b^{I'} = \{s_b\}$ and $E_f^{I'} = \{s_f\}$.
  - $C^{I'} = \{s_1, \ldots, s_\ell\}$.
  - $Q_k^{I'} = \{c_k\}$ for every $k \in [1, n]$.
  - $V_i^{I'} = \{a_i\}$ for every $i \in [1, p]$.
  - $A^{I'}$ is as follows:
    - It contains a tuple with only ds, except for a s_f in its first position and elements s_1, 0 in the last two positions. For example, if the arity of A is 8, we would create the following tuple: $A(s_f, d, d, d, d, d, s_1, 0)$.
    - For each $i \in [1, n]$, $A^{I'}$ will contain a tuple in which every position contains the element d, except for the first position that contains the element s_0, the second position that contains the element c_i, and the last position that contains the element 1. For example, if the arity of A is 8, we would create a tuple $A(s_a, c_i, d, d, d, d, d, 1)$ for each $1 \leq i \leq n$.

- For each $j \in [1, \ell]$, $A^{I'}$ will contain a tuple in which every position contains the element d, except for the first position that contains the element s_b, the second position that contains the element s_j, and the last position that contains the element 1. For example, if the arity of A is 8, we would create a tuple $A(s_b, s_j, d, d, d, d, d, 1)$ for each $1 \leq j \leq \ell$.
- For each subformula S_j of ψ such that $j \in \Theta$, assume that $S_j = R_i(\bar{z})$, $A^{I'}$ contains two tuples with only d's, except for an element s_j in the first position, a c_i in the $(\tau(j) + 2)th$ position and either the element 0 or the element 1 in the last position. For example, if the arity of A is 8, for $S_2 \equiv R_2$ and such that $\tau(2) = 1$, we would create the tuples: $A(s_2, d, c_2, d, d, d, d, 0)$ and $A(s_2, d, c_2, d, d, d, d, 1)$.
- Then for each subformula S_j, $j \notin \Theta$, such that $S_j \equiv (S_g \wedge S_h)$ or $S_j \equiv (S_g \vee S_h)$, $A^{I'}$ contains two tuples, both with only d's except for an element s_j in the first position, and elements s_g, s_h, and either the element 0 or the element 1 in the last three positions. Continuing with the example, if the arity of A is 8 and if $S_1 \equiv (S_2 \wedge S_3)$ then we create tuples $A(s_1, d, d, d, d, s_2, s_3, 0)$ and $A(s_1, d, d, d, d, s_2, s_3, 1)$.
- For each subformula S_j, $j \notin \Theta$, such that $S_j \equiv (\neg S_g)$, $A^{I'}$ contains two tuples, both with only d's except for the element s_j in the first position, and elements s_g and either the element 0 or the element 1 in the last two positions. Continuing with the example, if the arity of A is 8 and if $S_1 \equiv (\neg S_2)$ then we create tuples $A(s_1, d, d, d, d, d, s_2, 0)$ and $A(s_1, d, d, d, d, d, s_2, 1)$.

– Finally, we construct $Z^{I'}$ as follows:
- The tuple $(s_f, 0, 0, 1)$ belongs to $Z^{I'}$.
- For each subformula S_j, $j \in \Theta$, the following tuples belong to $Z^{I'}$: $(s_j, 0, 0, 1)$ and $(s_j, 0, 0, 0)$.
- For each subformula S_j, $j \notin \Theta$, such that $S_j \equiv S_g \vee S_h$, the following tuples belong to $Z^{I'}$: $(s_j, 1, 1, 1)$, $(s_j, 0, 1, 1)$,$(s_j, 1, 0, 1)$ and $(s_j, 0, 0, 0)$.
- For each subformula S_j, $j \notin \Theta$, such that $S_j \equiv S_g \wedge S_h$, the following tuples belong to $Z^{I'}$: $(s_j, 1, 1, 1)$, $(s_j, 0, 1, 0)$, $(s_j, 1, 0, 0)$ and $(s_j, 0, 0, 0)$.
- For each subformula S_j, $j \notin \Theta$, such that $S_j \equiv \neg S_g$, the following tuples belong to $Z^{I'}$: $(s_j, 0, 1, 0)$ and $(s_j, 0, 0, 1)$.

This finishes the definition of I'.

- The set Σ'_{st} of source-to-target dependencies is as follows:
  - We create a copy of A and Z into A' and Z', respectively:

$$A(\bar{x}) \quad \rightarrow \quad A'(\bar{x}), \tag{39}$$

$$Z(x, y, z, w) \quad \rightarrow \quad Z'(x, y, z, w). \tag{40}$$

  - For each $i \in [1, p]$, we copy every pair of the form (s_k, a_i), $k \in [1, \ell]$, into V'_i:

$$V_i(x) \wedge C(y) \quad \rightarrow \quad V'_i(y, x). \tag{41}$$

  - For each $i \in [1, p]$, we copy every pair of the form (a_j, s_a), (a_j, s_b) and (a_j, s_f), $j \in [1, p]$, into V'_i:

$$B(x) \wedge E_a(z) \quad \rightarrow \quad V'_i(z, x), \tag{42}$$

$$B(x) \wedge E_b(z) \quad \rightarrow \quad V_i'(z, x), \tag{43}$$

$$B(x) \wedge E_f(z) \quad \rightarrow \quad V_i'(z, x). \tag{44}$$

– Let $r_{\max}$ be $\max_{i \in [1,n]} r_i$. For each $i \in [1, n]$ we add the following st-tgds to Σ'_{st}:

$$\begin{aligned} & Q_i(y) \wedge E_a(z) \wedge D(w) \wedge O(v) \wedge B(x_i) \wedge \cdots \wedge B(x_{r_i}) \\ & \wedge D(x_{r_i+1}) \wedge \cdots \wedge D(x_{r_{\max}}) \\ & \rightarrow \quad \exists n \big(R'(y, x_1, \ldots, x_{r_{\max}}, n) \wedge R'(w, x_1, \ldots, x_1, n) \\ & \qquad \wedge F'(y, x_1, \ldots, x_1, n) \wedge Z'(z, v, v, n) \big). \end{aligned} \tag{45}$$

The main idea of this dependency is to populate the relation R with each possible tuple that can be constructed using an element from $c_1, \ldots, c_n$ in the first position and elements from $a_1, \ldots, a_p$ in the next $r_{\max}$ positions. As mentioned before, this tuple will encode all of the tuples in the relations $R_1, \ldots, R_n$ of the previous reduction. The problem is that the arity of these relations may not be the same. For that reason, for each $i \in [1, n]$, the tuples starting with c_i are only populated with combinations of length r_i. The remaining positions of the tuples are filled with the element d. More precisely, for each $i \in [1, n]$ and tuple $a_{j_1}, \ldots, a_{j_{r_i}}$ of elements in $\{a_1, \ldots, a_p\}$, we add the following tuples to the interpretation of R' in $\text{CAN}(I')$: $(c_i, a_{j_1}, \ldots, a_{j_{r_i}}, d, \ldots, d, \perp)$ and $(d, a_{j_1}, \ldots, a_{j_1}, a_{j_1}, \ldots, a_{j_1}, \perp)$, where $\perp$ is a fresh null value. In such case, we also add the tuple $(s_a, 0, 0, \perp)$ to the interpretation of Z' in $\text{CAN}(I')$, and the tuple $(c_i, a_{j_1}, \ldots, a_{j_1}, \perp)$ to the interpretation of F' in $\text{CAN}(I')$.

– We also add the following st-tgd to Σ'_{st}:

$$\begin{aligned} & C(y) \wedge E_b(z) \wedge D(w) \wedge O(v) \wedge B(x_1) \wedge \cdots \wedge B(x_m) \\ & \rightarrow \quad \exists n \big(F'(y, x_1, \ldots, x_m, n) \wedge Z'(z, v, v, n) \wedge R'(y, x_1, \ldots, x_1, n) \\ & \qquad \wedge R'(w, x_1, \ldots, x_1, n) \big). \end{aligned} \tag{46}$$

The idea is that the interpretation of F' in $\text{CAN}(I')$ contains for every $j \in [1, \ell]$ and every tuple $a_{i_1}, \ldots, a_{i_m}$ of elements in $\{a_1, \ldots, a_p\}$, the tuple $(s_j, a_{i_1}, \ldots, a_{i_m}, \perp)$, where $\perp$ is a fresh null value. In such case, we also add the tuples $(s_j, a_{i_1}, \ldots, a_{i_1}, \perp)$ and $(d, a_{i_1}, \ldots, a_{i_1}, \perp)$ to the interpretation of R', and the tuple $(s_b, 0, 0, \perp)$ to the interpretation of Z' in $\text{CAN}(I')$.

– The following are also in Σ'_{st}:

$$D(y) \wedge O(z) \wedge B(x_1) \wedge \cdots \wedge B(x_{r_{\max}}) \quad \rightarrow \quad R'(y, x_1, \ldots, x_{r_{\max}}, z), \tag{47}$$

$$D(y) \wedge U(z) \wedge B(x_1) \wedge \cdots \wedge B(x_{r_{\max}}) \quad \rightarrow \quad R'(y, x_1, \ldots, x_{r_{\max}}, z). \tag{48}$$

That is, every tuple of the form $(d, a_{i_1}, \ldots, a_{i_{r_{\max}}}, 0)$ and $(d, a_{i_1}, \ldots, a_{i_{r_{\max}}}, 1)$, where $a_{i_1}, \ldots, a_{i_{r_{\max}}}$ is a tuple of elements in $\{a_1, \ldots, a_p\}$, belongs to the interpretation of R' in $\text{CAN}(I')$.

