EXTENDING QUERY LANGUAGES FOR DATA EXCHANGE

JUAN REUTTER DE LA MAZA

Thesis submitted to the Office of Research and Graduate Studies in partial fulfillment of the requirements for the degree of Master of Science in Engineering

Advisor:
MARCELO ARENAS

Santiago de Chile, March 2009

© MMIX, JUAN REUTTER DE LA MAZA
EXTENDING QUERY LANGUAGES FOR DATA EXCHANGE

JUAN REUTTER DE LA MAZA

Members of the Committee:
MARCELO ARENAS
YADRAN ETEROVIC
PABLO BARCELÓ
JORGE VERA

Thesis submitted to the Office of Research and Graduate Studies in partial fulfillment of the requirements for the degree of Master of Science in Engineering

Santiago de Chile, March 2009
A mi familia, a Francisca
ACKNOWLEDGMENTS

I would like to thank Marcelo Arenas, my advisor, and Pablo Barceló, from Universidad de Chile, for their help, patience and dedication, I am sure they put their best efforts in explaining and teaching me many useful tools and advices, for which I am most grateful. Additional thanks to Jorge Pérez for his helpful comments and guidance on some difficult proofs.
# TABLE OF CONTENTS

Acknowledgments ................................................. iv

LIST OF TABLES ................................................ vii

ABSTRACT ......................................................... viii

RESUMEN ........................................................... ix

1. Introduction .................................................... 1
   1.1. Data Exchange ............................................ 1
   1.2. Summary of Contributions ................................. 3
   1.3. Thesis Outline/Document Organization ................... 6

2. Background ................................................... 7
   2.1. Preliminaries ............................................. 7
   2.2. Data exchange settings and solutions .................... 7
   2.3. Universal solutions and canonical universal solution .......... 9
   2.4. Certain answers ......................................... 10

3. Extending Query Languages for Data Exchange: DATALOG\(\text{C}(\neq)\) Programs . . . . . 12
   3.1. DATALOG\(\text{C}(\neq)\) Programs ............................... 13

4. Complexity and Expressiveness of DATALOG\(\text{C}(\neq)\) programs .................... 16

5. Conjunctive Queries with two Inequalities ........................ 29

6. Combined Complexity of Data Exchange .......................... 46
   6.1. Combined Complexity of DATALOG\(\text{C}(\neq)\) programs .................. 46
   6.2. Combined Complexity of CQ\(\neq\) .................................. 48

7. Conclusion and future Research .................................. 69
   7.1. General Remarks ........................................... 69
   7.2. Future Research Topics ..................................... 69
REFERENCES ..................................................... 70

APPENDIX A. Additional Proofs ...................................... 73
  Proof of Proposition 5.3 ........................................... 73
  Proof of Theorem 5.2 ............................................... 75
  Proof of Proposition 6.2 ........................................... 83
  Proof of Proposition 6.3 ........................................... 90
LIST OF TABLES

6.1 Combined complexity of computing certain answers. . . . . . . . . . . . . . 48
ABSTRACT

Data exchange is the problem of taking data structured under a source schema and creating an instance of a target schema that reflects as accurately as possible the source data.

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 (data complexity). But this is not the only class with this property; the certain answers to any DATALOG program can also 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 work, we propose a language called DATALOGC(≠) 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 DATALOGC(≠) program can be computed in polynomial time (data complexity), and that every union of conjunctive queries with at most one inequality or negated relational atom per disjunct, can be rewritten as a DATALOGC(≠) 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 DATALOGC(≠) programs and other related query languages. In particular, we show that this problem is EXPTIME-complete for DATALOGC(≠), even if one restricts to conjunctive queries with single inequalities, which is a fragment of DATALOGC(≠) by the result mentioned above.

Keywords: Data Exchange, Datalog, Query, Database Theory.
RESUMEN

El problema del Data Exchange consiste en tomar datos estructurados bajo un esquema fuente y crear una instancia de un esquema destino que refleje lo más adecuadamente posible los datos fuente.

La clase de unión de consultas conjuntivas (UCQ) se comporta particularmente bien en el entorno de Data Exchange; sus respuestas certeras se pueden computar en tiempo polinomial (complejidad de datos). Pero esta no es la única clase con esa propiedad: las respuestas certeras para cualquier programa en DATALOG también pueden ser computadas en tiempo polinomial. El problema es que tanto UCQ como DATALOG no permiten la negación de átomos, pues el añadir negación no restringida vuelve intratable el problema.

En este trabajo, proponemos un lenguaje llamado DATALOG\textsuperscript{C(\neq)}, que extiende al lenguaje DATALOG con una forma restringida de negación, y estudiamos algunas de sus propiedades fundamentales. En particular, mostramos que las respuestas certeras a programas DATALOG\textsuperscript{C(\neq)} pueden ser computadas en tiempo polinomial (complejidad de datos), y que toda unión de consultas conjuntivas con a lo más una desigualdad o átomo negado por disyunción puede ser reescrita como un programa en DATALOG\textsuperscript{C(\neq)}. Más aún, mostramos que este también es el caso para una restricción sintáctica de la clase de uniones de consultas conjuntivas con a lo más dos desigualdades por disyunción. Esta restricción es óptima, pues el computar respuestas certeras se vuelve intratable al remover cualesquiera de ellas. Finalmente, proveemos de un análisis detallado de la complejidad combinada de computar las respuestas certeras a un programa en DATALOG\textsuperscript{C(\neq)} y otros lenguajes de consulta relacionados. En particular, mostramos que este problema es EXPTIME-completo para DATALOG\textsuperscript{C(\neq)}, incluso si se restringe a consultas conjuntivas con una desigualdad, que es un fragmento de DATALOG\textsuperscript{C(\neq)} por los resultados enunciados anteriormente.

Palabras Claves: Data Exchange, Datalog, Consultas, Base de Datos.
1. INTRODUCTION

1.1. Data Exchange

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. Data exchange is considered to be an old and relevant database problem, and it is used in many tasks that require transferring information between independent applications. Nevertheless, the theoretical foundations of data exchange have only been laid out very recently by the seminal work of Fagin, Kolaitis, Miller and Popa (2005).

As an example, consider a biologic laboratory A that is currently analyzing a series of proteins and whose data is stored under a particular schema. Should the need arise to compare the results with a new protein, it could be more efficient to borrow the results from another laboratory, say B, that has already analyzed this protein. Unfortunately, since both labs are independent, they probably have their data structured under different schemas. Therefore, even if B give access to their data to A, the latter may be unable to find an appropriate answer for their queries. The problem then consists in how to restructure the data received from B in order to answer their queries correctly.

The first attempts to build a system that supported data exchange where made decades ago with the construction of the EXPRESS system (Shu, Housel, Taylor, Ghosh, & Lum, 1977). Nowadays, with the proliferation of web data in its various forms and the emergence of e-business applications that need to communicate data yet remain autonomous, the need for data exchange has steadily increased. In this context, 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. (Fagin, Kolaitis, Miller, & Popa, 2005; Fagin, Kolaitis, & Popa, 2005; Arenas, Barceló, Fagin, & Libkin, 2004; Fagin, Kolaitis, Popa, & Tan, 2005; Libkin, 2006; Kolaitis, Panttaja, & Tan, 2006; Kolaitis, 2005)). Furthermore, fueled by the incorporation of the Clio system into IBM’s software for database management (Haas, Hernández, Ho, Popa, & Roth, 2005), data exchange is slowly earning its place in the computer software industry.
Following the foundation laid by Fagin, Kolaitis, Miller and Popa (2005), in formal terms a data exchange setting is a triple $\mathcal{M} = (S, T, \Sigma_{st})$, where $S$ is a source schema, $T$ is a target schema, and $\Sigma_{st}$ is a mapping defined as a set of source-to-target dependencies (Fagin et al, 2005) that specifies how to map data from $S$ to $T$. 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 $\Sigma_{st}$.

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 (Fagin, Kolaitis, Miller, & Popa, 2005) 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 (Fagin, Kolaitis, Miller, & Popa, 2005) that for the aforementioned data exchange settings, 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 $\Sigma_{st}$ is assumed to be fixed. Thus, this result is stated in terms of data complexity (Vardi, 1982).

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 (Imielinski & Lipski, 1983; Abiteboul & Duschka, 1999; Lenzerini, 2002; Fagin, Kolaitis, Miller, & Popa, 2005). More formally, given a data exchange setting $\mathcal{M} = (S, T, \Sigma_{st})$ and a query $Q$ over $T$, 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 (2005) 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 $\Sigma_{st}$ 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 is as good as UCQ for data exchange purposes. In particular, the certain answers to a DATALOG program $\Pi$ over a source instance $I$ can also be computed efficiently by evaluating $\Pi$ over the canonical universal solution for $I$.

1.2. Summary of Contributions

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 work 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 expressibility 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 (Abiteboul & Duschka, 1999; Fagin, Kolaitis, Miller, & Popa, 2005). In this respect, the following are our main contributions.
• We introduce a query language called DATALOG\(_{\neq}\) 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 DATALOG\(_{\neq}\) program \(\Pi\) over a source instance \(I\) can be computed by evaluating \(\Pi\) over the canonical universal solution for \(I\). As a corollary, we obtain that computing certain answers to a DATALOG\(_{\neq}\) program can be done in polynomial time (in terms of data complexity).

• To show that DATALOG\(_{\neq}\) 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 DATALOG\(_{\neq}\) 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 (Fagin, Kolaitis, Miller, & Popa, 2005)), the use of DATALOG\(_{\neq}\) 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\(_{\neq}\) 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 in terms of data complexity (Abiteboul & Duschka, 1999; Fagin, Kolaitis, Miller, & Popa, 2005). In fact, it has been shown that the intractability holds even for the case of two inequalities (Madry, 2005). However, very little is known about tractable fragments of this class. In our research, we have found a syntactic restriction for the class of unions of conjunctive queries with at most two inequalities per disjunct, and proven that every query conforming to it can be expressed as a DATALOG\(_{\neq}\) 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 \( \text{coNP}\)-complete.

The study of the complexity of computing certain answers to \( \text{DATALOG}_{\neq}^C \) programs will not be complete if one does not consider the notion of \textit{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 (Vardi, 1982), 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 work, we study this problem and establish the following results.

- We show that the combined complexity of the problem of computing certain answers to \( \text{DATALOG}_{\neq}^C \) programs is \( \text{EXPTIME}\)-complete, even if one restricts to the class of conjunctive queries with single inequalities (which is a fragment of \( \text{DATALOG}_{\neq}^C \) by the result mentioned above). This refines a result in (Kolaitis et al., 2006) that shows that the combined complexity of the problem of computing certain answers to \textit{unions} of conjunctive queries with at most one inequality per disjunct is \( \text{EXPTIME}\)-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 \( \text{CONEXPTIME}\)-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 (Lenzerini, 2002), which are frequently used in practice. For the case of DATALOG\(^C(\neq)\) 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 this thesis that if one focus 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 \(\Pi^p_2\)-complete for the case of conjunctive queries with \(k\) inequalities, for every \(k \geq 2\).

1.3. Thesis Outline/Document Organization

In chapter 2 we introduce the terminology and some previous results that will be used in the thesis. Chapter 3 contains the syntax and semantics of DATALOG\(^C(\neq)\) programs. In chapter 4, we study some of the fundamental properties of DATALOG\(^C(\neq)\) programs concerning complexity and expressiveness. In chapter 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 per disjunct. Chapter 6 contains a thorough analysis of the combined complexity of computing certain answers to DATALOG\(^C(\neq)\) programs and other related query languages. Conclusions are in chapter 7.
2. BACKGROUND

2.1. Preliminaries

A schema \( R \) is a finite set \( \{R_1, \ldots, R_k\} \) of relation symbols, with each \( R_i \) having a fixed arity \( n_i > 0 \). Let \( D \) be a countably infinite domain. An instance \( I \) of \( R \) assigns to each relation symbol \( R_i \) of \( R \) a finite \( n_i \)-ary relation \( R^I_i \subseteq D^{n_i} \). The domain \( \text{dom}(I) \) 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 some of its extensions (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, \( \text{CQ}^{\neq} \) is the class of conjunctive queries with inequalities, and \( \text{UCQ}^{\neg} \) is the class of unions of conjunctive queries with negation. As usual in the database literature, we assume that the negated relational atoms and inequalities that appear in conjunctive queries are safe, in the sense that every variable that appears in a negated relational atom or an inequality in a query, also appears in a non-negated relational atom in the query.

We also assume familiarity with the theory of computational complexity.

2.2. Data exchange settings and solutions

As is customary in the data exchange literature, we consider instances with two types of values: constants and nulls (Fagin, Kolaitis, Miller, & Popa, 2005; Fagin, Kolaitis, & Popa, 2005). More precisely, let \( C \) and \( N \) be infinite and disjoint sets of constants and nulls, respectively, and assume that \( D = C \cup N \). If we refer to a schema \( S \) as a source schema, then for every instance \( I \) of \( S \), it holds that \( \text{dom}(I) \subseteq C \). On the other hand, if we refer to a schema \( T \) as a target schema, then for every instance \( J \) of \( T \), it holds that \( \text{dom}(J) \subseteq C \cup N \). Slightly abusing notation, we use \( C(\cdot) \) to denote a built-in unary predicate such that \( C(a) \) holds if and only if \( a \) is a constant, that is \( a \in C \).
A data exchange setting is a tuple $\mathcal{M} = (S, T, \Sigma_{st})$, where $S$ is a source schema, $T$ is a target schema, $S$ and $T$ do not have predicate symbols in common and $\Sigma_{st}$ is a set of FO-dependencies over $S \cup T$ (in Fagin, Kolaitis, Miller, & Popa, 2005) and (Fagin, Kolaitis, & Popa, 2005) a more general class of data exchange settings is presented, that also includes target dependencies. As usual in the data exchange literature (e.g., (Fagin, Kolaitis, Miller, & Popa, 2005; Fagin, Kolaitis, & Popa, 2005)), we restrict the study to data exchange settings in which $\Sigma_{st}$ consists of a set of source-to-target dependencies. A source-to-target dependency (std) is a dependency 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 $S$ and $\psi(\bar{x}, \bar{y})$ is a conjunction of relational atoms over $T$. A source (resp. target) instance $K$ for $\mathcal{M}$ is an instance of $S$ (resp. $T$). 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 (Lenzerini, 2002). 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) (Lenzerini, 2002) is a data exchange setting $\mathcal{M} = (S, T, \Sigma_{st})$, in which every std in $\Sigma_{st}$ is of the form $\forall \bar{x} \ (S(\bar{x}) \rightarrow \psi(\bar{x}))$, for some $S \in S$.

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

Example 2.1. Let $\mathcal{M} = (S, T, \Sigma_{st})$ be a data exchange setting. Assume that $S$ consists of one binary relation symbol $P$, and $T$ consists of two binary relation symbols $Q$ and $R$. Further, assume that $\Sigma_{st}$ contains stds $P(x, y) \rightarrow Q(x, y)$ 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 $n \in \mathbb{N}$, are solutions for $I$. In fact, $I$ has infinitely many solutions.  \qed


2.3. Universal solutions and canonical universal solution

It has been argued in (Fagin, Kolaitis, Miller, & Popa, 2005) 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 $R$. A homomorphism $h$ from $K_1$ to $K_2$ is a function $h : \text{dom}(K_1) \rightarrow \text{dom}(K_2)$ such that: (1) $h(c) = c$ for every $c \in \mathbb{C} \cap \text{dom}(K_1)$, and (2) for every $R \in R$ 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 $M$ be a data exchange setting, $I$ a source instance and $J$ a solution for $I$ under $M$. Then $J$ is a universal solution for $I$ under $M$, if for every solution $J'$ for $I$ under $M$, 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 (Fagin, Kolaitis, Miller, & Popa, 2005) 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 (Beeri & Vardi, 1984). We shall define canonical universal solutions directly as in (Arenas et al., 2004; Libkin, 2006).

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

$$\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{w} (T_1(\bar{x}_1, \bar{w}_1) \land \cdots \land T_k(\bar{x}_k, \bar{w}_k)),$$

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 $\mathbb{N}$. 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.

This definition differs from the one given in (Fagin, Kolaitis, Miller, & Popa, 2005), where a canonical universal solution is obtained by using the classical chase procedure. Since the result of the chase used in (Fagin, Kolaitis, Miller, & Popa, 2005) is not necessarily unique (it depends on the order in which the chase steps are applied), 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 $\mathcal{M} = (S, T, \Sigma_{st})$, we denote by $\text{CAN}$ the transformation from source instances to target instances, such that $\text{CAN}(I)$ is the canonical universal solution for $I$ under $\mathcal{M}$.

2.4. Certain answers

Queries in a data exchange setting $\mathcal{M} = (S, T, \Sigma_{st})$ are posed over the target schema $T$. 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 (Imielinski & Lipski, 1983; Abiteboul & Duschka, 1999; Lenzerini, 2002; Fagin, Kolaitis, Miller, & Popa, 2005).

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 $\text{certain}_{\mathcal{M}}(Q, I) = \text{true}$ iff $Q$ holds for every $J$ that is a solution for $I$ under $\mathcal{M}$. We write $\text{certain}_{\mathcal{M}}(Q, I) = \text{false}$ if it is not the case that $\text{certain}_{\mathcal{M}}(Q, I) = \text{true}$.

Let $\mathcal{M} = (S, T, \Sigma_{st})$ be a data exchange setting and $Q$ a query over $T$. 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 $\bar{t} \in \text{certain}_M(Q, I)$?

Since in the above definition both the setting and the query are fixed, it corresponds (in terms of Vardi’s taxonomy (Vardi, 1982)) to the data complexity of the problem of computing certain answers. Later, in chapter 6, we also study the combined complexity of this problem.
3. EXTENDING QUERY LANGUAGES FOR DATA EXCHANGE: DATALOG^{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 (Fagin, Kolaitis, Miller, & Popa, 2005). 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 $\text{CERTAIN-ANSWERS}(\mathcal{M},Q)$ 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. This is justified by the fact that negation can be used to express interesting properties in data exchange, as shown in the following example.\footnote{Note that this is not immediately evident as, for instance, for every single conjunctive query $Q$ with at least one negated relational atom, it holds that certain$_{\mathcal{M}}(Q,I) = \text{false}$ for every source instance $I$.}

**Example 3.1.** Consider a data exchange setting with $S = \{E(\cdot, \cdot), A(\cdot), B(\cdot)\}$, $T = \{G(\cdot, \cdot), P(\cdot), R(\cdot)\}$ 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 UCQ$^\neg$ query over $T$: $\exists x \exists y (P(x) \land R(y) \land G(x,y)) \lor \exists x \exists y \exists z (G(x,z) \land G(z,y) \land \neg G(x,y))$. It is not hard to prove that for every source instance $I$, certain$_M(Q,I) = \text{true}$ 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, certain$_M(Q,I) = \text{true}$ 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. (Libkin, 2004)) 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 $M$ are as in the previous example, then there is no union of conjunctive queries $Q'$ such that $Q'(\text{CAN}(I)) = \text{certain}_M(Q',I) = \text{certain}_M(Q,I)$, 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 (Arenas et al., 2004)). And not only that it follows from (Fagin, Kolaitis, Miller, & Popa, 2005) that inequalities also add expressive power to UCQ in the context of data exchange.