– Finally, we also add the following st-tgds to Σ'_{st}:

$$D(y) \wedge O(z) \wedge B(x_1) \wedge \cdots \wedge B(x_m) \quad \rightarrow \quad F'(y, x_1, \ldots, x_m, z), \tag{49}$$

$$D(y) \wedge U(z) \wedge B(x_1) \wedge \cdots \wedge B(x_m) \rightarrow F'(y, x_1, \ldots, x_m, z), \quad (50)$$

$$E_a(y) \wedge O(z) \wedge B(x_1) \wedge \cdots \wedge B(x_m) \rightarrow F'(y, x_1, \ldots, x_m, z), \quad (51)$$

$$E_b(y) \wedge O(z) \wedge B(x_1) \wedge \cdots \wedge B(x_m) \rightarrow F'(y, x_1, \ldots, x_m, z), \quad (52)$$

$$E_f(y) \wedge O(z) \wedge B(x_1) \wedge \cdots \wedge B(x_m) \rightarrow F'(y, x_1, \ldots, x_m, z). \quad (53)$$

That is, every tuple of the form $(d, a_{i_1}, \ldots, a_{i_m}, 0)$ and $(d, a_{i_1}, \ldots, a_{i_m}, 1)$, where $a_{i_1}, \ldots, a_{i_m}$ is a tuple of elements in $\{a_1, \ldots, a_p\}$, belongs to the interpretation of F' in $\text{CAN}(I')$. Also, every tuple of form $(s_a, a_{i_1}, \ldots, a_{i_m}, 0)$, $(s_b, a_{i_1}, \ldots, a_{i_m}, 0)$ or $(s_f, a_{i_1}, \ldots, a_{i_m}, 0)$, where $a_{i_1}, \ldots, a_{i_m}$ are elements in $\{a_1, \ldots, a_p\}$, belongs to the interpretation of F' in $\text{CAN}(I')$.

This finishes the definition of Σ'_{st}.

We now show the Boolean CQ query Q' with two inequalities. We first define a function $\kappa : \Theta \rightarrow \{1, \ldots, n\}$ such that $\kappa(j) = i$ iff the atomic formula S_j mentions the relation R_i. Moreover, for every $j \in \Theta$, we assume that every S_j is of the form $S_j = R_{\tau(j)}(\bar{x}_j, \bar{y}_j)$. The query Q' is as follows:

$$\begin{aligned}
Q' \equiv\ & \exists x_1 \cdots \exists x_p \exists y_1 \cdots \exists y_m \exists t_0 \exists t_1 \cdots \exists t_{|\Theta|} \exists z_1^a \cdots \exists z_{r_{\max}}^a \exists z_1^b \cdots \exists z_m^b \\
& \exists q \exists r \exists k \exists k' \exists u \exists v \exists w \exists w' \exists h_1 \cdots \exists h_{|\Theta|} \\
& \Big[A'(q, t_0, t_1, \ldots, t_{|\Theta|}, k, k', u) \wedge R'(t_0, z_1^a, \ldots, z_{r_{\max}}^a, v) \\
& \wedge F(t_0, z_1^b, \ldots, z_m^b, v) \wedge \bigwedge_{j \in \Theta} \big(R'(t_{\tau(j)}, \bar{x}_j, \bar{y}_j, \bar{h}_{\tau(j)}, v) \big) \wedge \bigwedge_{i \in [1,p]} V_i'(q, x_i) \\
& \wedge F'(q, y_1, \ldots, y_m, n) \wedge F'(k, y_1, \ldots, y_m, w) \wedge F'(k', y_1, \ldots, y_m, w') \\
& \wedge Z'(q, w, w', v) \wedge n \neq v \wedge u \neq v \Big]
\end{aligned}$$

where each tuple $\bar{h}_{\tau(j)}$, for $j \in \Theta$, is a tuple of variables $h_{\tau(j)}$ such that, if $S_j = R_i$ then $r_{\max} = r_i + |\bar{h}_{\tau(j)}|$.

Before we continue with the proof, we explain the intuition behind the query Q'. As opposed to the query of the first part of this proof, the query Q' is a single conjunctive query with two inequalities. Thus, the second reduction must correctly simulate the queries Q_α, Q_β, Q_γ and Q_δ that where used in the first part of the reduction using a single query. To this extent, we use the relation Z' to code the values previously stored in the relations C', D' and E'. We also use the relation A' as a controller for the query. The intuition behind the relation A can be explained as follows: as for the first part of the reduction, we are interested in those solutions for I in which each of the null values in the relations R' and F' in $\text{CAN}(I)$ are replaced by the element 0 or 1. Assume that Q' holds in one of these solutions J, and let ρ be an assignment for the variables of Q' that satisfy the body of the query. By taking a closer look at the possible tuples of A' in $\text{CAN}(I)$ we find several possible assignments for $\rho(q)$. Each of these possible assignments represent which part of Q_α, Q_β, Q_γ or Q_δ is Q' simulating. More precisely, when $\rho(q) = s_a$ or $\rho(q) = s_b$, the query Q' will represent

Q_{α_1} and Q_{α_2}, respectively. Further, if $\rho(q) = s_j$ for some $j \in \Theta$, then Q' will work as Q_β. On the other hand, if $\rho(q) = s_j$ for some $j \notin \Theta$, the query Q' will simulate the query Q_γ. Finally, the query Q_δ is simulated by Q' when $\rho(q) = s_f$.

We also make considerable use of tuples that contain the element d. Intuitively, this element is used as a special wildcard element by Q'. For example, let again J be a solution build by replacing the nulls in $\text{CAN}(I)$ by the element 0 or 1, assume that Q' holds in J, and let ρ again be a satisfying assignment for the variables of Q'. Assume also that $\rho(q) = s_a$. In this case, since the query Q' is intuitively simulating the query Q_{α_1}, one would expect no use for any predicate in Q' using the relation F. Nevertheless, query Q' contains, for example, the predicate $F'(k, y_1, \ldots, y_m, w)$. By looking at the relation A' in $\text{CAN}(I)$, one obtains that if $\rho(q) = s_a$, then ρ must assign the element d to k. Further, we know from the applications of st-tgds (49) and (50) that the solution J must contain tuples of the form $F'(d, \bar{c}, 0)$ and $F'(d, \bar{c}, 1)$ for every combination $\bar{c}$ of elements in $\{a_1, \ldots, a_p\}$. This ensures in particular that there will always be a witness for the predicate $F'(k, y_1, \ldots, y_m, w)$ of Q' in J when the assignment ρ assigns the element s_a to the variable q in Q'.