In this chapter, 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. DATALOG$^{G(\neq)}$ 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 (Abiteboul & Duschka, 1999; Fagin, Kolaitis, Miller, & Popa, 2005). More precisely, there is a setting $M$ and a query $Q$ in UCQ$^\neq$ such that the problem $\text{CERTAIN-ANSWERS}(M,Q)$ cannot be solved in polynomial time (unless $\text{PTIME} = \text{NP}$). 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 chapter 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 3.1** (DATALOGC(≠) programs). A constant-inequality Datalog rule is a rule of the form:

\[
S(\bar{x}) \leftarrow S_1(\bar{x}_1), \ldots, S_\ell(\bar{x}_\ell), C(y_1), \ldots, C(y_m), u_1 \neq v_1, \ldots, u_n \neq v_n, \tag{3.1}
\]

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 \( y_i \) (\( i \in [1, m] \)).

A constant-inequality Datalog program (DATALOGC(≠) program) \( \Pi \) is a finite set of constant-inequality Datalog rules.

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

\[
R(x, y) \leftarrow T(x, z), S(z, y), C(x), C(z), x \neq z
\]

\[
S(x) \leftarrow U(x, u, v, w), C(x), C(u), C(v), C(w), u \neq v, u \neq w
\]

For a rule of the form (3.1), we say that \( S(\bar{x}) \) is its head. The set of predicates of a program \( \Pi \), denoted by \( \text{Pred}(\Pi) \), is the set of predicate symbols mentioned in \( \Pi \), while the set of intensional predicates of \( \Pi \), denoted by \( \text{IPred}(\Pi) \), is the set of predicates symbols \( R \in \text{Pred}(\Pi) \) such that \( R(\bar{x}) \) appears as the head of some rule of \( \Pi \).

Assume that \( \Pi \) is a DATALOGC(≠) program and \( I \) is a database instance of the relational schema \( \text{Pred}(\Pi) \). Then \( T(I) \) is an instance of \( \text{Pred}(\Pi) \) such that for every \( R \in \text{Pred}(\Pi) \) and every tuple \( \bar{t} \), it holds that \( \bar{t} \in R^{T(I)} \) if and only if there exists a rule \( R(\bar{x}) \leftarrow R_1(\bar{x}_1), \ldots, R_\ell(\bar{x}_\ell), C(y_1), \ldots, C(y_m), u_1 \neq v_1, \ldots, u_n \neq v_n \) in \( \Pi \) and a variable assignment
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 \( T \) is used to define the semantics of constant-inequality Datalog programs. More precisely, define \( T^0_\Pi(I) \) to be \( I \) and \( T^{n+1}_\Pi(I) \) to be \( T(T^n_\Pi(I)) \cup T^n_\Pi(I) \), for every \( n \geq 0 \). Then the evaluation of \( \Pi \) over \( I \) is defined as \( T^\infty_\Pi(I) = \bigcup_{n \geq 0} T^n_\Pi(I) \).

A constant-inequality Datalog program \( \Pi \) is said to be defined over a relational schema \( R \) if \( R = Pred(\Pi) \setminus IPred(\Pi) \) and \( Answer \in IPred(\Pi) \). Given an instance \( I \) of \( R \) and a tuple \( \bar{t} \) in \( \text{dom}(I)^n \), where \( n \) is the arity of \( Answer \), we say that \( \bar{t} \in \Pi(I) \) if \( \bar{t} \in Answer^{T^\infty_\Pi(I_0)} \), where \( I_0 \) is an extension of \( I \) defined as: \( R^{I_0} = R^I \) for \( R \in R \) 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\((\neq)\) programs, we have that these programs are preserved under homomorphisms. From this we conclude that the certain answers to a DATALOG\((\neq)\) program \( \Pi \) can be computed by directly evaluating \( \Pi \) over a universal solution.

**Proposition 3.1.** Let \( \mathcal{M} = (S, T, \Sigma_{st}) \) be a data exchange setting, \( I \) a source instance, \( J \) a universal solution for \( I \) under \( \mathcal{M} \), and \( \Pi \) a DATALOG\((\neq)\) program over \( T \). 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 chapter 4 to show that DATALOG\((\neq)\) programs preserve the good properties of conjunctive queries for data exchange.
4. COMPLEXITY AND EXPRESSIVENESS OF DATALOG\(^C(\neq)\) PROGRAMS

We start this chapter by studying the expressive power of DATALOG\(^C(\neq)\) 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 is one of the few known extensions of the class UCQ for which the problem of computing certain answers is tractable (Fagin, Kolaitis, Miller, & Popa, 2005).

**Theorem 4.1.** Let \( Q \) be a UCQ query over a schema \( T \), with at most one inequality or negated relational atom per disjunct. Then there exists a DATALOG\(^C(\neq)\) program \( \Pi_Q \) over \( T \) such that for every data exchange setting \( \mathcal{M} = (S, T, \Sigma_{st}) \) and instance \( I \) of \( S \), \( \text{certain}_\mathcal{M}(Q, I) = \text{certain}_\mathcal{M}(\Pi_Q, I) \). Moreover, \( \Pi_Q \) 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 4.1.** Let \( \mathcal{M} \) be a data exchange setting such that \( S = \{E(\cdot, \cdot), A(\cdot)\}, \ T = \{G(\cdot, \cdot), P(\cdot)\} \) and
\[
\Sigma_{st} = \{ E(x,y) \rightarrow \exists z (G(x,z) \land G(z,y)), \ A(x) \rightarrow P(x) \}.
\]
Also, let \( Q(x) \) be the following query in UCQ\(^\neq\neg\):
\[
(P(x) \land G(x,x)) \lor \exists y (G(x,y) \land x \neq y) \lor \exists y \exists z (G(x,z) \land G(z,y) \land \neg G(x,y))
\]
We build a DATALOG\(^C(\neq)\) program \( \Pi_Q \) such that \( \text{certain}_\mathcal{M}(Q, I) = \text{certain}_\mathcal{M}(\Pi_Q, I) \). The set of intensional predicates of the DATALOG\(^C(\neq)\) program \( \Pi_Q \) is \( \{U_1(\cdot, \cdot, \cdot), U_2(\cdot, \cdot), \text{dom}(\cdot), \text{EQUAL}(\cdot, \cdot, \cdot), \text{ANSWER}(\cdot)\} \). The program \( \Pi_Q \) over \( T \) 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 $T$ where $\Pi_Q$ is evaluated:

$$\text{dom}(x) \leftarrow G(x, z) \quad (4.1)$$
$$\text{dom}(x) \leftarrow G(z, x) \quad (4.2)$$
$$\text{dom}(x) \leftarrow P(x) \quad (4.3)$$

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

$$\text{EQUAL}(x, x, z) \leftarrow \text{dom}(x), \text{dom}(z) \quad (4.4)$$
$$\text{EQUAL}(x, y, z) \leftarrow \text{EQUAL}(y, x, z) \quad (4.5)$$
$$\text{EQUAL}(x, y, z) \leftarrow \text{EQUAL}(x, w, z), \text{EQUAL}(w, y, z) \quad (4.6)$$

Predicate $\text{EQUAL}$ 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 $\text{EQUAL}$ is reflexive, as $\text{DATALOG}^C(\neq)$ programs are safe, i.e. every variable that appears in the head of a rule also has to appear in its body.

• Third, $\Pi_Q$ includes the rules:

$$U_1(x, y, z) \leftarrow G(x, y), \text{dom}(z) \quad (4.7)$$
$$U_2(x, z) \leftarrow P(x), \text{dom}(z) \quad (4.8)$$
$$U_1(x, y, z) \leftarrow U_1(u, v, z), \text{EQUAL}(u, x, z), \text{EQUAL}(v, y, z) \quad (4.9)$$
$$U_2(x, z) \leftarrow U_2(u, z), \text{EQUAL}(u, x, z) \quad (4.10)$$

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 $\Pi_Q$ is evaluated. The last two rules allow to replace equal elements in the interpretation of $U_1$ and $U_2$. 
• Fourth, $\Pi_Q$ includes the following rule for the third disjunct of $Q(x)$:

$$U_1(x, y, x) \leftarrow U_1(x, z, x), U_1(z, y, x)$$ (4.11)

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, $\Pi_Q$ includes the following rule for the second disjunct of $Q(x)$:

$$\text{EQUAL}(x, y, x) \leftarrow U_1(x, y, x)$$ (4.12)

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, $\Pi_Q$ includes two rules for collecting the certain answers to $Q(x)$:

$$\text{ANSWER}(x) \leftarrow U_2(x, x), U_1(x, x, x), C(x)$$ (4.13)

$$\text{ANSWER}(x) \leftarrow \text{EQUAL}(y, z, x), C(y), C(z), y \neq z$$ (4.14)

Intuitively, rule (4.13) 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 (4.14) says that if in the process of evaluating $\Pi_Q$ with parameter $a$, two distinct constants $b$ and $c$ are declared to be equal (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 (4.1), we have that dom$(a)$ holds in $\text{CAN}(I)$. Thus, we conclude by applying rule (4.7) that $U_1(a, n, a)$ and $U_1(n, b, a)$ hold in $\text{CAN}(I)$ and,
therefore, we obtain by using rule (4.12) 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 (4.5), 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 (4.2) and (4.4)). Thus, by applying rule (4.9), we obtain that $U_1(a, b, a)$ holds in $\text{CAN}(I)$. Therefore, by applying rule (4.12) again, we obtain that $\text{EQUAL}(a, b, a)$ holds in $\text{CAN}(I)$. This time, rule (4.12) tries to prove that $a$ is not in the certain answers to $Q(x)$ by forcing constants $a$ and $b$ to be the same values. But this cannot be the case and, thus, rule (4.14) 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 $h : \text{CAN}(I) \to J$. 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 $\Pi_Q$, for every source instance $I$. □

We now present the proof of Theorem 4.1. Proof: Assume that $T = \{T_1, \ldots, T_k\}$, where each $T_i$ has arity $n_i > 0$, and that $Q(\vec{x}) = Q_1(\vec{x}) \lor \cdots \lor Q_\ell(\vec{x})$, where $\vec{x} = (x_1, \ldots, x_m)$ and each $Q_i(\vec{x})$ is a conjunctive query with at most one inequality or negated relational atom. Then the set of intensional predicates of $\text{DATALOG}^{C(\neq)}$ program $\Pi_Q$ is

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

where each $U_i$ ($i \in [1, k]$) has arity $n_i + m$, $\text{DOM}$ has arity 1, $\text{EQUAL}$ has arity $2 + m$ and $\text{ANSWER}$ has arity $m$. Moreover, the set of rules of $\Pi_Q$ is defined as follows.
• For every predicate $T_i \in T$, $\Pi_Q$ includes the following $k$ rules:

\[
\begin{align*}
\text{DOM}(x) & \leftarrow T_i(x, y_2, y_3, \ldots, y_{n_i-1}, y_{n_i}) \\
\text{DOM}(x) & \leftarrow T_i(y_1, x, y_3, \ldots, y_{n_i-1}, y_{n_i}) \\
& \quad \ldots \\
\text{DOM}(x) & \leftarrow T_i(y_1, y_2, y_3, \ldots, y_{n_i-1}, x)
\end{align*}
\]

• $\Pi_Q$ includes the following rules for predicate \texttt{EQUAL}:

\[
\begin{align*}
\text{EQUAL}(x, x, z_1, \ldots, z_m) & \leftarrow \text{DOM}(x), \text{DOM}(z_1), \ldots, \text{DOM}(z_m) \\
\text{EQUAL}(x, y, z_1, \ldots, z_m) & \leftarrow \text{EQUAL}(y, x, z_1, \ldots, z_m) \\
\text{EQUAL}(x, y, z_1, \ldots, z_m) & \leftarrow \text{EQUAL}(x, w, z_1, \ldots, z_m), \text{EQUAL}(w, y, z_1, \ldots, z_m)
\end{align*}
\]

• For every predicate $U_i$, $\Pi_Q$ includes the following rules:

\[
\begin{align*}
U_i(y_1, \ldots, y_{n_i}, z_1, \ldots, z_m) & \leftarrow T_i(y_1, \ldots, y_{n_i}), \text{DOM}(z_1), \ldots, \text{DOM}(z_m) \\
U_i(y_1, \ldots, y_{n_i}, z_1, \ldots, z_m) & \leftarrow U_i(w_1, \ldots, w_{n_i}, z_1, \ldots, z_m), \\
& \quad \text{EQUAL}(w_1, y_1, z_1, \ldots, z_m), \ldots, \\
& \quad \text{EQUAL}(w_{n_i}, y_{n_i}, z_1, \ldots, z_m)
\end{align*}
\]

• 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 $\exists \bar{u} (T_{p_1}(\bar{u}_1) \land \cdots \land T_{p_n}(\bar{u}_n))$, 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 $\Pi_Q$ includes the following rule:

\[
\text{ANSWER}(\bar{x}) \leftarrow U_{p_1}(\bar{u}_1, \bar{x}), \ldots, U_{p_n}(\bar{u}_n, \bar{x}), C(x_1), \ldots, C(x_m) \tag{4.15}
\]

Notice that this rule is well defined since the set $\bar{x}$ is the set of free variables of $\exists \bar{u} (T_{p_1}(\bar{u}_1) \land \cdots \land T_{p_n}(\bar{u}_n))$. Second, assume that $Q_i(\bar{x})$ contains a negated relational atom. Then $Q_i(\bar{x})$ is equal to $\exists \bar{u} (T_{p_1}(\bar{u}_1) \land \cdots \land T_{p_n}(\bar{u}_n) \land \neg T_{p_{n+1}}(\bar{u}_{n+1}))$, 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 $\Pi_Q$ includes the following rule:

$$U_{p_{n+1}}(\bar{u}_{n+1}, \bar{x}) \leftarrow U_{p_1}(\bar{u}_1, \bar{x}), \ldots, U_{p_n}(\bar{u}_n, \bar{x}). \quad (4.16)$$

This rule is well defined since $\exists \bar{u} (T_{p_1}(\bar{u}_1) \land \cdots \land T_{p_n}(\bar{u}_n) \land \neg T_{p_{n+1}}(\bar{u}_{n+1}))$ is a safe query. Finally, assume that $Q_i(\bar{x})$ contains an inequality. Then $Q_i(\bar{x})$ is equal to $\exists \bar{u} (T_{p_1}(\bar{u}_1) \land \cdots \land T_{p_n}(\bar{u}_n) \land v_1 \neq v_2)$, 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 $\Pi_Q$ includes the following rules:

$$\text{EQUAL}(v_1, v_2, \bar{x}) \leftarrow U_{p_1}(\bar{u}_1, \bar{x}), \ldots, U_{p_n}(\bar{u}_n, \bar{x}) \quad (4.17)$$

$$\text{ANSWER}(\bar{x}) \leftarrow \text{EQUAL}(u, v, \bar{x}), C(u), C(v), \quad u \neq v, C(x_1), \ldots, C(x_m) \quad (4.18)$$

We note that the first rule above is well defined since $\exists \bar{u} (T_{p_1}(\bar{u}_1) \land \cdots \land T_{p_n}(\bar{u}_n) \land v_1 \neq v_2)$ 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 $\Pi_Q$ 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 $\Pi_Q$ try to construct from $\text{CAN}(I)$ a solution $J$ for $I$ such that $\bar{a} \not\in Q(J)$. 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 $\exists \bar{u} (T_{p_1}(\bar{u}_1) \land \cdots \land T_{p_n}(\bar{u}_n) \land \neg T_{p_{n+1}}(\bar{u}_{n+1}))$, then $\Pi_Q$ uses rule $(4.16)$ 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 $\exists \bar{u} (T_{p_1}(\bar{u}_1) \land \cdots \land T_{p_n}(\bar{u}_n) \land v_1 \neq v_2)$, then $\Pi_Q$ uses rule $(4.17)$ to create a new solution where the values assigned to $v_1$ and $v_2$ are equal (predicate $\text{EQUAL}$ 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 (Fagin, Kolaitis, Miller, & Popa, 2005) 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 (4.18) 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 \( \mathcal{M} = (S, T, \Sigma_{st}) \) and for every instance \( I \) of \( S \), 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. \( \square \)

At this point, a natural question about DATALOG\(^C(\neq)\) 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 (Fagin, Kolaitis, Miller, & Popa, 2005) 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 \( Q^* \) such that for every source instance \( I \), it holds that \( \text{certain}_\mathcal{M}(Q, I) = Q^*(\text{CAN}(I)) \). Thus, given that a non-recursive DATALOG\(^C(\neq)\) program is equivalent to a first-order query, we conclude from Proposition 3.1 that recursion is necessary for capturing the class of unions of conjunctive queries with at most one negated atom per disjunct.

**Proposition 4.1** ((Fagin, Kolaitis, Miller, & Popa, 2005)). *There exist a data exchange setting \( \mathcal{M} \) and a Boolean conjunctive query \( Q \) with a single inequality such that for every non-recursive DATALOG\(^C(\neq)\) program \( \Pi \), it is the case that \( \text{certain}_\mathcal{M}(Q, I) \neq \text{certain}_\mathcal{M}(\Pi, I) \) holds, 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 (Fagin, Kolaitis, Miller, & Popa, 2005) mentioned above, as there are DATALOG\(^C(\neq)\) programs without inequalities that are not expressible in first-order logic.
**Proposition 4.2.** There exist a data exchange setting \( M \) and a Boolean conjunctive query \( Q \) with a single inequality such that for every DATALOG\(^{C(\neq)}\) program \( \Pi \) without inequalities, \( \text{certain}_M(Q, I) \neq \text{certain}_M(\Pi, I) \) for some source instance \( I \).

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

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

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

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

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 \( \text{certain}_M(Q, I_1) = \text{true} \), we have that \( \Pi_0(J_1) = \text{true} \). Let \( h \) be a function from \( \text{dom}(J_1) \) to \( \text{dom}(J_2) \) defined as \( h(a) = h(b) = c \). Since \( \Pi_0 \) is a DATALOG\(^{C(\neq)}\) 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 \( \Pi_0(J_2) = \text{true} \). Hence, given that \( J_2 \) is a universal solution for \( I_2 \), we conclude from Proposition 3.1 that \( \text{certain}_M(\Pi_0, I_2) = \text{true} \).

But we assume that \( \text{certain}_M(Q, I_2) = \text{certain}_M(\Pi_0, I_2) \) and, therefore, we obtain a contradiction since \( \text{certain}_M(Q, I_2) = \text{false} \).