We now show that ϕ is satisfiable is and only if $\text{certain}_{\mathcal{M}'}(Q', I') = \texttt{false}$.

($\Leftarrow$) Assume first that ϕ is satisfiable. Then, it is satisfiable by a structure of cardinality at most p. Let $\mathcal{A}$ be such structure, and assume without loss of generality that the elements of $\mathcal{A}$ are $\{a_1, \ldots, a_p\}$ and that $\mathcal{A}$ satisfies $\forall y_1 \cdots \forall y_m\ \psi$ when we assign to each free variable x_i in ψ the corresponding element a_i in $\mathcal{A}$, $i \in [1, p]$. Define a function h from $\text{CAN}(I')$ to $\text{CAN}(I')$ as follows:

- If v is a constant, then $h(v) = v$;
- $h(v) = 1$, if v is the null value $\perp$ such that the tuple $R'(c_i, a_{i_1}, \ldots, a_{i_{r_i}}, d, \ldots, d, \perp)$ belongs to $\text{CAN}(I')$ and the interpretation of the relation R_i in $\mathcal{A}$ contains the tuple $(a_{i_1}, \ldots, a_{i_{r_i}})$, $a_{i_l} \in \{a_1, \ldots, a_p\}$, $l \in [1, r_i]$;
- $h(v) = 1$, if v is a null value $\perp$ such that the tuple $F(s_j, a_{j_1}, \ldots, a_{j_m}, \perp)$ belongs to $\text{CAN}(I')$ and the subformula S_j holds in $\mathcal{A}$ when we assign to the universally quantifies variables $y_1, \ldots, y_m$ the elements $a_{j_1}, \ldots, a_{j_m}$, $b_{j_l} \in \{a_1, \ldots, a_p\}$, $l \in [1, m]$, and to the existentially quantifies variables $x_1, \ldots, x_b$ the elements $a_1, \ldots, a_p$; and
- otherwise, $h(v) = 0$.

Let J^* be the solution obtained by replacing each element v in $\text{CAN}(I')$ for $h(v)$. Notice also that the function h assigns to each null in $\text{CAN}(I')$ an element in $\{0, 1\}$. We now show that the evaluation of Q' over J^* is false, and thus $\text{certain}_{\mathcal{M}'}(Q', I') = \texttt{false}$.

Assume for the sake of contradiction that $Q'(J^*) = \texttt{true}$. Then, there is a function $f : \{x_1, \ldots, x_p, y_1, \ldots, y_m, t_0, t_1, \ldots, t_{|\Theta|}, z_1^a, \ldots z_{r_{\max}}^a, z_1^b, \ldots, z_m^b, q, r, k, k', u, v, w, w', h_1, \ldots, h_\Theta\} \to \text{dom}(J^*)$, such that for every conjunct $P(\bar{x})$ of Q' it is the case that $f(P(\bar{x}))$ belongs to J^*, and that $f(v) \neq f(u)$ and $f(v) \neq f(n)$.

From the construction of J^*, it is easy to see that f must map the variable q in the query Q' to an element in $\{s_a, s_b, s_f, s_1, \ldots, s_\ell\}$. Thus, depending of the value of $f(q)$, we have several cases:

- Assume first that $f(q) = s_a$. Notice that the only tuples in the interpretation of A' in J^* that contain the element s_a in their first position are of the form

$A'(s_a, c_i, d, \ldots, d, 1)$, for some $1 \leq i \leq n$. Further, the only tuple in the interpretation of the relation F in J^* with the element s_a in it's first position is $F(s_a, d, \ldots, d, 0)$. Thus, we obtain that $f(u) = 1$ and $f(n) = 0$. However, we know that f is a function such that $f(v) \neq f(u)$ and $f(v) \neq f(n)$. It then must be the case that $f(v) \neq 1$ and $f(v) \neq 0$. This is a contradiction: we know that $f(t_0) = c_i$ for some $1 \leq i \leq n$. Further, every tuple in the interpretation of R' in $\textsc{Can}(I')$ that contains an element c_i, $1 \leq i \leq n$, in it's first position has a null value in it's last position. Thus, from the construction of h, it must be the case that $f(v) = 1$ or $f(v) = 0$.