Notice that as a corollary of Proposition 4.2 and Theorem 4.1, we obtain that DATALOG\(^{C(\neq)}\) programs are strictly more expressive than DATALOG\(^{C(\neq)}\) programs without inequalities.
We conclude this chapter by studying the complexity of the problem of computing certain answers to $\text{DATALOG}^{C(\neq)}$ programs. It was shown in Proposition 3.1 that the certain answers of a $\text{DATALOG}^{C(\neq)}$ program $\Pi$ can be computed by directly posing $\Pi$ over $\text{CAN}(I)$. This implies that for each data exchange setting $\mathcal{M}$, the problem $\text{CERTAIN-ANSWERS}(\mathcal{M}, \Pi)$ can be solved in polynomial time if $\Pi$ is a $\text{DATALOG}^{C(\neq)}$ program (since $\text{CAN}(I)$ can be computed in polynomial time and $\Pi$ has polynomial time data complexity).

**Proposition 4.3.** The problem $\text{CERTAIN-ANSWERS}(\mathcal{M}, \Pi)$ can be solved in polynomial time, for every data exchange setting $\mathcal{M}$ and $\text{DATALOG}^{C(\neq)}$ program $\Pi$.

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 (Fagin, Kolaitis, Miller, & Popa, 2005), which is stated for the class of unions of conjunctive queries with at most one inequality per disjunct. The proof of the result in (Fagin, Kolaitis, Miller, & Popa, 2005) uses different techniques, based on the chase procedure. In chapter 5, we show that $\text{DATALOG}^{C(\neq)}$ 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 $\text{CERTAIN-ANSWERS}(\mathcal{M}, \Pi)$ is PTIME-complete for some data exchange setting $\mathcal{M}$ and $\text{DATALOG}^{C(\neq)}$ program $\Pi$. It is easy to see that this is the case given that the data complexity of the evaluation problem for $\text{DATALOG}$ 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 $\text{CERTAIN-ANSWERS}(\mathcal{M}, Q)$ is PTIME-complete.

**Proposition 4.4.** There exist a LAV data exchange setting $\mathcal{M}$ and a Boolean conjunctive query $Q$ with one inequality such that $\text{CERTAIN-ANSWERS}(\mathcal{M}, Q)$ is PTIME-complete.
Proof: Let $\mathcal{M} = (S, T, \Sigma_{st})$ be a data exchange setting defined as follows. Source schema $S$ consists of a unary relation $V$, a binary relation $S$, and a 4-ary relation $P$. Target schema $T$ consists of a binary relation $T$ and a 4-ary relation $R$. Set $\Sigma_{st}$ consists of the following source-to-target dependencies:

\begin{align*}
V(x) & \rightarrow \exists y T(x, y) \quad (4.19) \\
S(x, y) & \rightarrow T(x, y) \quad (4.20) \\
P(x, y, w, z) & \rightarrow R(x, y, w, z) \quad (4.21)
\end{align*}

Furthermore, Boolean query $Q$ over $T$ is defined as:

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

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

Membership of $\text{CERTAIN-ANSWERS}(\mathcal{M}, Q)$ in PTIME follows from (Fagin, Kolaitis, Miller, & Popa, 2005). PTIME-hardness is established from a LOGSPACE reduction from Horn-3CNF to the complement of $\text{CERTAIN-ANSWERS}(\mathcal{M}, Q)$, 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., (Greenlaw, Hoover, & Ruzzo, 1995)). More precisely, for every Horn-3CNF formula $\phi$, we construct in logarithmic space an instance $I_\phi$ of $S$ such that $\phi$ is satisfiable if and only if $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \text{false}$.

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

- The interpretation of unary relation $V$ in $I_\phi$ is the set of propositional variables mentioned in $\phi$. 
• The interpretation of binary relation $S$ in $I_\phi$ is the set of tuples $\{(b, b), (h, f)\}$, where $b$, $h$ and $f$ are fresh constants (not mentioned as propositional variables in $\phi$).

• For every clause $C_i$ in $I_\phi (i \in \{1, \ldots, k\})$, the interpretation of 4-ary relation $P$ in $I_\phi$ contains the following tuple:
  - $(p, q, r, b)$ if $C_i = p \lor \neg q \lor \neg r$,
  - $(p, b, b, b)$ if $C_i = p$, and
  - $(h, p, q, r)$ if $C_i = \neg p \lor \neg q \lor \neg r$.

Clearly, $I_\phi$ can be constructed in logarithmic space from $\phi$.

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

$(\Rightarrow)$ Assume first that $\text{certain}_M(Q, I_\phi) = \text{false}$.

In the setting $M$, the canonical universal solution $\text{CAN}(I_\phi)$ for $I_\phi$ is as follows. Assume that $\bot_q$ is the null generated by applying rule (4.20) to each atom $V(q)$ in $I_\phi$. 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 $(b, b)$, $(h, f)$ and $(q, \bot_q)$ for every propositional variable $q$ mentioned in $\phi$.

Given that $\text{certain}_M(Q, I_\phi) = \text{false}$, there exists a solution $J$ for $I$ such that $Q(J) = \text{false}$. Let $h : \text{CAN}(I) \rightarrow J$ be an homomorphism from $\text{CAN}(I)$ into $J$, and let $\sigma$ be the following truth assignment for the propositional variables mentioned in $\phi$: $\sigma(q) = 1$ iff $h(\bot_q) = q$. Next we show that $\sigma$ satisfies $\phi$. 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 $R(p, b, b, b)$ belongs to $J$, and also $T(b, b)$ belongs to $J$, it must be the case that $h(\bot_p) = p$ since $Q(J) = \text{false}$ and $(p, \bot_p)$ belongs to the interpretation of $T$ in $\text{CAN}(I)$. We conclude that $\sigma(p) = 1$ and, hence, $\sigma(C_i) = 1$.

• Assume that $C_i = p \lor \neg q \lor \neg r$ and $\sigma(q) = \sigma(r) = 1$. Then by definition of $h$, we have that $h(\bot_q) = q$ and $h(\bot_r) = r$ and, therefore, $(q, q)$ and $(r, r)$ belong to the
Let \( J \) prove that this leads to a contradiction, we consider three cases.

- 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 \( h(\bot_p) = p \), \( h(\bot_q) = q \) and \( h(\bot_r) = r \) and, therefore, \( (p, p), (q, q) \) and \( (r, r) \) belong to the interpretation of \( T \) in \( J \). Thus, given that \( R(h, p, q, r) \), \( T(h, f) \) belong to \( J \) and \( h \neq f \) holds, we conclude that \( Q(J) = \text{true} \), which contradicts our initial assumption. We conclude that \( \sigma(p) = 0 \) or \( \sigma(q) = 0 \) or \( \sigma(r) = 0 \), which implies that \( \sigma(C_i) = 1 \).

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

\[
    f(v) = \begin{cases} 
        q & v = \bot_q \text{ and } \sigma(q) = 1 \\
        v & \text{otherwise} 
    \end{cases}
\]

Let \( J^* \) be a solution for \( I_\phi \) under \( M \) obtained from \( \text{CAN}(I_\phi) \) by replacing each occurrence of an element \( v \) with \( f(v) \). Next we show that \( Q(J^*) = \text{false} \) and, thus, we conclude that \( \text{certain}_M(Q, I_\phi) = \text{false} \).

Assume, for the sake of contradiction, that \( Q(J^*) = \text{true} \). Then, there exists a function \( h : \{x, y, w, z, x' \} \rightarrow \text{dom}(J^*) \) 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 \( J^* \), 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 \( h(y) = h(w) = h(z) = b \). Then by definition of \( M \) and \( I_\phi \), we have that \( p \) is a clause in \( \phi \). But given that \( h(x) = p \), \( h(x) \neq h(x') \) and \( T(h(x), h(x')) \) is a tuple in \( J^* \), it is the case that \( h(x') = \bot_p \). Thus, given that \( \bot_p \) is an element of \( J^* \), 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 \( h(z) = b \). Then by definition of \( M \) and \( I_\phi \), we have that \( p \land \neg q \land \neg r \) is a clause in \( \phi \). But given that \( h(x) = p, h(x) \neq h(x') \) and \( T(h(x), h(x')) \) is a tuple in \( J^* \), it is the case that \( h(x') = \bot_p \). Thus, given that \( \bot_p \) is an element of \( J^* \), 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 \( J^* \), it holds that \( T(q, q) \) and \( T(r, r) \) are tuples in \( J^* \). 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 \( h(x) = h, h(y) = p, h(w) = q \) and \( h(z) = r \), where \( p, q \) and \( r \) are propositional variables. Then by definition of \( M \) and \( I_\phi \), we have that \( \neg p \lor \neg q \lor \neg r \) is a clause in \( \phi \). But given that \( T(h(y), h(y)), T(h(w), h(w)) \) and \( T(h(z), h(z)) \) are all tuples in \( J^* \), it holds that \( T(p, p), T(q, q) \) and \( T(r, r) \) are all tuples in \( J^* \). 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 \( \sigma(p) = 0 \) and \( \sigma(q) = \sigma(r) = 1 \), which contradicts our initial assumption.