- Assume that $f(q) = s_b$. Then, using the same argument that in the previous paragraph we obtain that $f(u) = 1$. Further, since the only tuple in the interpretation of the relation F' in J^* with the element s_b in it's first position is $F'(s_b, d, \ldots, d, 0)$, it must be the case that $f(n) = 0$. Again, f is such that $f(v) \neq f(u)$ and $f(v) \neq f(n)$, and thus we obtain that $f(v) \neq 1$ and $f(v) \neq 0$. Using the arguments shown in the previous paragraph it can be shown that this is a contradiction: all tuples in the interpretation of A' over J^* that start with the element s_b contain an element in $s_1, \ldots, s_\ell$ in their second position. Thus, $f(t_0) \in \{s_1, \ldots, s_j\}$, and thus, since the only tuples in the interpretation of F' in J^* that start with an element s_j contain a null in their last position, we conclude that $f(n) = 1$ or $f(n) = 1$.
- Assume now that $f(q) = s_j$ for some $j \in \Theta$ such that $S_j = R_i(\bar{x}_j, \bar{y}_j)$ for some $1 \leq i \leq n$, and where $\bar{x}_j$ is a tuple of variables in $\{x_1, \ldots, x_p\}$ and $\bar{y}_j$ is a tuple of variables in $\{y_1, \ldots, y_m\}$. Notice that, since $f(q) = s_j$, from the construction of the interpretation of the relation A in I' it must be the case that $f(t_{\tau(j)}) = c_i$, and that $f(t_{\tau(k)}) = d$ for every other $k \in [1, |\Theta|]$, $k \neq j$. Assume now that $f(u) = 1$ (the case when $f(u) = 0$ is completely symmetrical). Then, since we know that $f(v) \neq f(u)$ and $f(v) \neq f(n)$, it must be that $f(v) = 0$, and thus $f(n) \neq 0$.

  Let $\perp$ be the null value such that the tuple $F'(s_j, f(y_1), \ldots, f(y_m), \perp)$ belongs to $\textsc{Can}(I')$. Then, $F'(s_j, f(y_1), \ldots, f(y_m), h(\perp))$ belongs to J^*, and so it must be the case that $f(n) = h(\perp)$. Since h assigns the value 0 or 1 to every null in $\textsc{Can}(I')$, and since $f(n) \neq 0$. It must be the case that $f(n) = 1$, and thus $h(\perp) = 1$. Then, from the construction of h, the structure $\mathcal{A}$ satisfies S_j when we assign the elements $a_1, \ldots, a_p$ to the variables $x_1, \ldots, x_p$ and the elements $f(y_1), \ldots, f(y_m)$ to the universally quantified variables $y_1, \ldots, y_m$ in ϕ. However, since $f(v) = 0$, J^* must contain the tuple $R'(c_i, f(\bar{x}_j), f(\bar{y}_j), f(\bar{h}_j), 0)$. Let now $\perp$ be the null value such that the tuple $R'(c_i, f(\bar{x}_j), f(\bar{y}_j), f(\bar{h}_j), \perp)$ belongs to $\textsc{Can}(I')$. It then must be the case that $h(\perp) = 0$. Further, from the construction of the relation V' we obtain that f assigns the element a_k to each variable x_k in Q', that is, $f(x_k) = a_k$, for every $1 \leq k \leq p$. We then conclude from the construction of h that $R_i(\bar{x}_j, \bar{y}_j)$ does not hold in $\mathcal{A}$ when we assign the elements $a_1, \ldots, a_p$ to the existentially quantifies variables $x_1, \ldots, x_p$ and the elements $f(y_1), \ldots, f(y_m)$ to the universally quantified variables $y_1, \ldots, y_m$ in ϕ. This is a contradiction.
- Next, assume that $f(q) = s_j$ for some $J \notin \Theta$, such that $S_j = S_{j_1} \vee S_{j_2}$ (the other two cases are completely symmetrical). Further, assume that $f(u) = 1$ (the case when $f(u) = 0$ is also symmetrical). Then, since f is such that $f(v) \neq f(u)$ and $f(v) \neq f(n)$, we obtain that $f(v)$ must be different from 1. A close inspection to the interpretation Z' in J^* reveals that $f(v)$ must be the element 0,

and then, correspondingly, $f(n) \neq 0$. Let $\perp$ be the null value such that the tuple $F'(s_j, f(y_1), \ldots, f(y_m), \perp)$ belongs to $\text{CAN}(I')$. Notice that $h(\perp) = 1$ (otherwise $f(n) = 0$), and then from the construction of h, we obtain that $\mathcal{A}$ satisfies S_j when we assign the elements $a_1, \ldots, a_p$ to the existentially quantifies variables $x_1, \ldots, x_p$ and the elements $f(y_1), \ldots, f(y_m)$ to the universally quantified variables $y_1, \ldots, y_m$ in ϕ. Let also $\perp_{j_1}$ and $\perp_{j_2}$ be null values such that the tuples $F(s_{j_1}, f(y_1), \ldots, f(y_m), \perp_{j_1})$ and $F(s_{j_2}, f(y_1), \ldots, f(y_m), \perp_{j_2})$ belong to $\text{CAN}(I')$. From the construction of the relation Z in I, the only tuple in the interpretation of the relation Z' in $\text{CAN}(I')$ (and thus in J^*) with the element s_j in it's first position and the element 0 in it's last position is the tuple $(s_j, 0, 0, 0)$. Then, it must be the case that $f(w) = f(w') = 0$. We conclude then that $h(\perp_{j_1}) = h(\perp_{j_2}) = 0$, which means that $\mathcal{A}$ does not satisfy neither S_{j_1} nor S_{j_2} when we assign the elements $a_1, \ldots, a_p$ to the existentially quantifies variables $x_1, \ldots, x_p$ and the elements $f(y_1), \ldots, f(y_m)$ to the universally quantified variables $y_1, \ldots, y_m$. This is a contradiction.

- Finally, assume that $f(q) = s_f$. Since the only tuple in the interpretation of A' in $\text{CAN}(I')$ with the element s_f in it's first position is $(s_f, d, \ldots, d, s_1, 0)$, it must be that $f(k') = s_1$ and $f(u) = 0$. Let $\perp$ be the null value such that the tuple $F'(s_1, f(y_1), \ldots, f(y_m), \perp)$ belongs to $\text{CAN}(I')$. From the construction of Z in I, it must also be the case that $f(v) = 1$ and $f(w') = 0$, in other words, it must also be that $h(\perp) = 0$, and then $\mathcal{A}$ does not satisfy S_1 when we assign the elements $a_1, \ldots, a_p$ to the existentially quantifies variables $x_1, \ldots, x_p$ and the elements $f(y_1), \ldots, f(y_m)$ to the universally quantified variables $y_1, \ldots, y_m$. This is a contradiction, because we assumed that ϕ is satisfiable under this valuation.

($\Rightarrow$) Assume that $\text{certain}_{\mathcal{M}'}(Q', I') = \texttt{false}$. Then there exists a solution J^* such that $Q'(J^*) = \texttt{false}$. Construct from J^* a structure $\mathcal{A}$ as follows: The domain of $\mathcal{A}$ is $\{a_1, \ldots, a_p\}$. The interpretation of the relation R_i, $i \in [1, n]$ is the following: The tuple $a_{i_1}, \ldots, a_{i_{r_i}}$ belongs to the interpretation of R_i in $\mathcal{A}$ iff the tuple $R'(c_i, a_{i_1}, \ldots, a_{i_{r_i}}, d, \ldots, d, 1)$ belongs to J. To prove that $\mathcal{A}$ satisfies ϕ, we will prove the following: for every $j \in [1, \ell]$, if the tuple $F'(s_j, a_{j_1}, \ldots, a_{j_m}, 1)$ belongs to J^*, then $\mathcal{A}$ satisfies the subformula S_j whenever we assign $a_{j_1}, \ldots, a_{j_m}$ to $y_1, \ldots, y_m$, and $a_1, \ldots, a_p$ to $x_1, \ldots, x_p$. We prove this by induction on the structure of the subformulas of ψ.

Let h be an homomorphism from $\text{CAN}(I')$ to J^*. We first prove that for every null value $\perp$ in $\text{CAN}(I')$, since $Q(J^*) = \texttt{false}$, it must be the case that $h(\perp) = 1$ or $h(\perp) = 0$. Assume that there is a null value $\perp$ in $\text{CAN}(I')$ such that $h(\perp) \neq 1$ and $h(\perp) \neq 0$.

- Assume first that $\perp$ belongs to a tuple of the form $R'(c_i, a_{i_1}, \ldots, a_{i_{r_{\max}}}, \perp)$ in $\text{CAN}(I')$. From the construction of I' and $\mathcal{M}'$, the following tuples are in $\text{CAN}(I')$
  - $A'(s_a, c_i, d, \ldots, d, 1)$.
  - $F'(c_i, a_1, \ldots, a_1, \perp)$, obtained with (45).
  - $R'(d, a_1, \ldots, a_1, \perp)$, obtained with (45).
  - $V'_k(s_a, a_1)$, for every $1 \leq k \leq p$, obtained with (41).
  - $F'(s_a, a_1, \ldots, a_1, 0)$, obtained with (51).
  - $Z'(s_a, 0, 0, \perp)$, obtained with (45).

It is easy to see that if $h(\perp) \neq 1$ and $h(\perp) \neq 0$, then $Q(J^*) = \texttt{true}$.

- Assume now that $\perp$ belongs to a tuple of the form $F'(s_j, a_{j_1}, \ldots, a_{j_m}, \perp)$ in $\textsc{Can}(I')$. From the construction of I' and $\mathcal{M}'$, the following tuples are in $\textsc{Can}(I')$
  - $A'(s_b, s_j, d, \ldots, d, 1)$.
  - $R'(s_j, a_1, \ldots, a_1, \perp)$, obtained with (46).
  - $R'(d, a_1, \ldots, a_1, \perp)$, obtained with (46).
  - $V'_k(s_b, a_1)$, for every $1 \leq k \leq p$, obtained with (41).
  - $F'(s_b, a_1, \ldots, a_1, 0)$, obtained with (52).
  - $Z'(s_b, 0, 0, \perp)$, obtained with (46).

  It is easy to see that if $h(\perp) \neq 1$ and $h(\perp) \neq 0$, then $Q(J^*) = \texttt{true}$.

We now continue with the proof of the induction.