This concludes the proof of the proposition. \( \square \)

It is worth mentioning that it follows from Proposition 3.1 in (Kolaitis et al., 2006) that there exist a data exchange setting \( M \) containing some target dependencies and a conjunctive query \( Q \) with one inequality such that the problem \textsc{Certain-Answers}(\( M, Q \)) is PTIME-complete. Proposition 4.4 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 LA**V** setting $\mathcal{M}$ and a Boolean conjunctive query $Q$ with two inequalities such that $\text{CERTAIN-ANSWERS}(\mathcal{M}, Q)$ is $\text{coNP}$-complete (Madry, 2005). Therefore, unless $\text{P} = \text{NP}$, 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 chapter, 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$^{C(\neq)}$ program $\Pi_Q$ such that $\text{certain}_\mathcal{M}(Q, I) = \text{certain}_\mathcal{M}(\Pi_Q, I)$ for every source instance $I$. It should be noticed that DATALOG$^{C(\neq)}$ programs are used in this case 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 $\mathcal{M} = (S, T, \Sigma_{st})$ be a data exchange setting. Then for every $n$-ary relation symbol $T$ in $T$, we say that the $i$-th attribute of $T$ ($1 \leq i \leq n$) can be nullified under $\mathcal{M}$, if there is an std $\alpha$ in $\Sigma_{st}$ such that the $i$-th attribute of $T$ is existentially quantified in the right hand side of $\alpha$. 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 $T$ 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 $\text{CERTAIN-ANSWERS}(\mathcal{M}, Q)$ is tractable. Indeed, the previous condition implies that for every source instance $I$, if $Q$ holds
in \( \text{CAN}(I) \), then all the witnesses for \( Q \) in \( \text{CAN}(I) \) make comparisons of the form \( c \neq c' \), where \( c \) and \( c' \) are constants. Thus, we have that \( \text{certain}_M(Q, I) \) can be computed by simply evaluating \( Q \) over \( \text{CAN}(I) \). 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 \( T \). Assume that the quantifier free part of \( Q \) is of the form \( \phi(x_1, \ldots, x_m) \land u_1 \neq v_1 \land u_2 \neq v_2 \), where \( \phi \) is a conjunction of relational atoms over \( T \) 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 (Abiteboul, Hull, & Vianu, 1995)). 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 \( M \), if \( u_1 \) or \( v_1 \) cannot be nullified under \( Q \) and \( M \), and \( u_2 \) or \( v_2 \) cannot be nullified under \( Q \) and \( M \). Intuitively, this means that to satisfy \( Q \) in the canonical universal solution of a source instance, one can only make comparisons of the form \( c \neq \bot \) and \( c \neq c' \), where \( c, c' \) are constants and \( \bot \) is a null value. Moreover, we say that \( Q \) has **constant joins** under \( M \), if for every variable \( x \) that appears at least twice in \( \phi \), \( x \) cannot be nullified under \( Q \) and \( M \). 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 \( M \) be a data exchange setting containing two stds: \( P(x, y) \rightarrow T(x, y) \) and \( P(x, y) \rightarrow \exists z U(x, z) \). The first and second attribute of \( T \), as well as the first attribute of \( U \), cannot be nullified under \( M \). On the other hand, the second attribute of \( U \) can be nullified under \( M \).

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

Although the notions of constant joins and almost constant inequalities were defined for UCQ$\neq$ 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 chapter.

**Theorem 5.1.** Let $\mathcal{M} = (S, T, \Sigma_{st})$ be a data exchange setting and $Q$ a UCQ$\neq$ query over $T$ 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 DATALOG$^C(\neq)$ program $\Pi_Q$ over $T$ such that for every instance $I$ of $S$, $\text{certain}_\mathcal{M}(Q, I) = \text{certain}_\mathcal{M}(\Pi_Q, I)$. Moreover, $\Pi_Q$ can be effectively constructed from $Q$ and $\mathcal{M}$ in polynomial time.

**Proof:** Assume that $T = \{T_1, \ldots, T_k\}$, where each $T_i$ has arity $n_i > 0$, and that $Q(\bar{x}) = Q_1(\bar{x}) \lor \cdots \lor Q_\ell(\bar{x})$, 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 or negated relational atom, 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 DATALOG$^C(\neq)$ program $\Pi_Q$ is

$$\{U_1, \ldots, U_k, \text{DOM, EQUAL, (EQUAL}^{\text{left}, j})_{j \in W}, (\text{EQUAL}^{\text{right}, j})_{j \in W}, \text{ANSWER, (ANSWER}^{\text{left}, j})_{j \in W}, (\text{ANSWER}^{\text{right}, j})_{j \in W}\},$$

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 $\text{EQUAL}^{\text{left}, j}$ or $\text{EQUAL}^{\text{right}, j}$, for $j \in W$, has arity $2 + p_j + m$;
• ANSWER has arity $m$; and
• each predicate of the form $\text{ANSWER}^{\text{left}, j}$ or $\text{ANSWER}^{\text{right}, j}$, for $j \in W$, has arity $p_j + m$.

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

• For every predicate $T_i \in \mathcal{T}$, $\Pi_Q$ includes the following $n_i$ rules:

\[
\begin{align*}
\text{DOM}(x) & \leftarrow T_i(x, y_2, y_3, \ldots, y_{n_i-1}, y_{n_i}) \\
\text{DOM}(x) & \leftarrow T_i(y_1, x, y_3, \ldots, y_{n_i-1}, y_{n_i}) \\
\cdots \\
\text{DOM}(x) & \leftarrow T_i(y_1, y_2, y_3, \ldots, y_{n_i-1}, x)
\end{align*}
\]

Intuitively, predicate $\text{DOM}$ collects the elements that belong to the domain of the extensional instance.

• $\Pi_Q$ includes the following rules for predicate EQUAL:

\[
\begin{align*}
\text{EQUAL}(x, x, \bar{z}) & \leftarrow \text{DOM}(x), \text{DOM}(\bar{z}) \\
\text{EQUAL}(x, y, \bar{z}) & \leftarrow \text{EQUAL}(y, x, \bar{z}) \\
\text{EQUAL}(x, y, \bar{z}) & \leftarrow \text{EQUAL}(x, w, \bar{z}), \text{EQUAL}(w, y, \bar{z})
\end{align*}
\]
\(\Pi_Q\) includes the following rules for predicate \(\text{EQUAL}^{\text{left},j}\), for each \(j \in W\), where \(\bar{u}\) is a tuple of \(p_j\) fresh variables:

\[
\begin{align*}
\text{EQUAL}^{\text{left},j}(x, x, \bar{u}, \bar{z}) & \leftarrow \text{DOM}(x), \text{DOM}(\bar{u}), \text{DOM}(\bar{z}) \\
\text{EQUAL}^{\text{left},j}(x, y, \bar{u}, \bar{z}) & \leftarrow \text{EQUAL}^{\text{left},j}(y, x, \bar{u}, \bar{z}) \\
\text{EQUAL}^{\text{left},j}(x, y, \bar{u}, \bar{z}) & \leftarrow \text{EQUAL}^{\text{left},j}(x, w, \bar{u}, \bar{z}), \text{EQUAL}^{\text{left},j}(w, y, \bar{u}, \bar{z})
\end{align*}
\]

\(\Pi_Q\) includes the following rules for predicate \(\text{EQUAL}^{\text{right},j}\), for each \(j \in W\), where \(\bar{u}\) is a tuple of \(p_j\) fresh variables:

\[
\begin{align*}
\text{EQUAL}^{\text{right},j}(x, x, \bar{u}, \bar{z}) & \leftarrow \text{DOM}(x), \text{DOM}(\bar{u}), \text{DOM}(\bar{z}) \\
\text{EQUAL}^{\text{right},j}(x, y, \bar{u}, \bar{z}) & \leftarrow \text{EQUAL}^{\text{right},j}(y, x, \bar{u}, \bar{z}) \\
\text{EQUAL}^{\text{right},j}(x, y, \bar{u}, \bar{z}) & \leftarrow \text{EQUAL}^{\text{right},j}(x, w, \bar{u}, \bar{z}), \text{EQUAL}^{\text{right},j}(w, y, \bar{u}, \bar{z})
\end{align*}
\]

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

\[
U_i(\bar{y}, \bar{z}) \leftarrow T_i(\bar{y}), \text{DOM}(\bar{z})
\]

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 \(\exists \bar{u} \left( T_{s_1}(\bar{u}_1) \land \cdots \land T_{s_n}(\bar{u}_n) \right)\), 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 \(\Pi_Q\) includes the following rule:

\[
\text{ANSWER}(\bar{x}) \leftarrow U_{s_1}(\bar{u}_1, \bar{x}), \ldots, U_{s_n}(\bar{u}_n, \bar{x}), C(x_1), \ldots, C(x_m)
\]

Notice that this rule is well defined since the set \(\bar{x}\) is the set of free variables of \(\exists \bar{u} \left( T_{s_1}(\bar{u}_1) \land \cdots \land T_{s_n}(\bar{u}_n) \right)\).