- For the base case, assume that $S_j = R_i(\bar{x}_j, \bar{y}_j)$, where $\bar{x}_j$ is a tuple of variables in $\{x_1, \ldots, x_p\}$, and $\bar{y}_j$ is a tuple of variables in $\{y_1, \ldots, y_m\}$. Further, assume that the tuple $F'(s_j, a_{j_1}, \ldots, a_{j_m}, 1)$ belongs to J^*. Let $g : \{x_1, \ldots, x_p, y_1, \ldots, y_m\} \to \text{dom}(J')$ be a function such that $g(x_i) = a_i$ for each $i \in [1, p]$, and $g(y_i) = a_{j_i}$ for each $i \in [1, m]$. From the construction of I' and $\mathcal{M}'$, $\textsc{Can}(I')$ contains the following tuples (where $\perp$ is a null value):
  - $A'(s_j, d, \ldots, d, c_i, d, \ldots, d, 1)$.
  - $Z'(s_j, 0, 0, 0)$.
  - $F'(d, a_{j_1}, \ldots, a_{j_m}, 1)$, obtained with (50).
  - $F'(d, a_{j_1}, \ldots, a_{j_m}, 0)$, obtained with (49).
  - $V'_k(s_j, a_k)$ for every $1 \leq k \leq p)$, obtained with (41).
  - $F'(d, a_1, \ldots, a_1, 0)$, obtained with (49).
  - $R'(d, a_1, \ldots, a_1, 0)$, obtained with (47).
  - $R'(c_i, g(\bar{x}_i), g(\bar{y}_i), d, \ldots, d\perp)$, obtained with (45).
  - $R'(d, c_1, \ldots, c_{r_{i'}}, d, \ldots, d, 0)$, for every combination of elements in $\{a_1, \ldots, a_p\}$ and for every $i' \neq i$, obtained with (47).

  We know that $h(\perp) = 0$ or $h(\perp) = 1$. It is easy to see that if $h(\perp) = 0$ then the evaluation of Q' over J^* is true. Then, since $Q(J^*) = \texttt{false}$, it must be the case that $h(\perp) = 1$. It follows that $R'(c_i, g(\bar{x}_j), g(\bar{y}_j), 1)$ belongs to J^*, and, by the definition of $\mathcal{A}$, $\mathcal{A}$ satisfies S_j when we assign $a_1, \ldots, a_p$ to $x_1, \ldots, x_p$ and $a_{j_1}, \ldots, a_{j_m}$ to $y_1, \ldots, y_m$.
- For the inductive case, assume that $S_j \equiv S_{j_1} \wedge S_{j_2}$ (The cases where $S_j \equiv S_{j_1} \wedge S_{j_2}$ or $S_j \equiv \neg S_{j_1}$ are completely symmetrical). Further, assume that $F'(s_j, a_{i_1}, \ldots, a_{i_m}, 1)$ belongs to J^*. We also know there are tuples $F'(s_{j_1}, a_{i_1}, \ldots, a_{i_m}, \perp_{j_1})$ and $F'(s_{j_2}, a_{i_1}, \ldots, a_{i_m}, \perp_{j_2})$ in $\textsc{Can}(I')$. Therefore, J^* contains the tuples $F'(s_{j_1}, a_{i_1}, \ldots, a_{i_m}, h(\perp_{j_1}))$ and $F'(s_{j_2}, a_{i_1}, \ldots, a_{i_m}, h(\perp_{j_2}))$. We claim that $h(\perp_{j_1}) = h(\perp_{j_2}) = 1$. Assume for the sake of contradiction that $h(\perp_{j_1}) = 0$ (the case when $h(\perp_{j_2}) = 0$ is completely symmetrical). From the construction of I' and $\mathcal{M}'$, we also know that the following tuples belong to $\textsc{Can}(I')$, an thus belong to J^*:
  - $A'(s_j, d, \ldots, d, s_{j_1}, s_{j_2}, 1)$.
  - $Z'(s_j, 0, 1, 0)$ and $Z'(s_j, 0, 0, 0)$.
  - $V'_k(s_j, a_k)$ for every $1 \leq k \leq p)$, obtained with (41).
  - $F'(d, a_1, \ldots, a_1, 0)$, obtained with (49).

- $R'(d, a_1, \ldots, a_1, 0)$, obtained with (47).
- $R'(d, c_1, \ldots, c_{r_{i'}}, d, \ldots, d, 0)$, for every combination of elements in $\{a_1, \ldots, a_p\}$ and for every $i' \in [1, n]$, obtained with (47).

It is now easy to see that $Q'(J^*) = \texttt{true}$. Then, it must be that $h(\perp_{j_1}) = h(\perp_{j_2}) = 1$. Then, by the inductive hypothesis, $\mathcal{A}$ satisfies S_{j_1} and S_{j_2} when we assign $a_1, \ldots, a_p$ to $x_1, \ldots, x_p$ and $a_{j_1}, \ldots, a_{j_m}$ to $y_1, \ldots, y_m$. It follows that $\mathcal{A}$ satisfies S_j when we assign $a_1, \ldots, a_p$ to $x_1, \ldots, x_p$ and $a_{j_1}, \ldots, a_{j_m}$ to $y_1, \ldots, y_m$. This finishes the induction.

All that is left to prove is that for every tuple $c_1, \ldots, c_m$ of elements in $\{a_1, \ldots, a_p\}$, the tuple $F'(s_1, c_1, \ldots, c_m, 1)$ belongs to J'. Notice that, from the previous induction, this implies that $\mathcal{A}$ satisfies S_1 for every assignment of the variables $y_1, \ldots, y_m$ when we assign $a_1, \ldots, a_p$ to $x_1, \ldots, x_p$, and thus that $\mathcal{A}$ is a satisfies ϕ.