Second, assume that \(Q_i(\bar{x})\) contains an inequality. Then \(Q_i(\bar{x})\) is equal to the formula \(\exists \bar{u} \left( T_{s_1}(\bar{u}_1) \land \cdots \land T_{s_n}(\bar{u}_n) \land v_1 \neq v_2 \right)\), 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
In this case, program $\Pi_Q$ includes the following rules:

$$\text{EQUAL}(v_1, v_2, \bar{x}) \leftarrow U_{s_1}(\bar{u}_1, \bar{x}), \ldots, U_{s_n}(\bar{u}_n, \bar{x})$$

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

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

$$\text{EQUAL}^{\text{left}, j}(v_1, v_2, \bar{y}, \bar{x}) \leftarrow U_{s_1}(\bar{u}_1, \bar{x}), \ldots, U_{s_n}(\bar{u}_n, \bar{x}), \text{DOM}(\bar{y})$$

$$\text{EQUAL}^{\text{right}, j}(v_1, v_2, \bar{y}, \bar{x}) \leftarrow U_{s_1}(\bar{u}_1, \bar{x}), \ldots, U_{s_n}(\bar{u}_n, \bar{x}), \text{DOM}(\bar{y})$$

We note that the rules above are also well defined since $\exists \bar{u} \left( T_{s_1}(\bar{u}_1) \land \cdots \land T_{s_n}(\bar{u}_n) \land v_1 \neq v_2 \right)$ 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 $\exists \bar{u} \left( T_{s_1}(\bar{u}_1) \land \cdots \land T_{s_n}(\bar{u}_n) \land v_1 \neq v_2 \land v_3 \neq v_4 \right)$, 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 $\Pi_Q$ includes the following rules:

$$\text{EQUAL}^{\text{left}, i}(v_1, v_2, \bar{u}, \bar{x}) \leftarrow U_{s_1}(\bar{u}_1, \bar{x}), \ldots, U_{s_n}(\bar{u}_n, \bar{x})$$

$$\text{EQUAL}^{\text{right}, i}(v_3, v_4, \bar{u}, \bar{x}) \leftarrow U_{s_1}(\bar{u}_1, \bar{x}), \ldots, U_{s_n}(\bar{u}_n, \bar{x})$$
Further, in this case $\Pi_Q$ also contains the following rules for each $j \in W$, assuming $\vec{y}$ is a tuple of $p_j$ fresh variables:

\[
\begin{align*}
\text{EQUAL}_\text{left}^j(v_3, v_4, \vec{y}, \vec{x}) & \leftarrow U_{p_1}(\vec{u}_1, \vec{x}), \ldots, U_{p_n}(\vec{u}_n, \vec{x}), \mathbf{C}(v_1), \\
\text{EQUAL}_\text{left}^j(v_2, w, \vec{y}, \vec{x}), \mathbf{C}(w), v_1 \neq w \\
\text{EQUAL}_\text{left}^j(v_3, v_4, \vec{y}, \vec{x}) & \leftarrow U_{p_1}(\vec{u}_1, \vec{x}), \ldots, U_{p_n}(\vec{u}_n, \vec{x}), \mathbf{C}(v_2), \\
\text{EQUAL}_\text{left}^j(v_1, w, \vec{y}, \vec{x}), \mathbf{C}(w), v_2 \neq w \\
\text{EQUAL}_\text{left}^j(v_1, v_2, \vec{y}, \vec{x}) & \leftarrow U_{p_1}(\vec{u}_1, \vec{x}), \ldots, U_{p_n}(\vec{u}_n, \vec{x}), \mathbf{C}(v_3), \\
\text{EQUAL}_\text{left}^j(v_4, w, \vec{y}, \vec{x}), \mathbf{C}(w), v_3 \neq w \\
\text{EQUAL}_\text{right}^j(v_3, v_4, \vec{y}, \vec{x}) & \leftarrow U_{p_1}(\vec{u}_1, \vec{x}), \ldots, U_{p_n}(\vec{u}_n, \vec{x}), \mathbf{C}(v_1), \\
\text{EQUAL}_\text{right}^j(v_2, w, \vec{y}, \vec{x}), \mathbf{C}(w), v_1 \neq w \\
\text{EQUAL}_\text{right}^j(v_3, v_4, \vec{y}, \vec{x}) & \leftarrow U_{p_1}(\vec{u}_1, \vec{x}), \ldots, U_{p_n}(\vec{u}_n, \vec{x}), \mathbf{C}(v_2), \\
\text{EQUAL}_\text{right}^j(v_1, w, \vec{y}, \vec{x}), \mathbf{C}(w), v_2 \neq w \\
\text{EQUAL}_\text{right}^j(v_1, v_2, \vec{y}, \vec{x}) & \leftarrow U_{p_1}(\vec{u}_1, \vec{x}), \ldots, U_{p_n}(\vec{u}_n, \vec{x}), \mathbf{C}(v_3), \\
\text{EQUAL}_\text{right}^j(v_4, w, \vec{y}, \vec{x}), \mathbf{C}(w), v_3 \neq w \\
\text{EQUAL}_\text{right}^j(v_1, v_2, \vec{y}, \vec{x}) & \leftarrow U_{p_1}(\vec{u}_1, \vec{x}), \ldots, U_{p_n}(\vec{u}_n, \vec{x}), \mathbf{C}(v_4), \\
\text{EQUAL}_\text{right}^j(v_3, w, \vec{y}, \vec{x}), \mathbf{C}(w), v_4 \neq w \\
\end{align*}
\]

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

\[
\begin{align*}
\text{ANSWER}_\text{left}^j(\vec{y}, \vec{x}) & \leftarrow \text{EQUAL}_\text{left}^j(u, v, \vec{y}, \vec{x}), \mathbf{C}(u), \mathbf{C}(v), u \neq v \\
\text{ANSWER}_\text{right}^j(\vec{y}, \vec{x}) & \leftarrow \text{EQUAL}_\text{right}^j(u, v, \vec{y}, \vec{x}), \mathbf{C}(u), \mathbf{C}(v), u \neq v \\
\text{ANSWER}(\vec{x}) & \leftarrow \text{ANSWER}_\text{left}^j(\vec{y}, \vec{x}), \text{ANSWER}_\text{right}^j(\vec{y}, \vec{x}), \mathbf{C}(x_1), \ldots, \mathbf{C}(x_m)
\end{align*}
\]
It can be proved that for every data exchange setting \( M = (S, T, \Sigma_{st}) \) and instance \( I \) of \( S \),
\[
certain_M(Q, I) = certain_M(\Pi_Q, I).
\]
This concludes the proof of the theorem. \( \Box \)

It immediately follows from Proposition 4.3 that if a data exchange setting \( M \) and a UCQ\( ^\neq \) query \( Q \) satisfy the conditions mentioned in Theorem 5.1, then the problem denoted by \( \text{CERTAIN-ANSWERS}(M, Q) \) 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 5.1.** Let \( Q \) be a UCQ\( ^\neq \) query with at most one inequality per disjunct. If every disjunct of \( Q \) has constant joins under a setting \( M \), then \( \text{CERTAIN-ANSWERS}(M, Q) \) is in NLOGSPACE, and if in addition every disjunct of \( Q \) has almost constant inequalities under \( M \), then \( \text{CERTAIN-ANSWERS}(M, Q) \) 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 the \( \text{CAN}(I) \) for an arbitrary source instance \( I \) under \( M \), and the \( i \)-th attribute of \( T \) (\( 1 \leq i \leq n \)) is not existentially quantified in \( M \), then \( p_i \) has to be a constant.

We now prove the proposition, and start with part (1). Let \( M = (S, T, \Sigma_{st}) \) 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 \( certain_M(Q, I) = Q'(\text{CAN}(I)) \), for every source instance \( I \). It will immediately follow that \( \text{CERTAIN-ANSWERS}(M, Q) \) is in NLOGSPACE (the algorithm first computes \( \text{CAN}(I) \) in LOGSPACE, and then evaluates \( Q' \) over \( \text{CAN}(I) \) in NLOGSPACE).
The query $Q'$ will be defined in transitive closure logic (for a precise definition of this logic, see e.g. Chapter 10.6 in (Libkin, 2004)). In order to do so, we need to introduce some extra terminology and prove an intermediate result (Lemma 5.1).

Assume that $Q$ is $Q_1(\bar{x}) \lor \cdots \lor Q_\ell(\bar{x})$, 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 $\mu$ and $\nu$, i.e. neither $\mu$ nor $\nu$ belongs to $\text{CAN}(I)$;
- there exists an edge between elements $p$ and $p'$ in $G(Q, I, \bar{t})$, $p, p' \in \text{dom}(\text{CAN}(I))$, iff for some $i \in [1, \ell]$, (1) $Q_i(\bar{x})$ is of the form $\exists \bar{y}\phi(\bar{x}, \bar{y}) \land u \neq v$, where $\phi(\bar{x}, \bar{y})$ is a conjunction of relational atoms over $T$, and $u, v \in \{\bar{x}, \bar{y}\}$, and (2) there is an assignment $\sigma : \{\bar{x}, \bar{y}\} \rightarrow \text{dom}(\text{CAN}(I))$, such that $\sigma(\bar{x}) = \bar{t}$, $(\text{CAN}(I), \sigma) \models \phi(\bar{x}, \bar{y}) \land u \neq v$, $\sigma(u) = p$ and $\sigma(v) = p'$; and
- there exists an edge between $\mu$ and $\nu$ 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 $T$, 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 $\mu$ and $\nu$. Next claim shows that checking whether a tuple $\bar{t}$ of constants from $\text{CAN}(I)$ belongs to the certain answers of $Q$ for $I$ is equivalent to checking for the presence of a $c$-path in $G(Q, I, \bar{t})$.

**Lemma 5.1.** Let $Q$ be as defined above. For every source instance $I$ and tuple $\bar{t}$ of constants from $I$, it is the case that

$$\bar{t} \in \text{certain}_M(Q, I) \iff G(Q, I, \bar{t}) \text{ has a } c\text{-path}.$$  

**Proof:** Fix a source instance $I$ and a tuple $\bar{t}$ of constants from $I$. Assume first that there is no path in $G(Q, I, \bar{t})$ from some constant $c \in \text{dom}(\text{CAN}(I))$ to some different constant...
\( c' \in \text{dom}(\text{CAN}(I)) \), and there is no edge in \( G(Q, I, \bar{t}) \) between \( \mu \) and \( \nu \). Thus, each connected component of \( G(Q, I, \bar{t}) \) with elements in \( \text{dom}(\text{CAN}(I)) \) consists of at most one constant and zero or more null values. With each connected component \( M \) with elements in \( \text{dom}(\text{CAN}(I)) \), we identify an element \( \text{id}(M) \in M \) as follows: \( \text{id}(M) = c \), if \( M \) contains a constant \( c \), and \( \text{id}(M) = n \) for an arbitrary null value \( n \in M \), otherwise. Let \( J^* \) be the solution obtained from \( \text{CAN}(I) \) by replacing each occurrence of an element \( p \in \text{dom}(\text{CAN}(I)) \) in connected component \( M \) with the element \( \text{id}(M) \). It is not hard to see that every element in \( \bar{t} \) belongs to \( J^* \). We prove that \( J^* \models Q(\bar{t}) \), and, thus, that certain \( \mu(Q, I) = \text{false} \).