We now prove that for every tuple $c_1, \ldots, c_m$ of elements in $\{a_1, \ldots, a_p\}$, the tuple $F'(s_1, c_1, \ldots, c_m, 1)$ belongs to J'. Assume for the sake of contradiction that there exists elements $a_{i_1}, \ldots, a_{i_m}$, $a_{i_k} \in \{a_1, \ldots, a_p\}$ for every $1 \leq k \leq m$, such that $F'(s_1, a_{i_1}, \ldots, a_{i_m}, 1)$ does not belong to J^*. We know that there exists a null $\perp$ such that the tuple $F'(s_1, a_{i_1}, \ldots, a_{i_m}, \perp)$ belongs to $\textsc{Can}(I')$. Then, it must be the case that $h(\perp) = 0$, and thus he tuple $F'(s_1, a_{i_1}, \ldots, a_{i_m}, 0)$ must belong to J^*. We also know that the following tuples belong to J^*:

- $A'(s_f, d, \ldots, d, s_1, 0)$
- $Z'(s_f, 0, 0, 1)$
- $V'_i(s_f, a_k)$ for every $1 \leq k \leq p$ and every $1 \leq i \leq p$, obtained with (41).
- $F'(d, a_1, \ldots, a_1, 1)$, obtained with (50).
- $R'(d, a_1, \ldots, a_1, 1)$, obtained with (48).
- $R'(d, c_1, \ldots, c_{r_{i'}}, d, \ldots, d, 0)$, for every combination of elements in $\{a_1, \ldots, a_p\}$ and for every $i' \in [1, n]$, obtained with (47).
- $F'(d, a_{i_1}, \ldots, a_{i_m}, 0)$, obtained with (49).

It is easy to see that $Q(J^*) = \texttt{true}$, which is a contradiction. This concludes the proof of the theorem.

References

1. Abiteboul, S., Duschka, O.: Answering queries using materialized views. Gemo report 383
2. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
3. Afrati, F.N., Li, C., Pavlaki, V.: Data exchange in the presence of arithmetic comparisons. In: Proceedings of the 11th International Conference on Extending Database Technology (EDBT), pp. 487–498 (2008)
4. Arenas, M., Barceló, P., Fagin, R., Libkin, L.: Locally consistent transformations and query answering in data exchange. In: Proceedings of the 23rd ACM Symposium on Principles of Database Systems (PODS), pp. 229–240 (2004)
5. Beeri, C., Vardi, M.Y.: A proof procedure for data dependencies. J. ACM **31**(4), 718–741 (1984)
6. Börger, E., Grädel, E., Gurevich, Y.: The Classical Decision Problem. Springer, Berlin (2001)
7. Deutsch, A., Nash, A., Remmel, J.B.: The chase revisited. In: Proceedings of the 27th ACM Symposium on Principles of Database Systems (PODS), pp. 149–158 (2008)
8. Fagin, R., Kolaitis, P., Popa, L., Tan, W.C.: Composing schema mappings: Second-order dependencies to the rescue. In: Proceedings of the 23rd ACM Symposium on Principles of Database Systems (PODS), pp. 83–94 (2004)

9. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. Theor. Comput. Sci. **336**(1), 89–124 (2005)
10. Fagin, R., Kolaitis, P.G., Popa, L.: Data exchange: getting to the core. ACM Trans. Database Syst. **30**(1), 174–210 (2005)
11. Gottlob, G., Papadimitriou, C.: On the complexity of single-rule datalog queries. Inf. Comput. **183**(1), 104–122 (2003)
12. Greenlaw, R., Hoover, H.J., Ruzzo, W.L.: Limits to Parallel Computation: P-Completeness Theory. Oxford University Press, London (1995)
13. Kolaitis, P.: Schema mappings, data exchange, and metadata management. In: Proceedings of the 24th ACM Symposium on Principles of Database Systems (PODS), pp. 61–75 (2005)
14. Kolaitis, P., Panttaja, J., Tan, W.-C.: The complexity of data exchange. In: Proceedings of the 25th ACM Symposium on Principles of Database Systems (PODS), pp. 30–39 (2006)
15. Imielinski, T., Lipski, W.: Incomplete information in relational databases. J. ACM **31**, 761–791 (1984)
16. Lenzerini, M.: Data integration: A theoretical perspective. In: Proceedings of the 21st ACM Symposium on Principles of Database Systems (PODS), pp. 233–246 (2002)
17. Libkin, L.: Elements of Finite Model Theory. Springer, Berlin (2004)
18. Libkin, L.: Data exchange and incomplete information. In: Proceedings of the 25th ACM Symposium on Principles of Database Systems (PODS), pp. 60–69 (2006)
19. Libkin, L., Sirangelo, C.: Data exchange and schema mappings in open and closed worlds. In: Proceedings of the 27th ACM Symposium on Principles of Database Systems (PODS), pp. 139–148 (2008)
20. Mądry, A.: Data exchange: On the complexity of answering queries with inequalities. Inf. Process. Lett. **94**(6), 253–257 (2005)
21. Papadimitriou, C.H.: Computational Complexity. Addison Wesley, Reading (1994)
22. Vardi, M.Y.: The complexity of relational query languages. In: Proceedings of the 14th ACM Symposium on Theory of Computing (STOC), pp. 137–146 (1982)