Assume otherwise. Then there exists \( i \in [1, \ell] \) such that \( J^* \models Q_i(\bar{t}) \). Assume first that \( Q_i \) is of the form \( \exists y_1, \ldots, y_n(T_1(\bar{x}_1, y_1) \land \cdots \land T_n(\bar{x}_n, y_n) \land u \neq v) \), where \( \{T_1, \ldots, T_n\} \subseteq T \), \( \bar{x} = \{\bar{x}_1, \ldots} \cup \cdots \cup \{\bar{x}_n\} \), and \( u, v \in \{\bar{x}, \bar{y}_1, \ldots, \bar{y}_n\} \). Thus, there exist tuples \( \bar{p}_1, \ldots, \bar{p}_n \) of elements in \( \text{CAN}(I) \) and an assignment \( \sigma : \{\bar{x}\} \cup \{\bar{y}_1\} \cup \cdots \cup \{\bar{y}_n\} \rightarrow \text{dom}(J^*) \) defined by \( \sigma(\bar{x}) = \bar{t} \) and \( \sigma(\bar{y}_j) = \bar{p}_j \), for every \( 1 \leq j \leq n \), such that \( (J^*, \sigma) \models T_1(\bar{x}_1, y_1) \land \cdots \land T_n(\bar{x}_n, y_n) \land u \neq v \). In particular, \( \sigma(u) \neq \sigma(v) \).

For every \( j \in [1, n] \), let us denote by \( \bar{t}_j \) the value of \( \sigma(\bar{x}_j) \). By definition of \( J^* \), every tuple \( T_j(\bar{t}_j, \bar{p}_j) \) \((1 \leq j \leq n)\) is obtained from a tuple \( T_j(\bar{r}_j, r_j) \) in \( J \) by replacing each element \( r \in \bar{r}_j \) in the connected component \( M \) of \( G(Q, I, \bar{t}) \) by \( \text{id}(M) \). Let us define an assignment \( \sigma' : \{\bar{x}\} \cup \{\bar{y}_1\} \cup \cdots \cup \{\bar{y}_n\} \rightarrow \text{dom}(\text{CAN}(I)) \) as follows: \( \sigma'(\bar{x}) = \bar{t} \) and \( \sigma'(\bar{y}_j) = \bar{r}_j \), for each \( 1 \leq j \leq n \). We show that \( \sigma' \) is well-defined. Assume that \( z \) is a variable that appears in at least two different positions in \( (\bar{y}_1, \ldots, \bar{y}_n) \). We show that \( \sigma' \) assigns the same value to each appearance of \( z \). Indeed, since \( z \) appears in two different positions in \( (\bar{y}_1, \ldots, \bar{y}_n) \) it must be the case that \( z \) is a join variable in \( T_1(\bar{x}_1, y_1) \land \cdots \land T_n(\bar{x}_n, y_n) \). By hypothesis, \( Q \) has constant joins in \( \mathcal{M} \), and, thus, \( z \) does not appear existentially quantified in \( \mathcal{M} \). Thus, \( \sigma(z) \) is a constant and \( \sigma(z) = \sigma'(z) \) (because \( \text{CAN}(I) \) only contain constants in the positions of attributes that do not appear existentially quantified in \( \mathcal{M} \)). It immediately follows that every appearance of \( z \) in \( (\bar{y}_1, \ldots, \bar{y}_n) \) is assigned the same value by \( \sigma' \).

We conclude that \( (\text{CAN}(I), \sigma') \models T_1(\bar{x}_1, y_1) \land \cdots \land T_n(\bar{x}_n, y_n) \). If \( \sigma'(u) = \sigma'(v) \), then \( \sigma(u) = \sigma(v) \), which is a contradiction. Assume then that \( \sigma'(u) \neq \sigma'(v) \). By definition of
$G(Q, I, \bar{t})$, $\sigma'(u)$ and $\sigma'(v)$ are in the same connected component of $G(Q, I, \bar{t})$. But then, by definition of $J^*$ it must be the case that $\sigma(u) = \sigma(v)$, which is our desired contradiction.

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

Assume, on the other hand, that there is a path in $G(Q, I, \bar{t})$ from a constant $c$ such that $c \in \text{dom} \text{(CAN}(I))$ to a different constant $c' \in \text{dom}(\text{CAN}(I))$ (the case when there is an edge between $u$ and $v$ in $G(Q, I, \bar{t})$ can be handled similarly). Let $J^*$ be an arbitrary solution for $I$, and let $h$ be a homomorphism from $\text{CAN}(I)$ to $J^*$. Then there must be two adjacent elements $p$ and $p'$ in the connected component of $c$ in $G(Q, I, \bar{t})$, such that $h(p) \neq h(p')$. Further, since $p$ and $p'$ are adjacent in $G(Q, I, \bar{t})$, there must be an $i \in [1, \ell]$ such that (1) $Q_i(\bar{x})$ is of the form $\exists \bar{y}(\phi(\bar{x}, \bar{y}) \land u \neq v)$, where $\phi(\bar{x}, \bar{y})$ is a conjunction of relational atoms over $T$, and $u, v \in \{\bar{x}, \bar{y}\}$, and (2) there is an assignment $\sigma : \{\bar{x}, \bar{y}\} \rightarrow \text{dom}(\text{CAN}(I))$, such that $\sigma(\bar{x}) = \bar{t}$, $(\text{CAN}(I), \sigma) \models \phi(\bar{x}, \bar{y}) \land u \neq v$, $\sigma(u) = p$ and $\sigma(v) = p'$. Then $J^* \models \phi(h(\bar{t}), h(\sigma(\bar{y}))) \land h(\sigma(u)) \neq h(\sigma(v))$, because conjunctive queries are preserved under homomorphisms and $h(p) = h(\sigma(u)) \neq h(\sigma(v)) = h(p')$. Thus, $(J^*, \bar{t}) \models \exists \bar{y}(\phi(\bar{x}, \bar{y}) \land u \neq v)$, because $\bar{t} = h(\bar{t})$. We conclude that $J^* \models Q(\bar{t})$, and, therefore, since $J^*$ was arbitrarily chosen, that $\bar{t} \in \text{certain}_{\mathcal{M}}(Q, I)$. This finishes the proof of the lemma. 

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 $\exists \bar{y}_i(\phi_i(\bar{x}, \bar{y}_i) \land u_i \neq v_i)$, where $\phi_i(\bar{x}, \bar{y}_i)$ is a conjunction of relational atoms over $T$, 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 $T$. Then:
(i) 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 $Q(\bar{x})$:

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

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 $\text{CAN}(I)$;

(ii) 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 $\mu$ and $\nu$ in $G(Q, I, \bar{x})$; and

(iii) finally, the query $Q'(\bar{x})$ is defined as:

$$(\alpha(\bar{x}) \lor \exists w_1 \exists w_2 (C(w_1) \land C(w_2) \land w_1 \neq w_2$$

$$\land (w_1, w_2) \in \text{TrCl}.A(u, v, \bar{x})) \land C(x_1) \land \ldots C(x_m)$$

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 5.1 that for every source instance $I$, $\text{certain}_M(Q, I) = Q'(\text{CAN}(I))$. Further, it is well-known that the data complexity of any formula in transitive closure logic is in NLOGSPACE (see e.g. Chapter 10.6 in (Libkin, 2004)). 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 5.1 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 3. Thus, in this case $Q'(\bar{x})$ can be expressed as the FO formula that checks whether there is an edge between $\mu$ and $\nu$ in $G(Q, I, \bar{t})$, or a $c$-path of length at most 3 in $G(Q, I, \bar{t})$. 
Since the data complexity of any FO formula is in LOGSPACE (see e.g. Chapter 6 in (Libkin, 2004)), 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 are the conditions used in Theorem 5.1. 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 UCQ≠ queries with two inequalities.

**Theorem 5.2.**

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 \text{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 \text{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 \text{CERTAIN-ANSWERS}(\mathcal{M}, Q) \) is coNP-complete.

**Proof:** We will only show the proof for the first part of the theorem. For the details of the second and third part, see the appendix A. The proof for (1) is as follows:

We prove that there exists 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 \text{CERTAIN-ANSWERS}(\mathcal{M}, Q) \) is coNP-complete.
The LAV setting $\mathcal{M} = (S, T, \Sigma_{st})$ is as follows. The source schema $S$ consists of two relations: A binary relation $P$ and a ternary relation $R$. The target schema $T$ consists of three relations: Two binary relations $T$ and $S$, and a ternary relation $U$. Further, $\Sigma_{st}$ is the following set of source-to-target dependencies:

$$
P(x, y) \rightarrow \exists z (T(x, z) \land T(y, z) \land S(x, y))$$
$$R(x, y, z) \rightarrow U(x, y, z)$$

Furthermore, Boolean query $Q$ is defined as:

$$
\exists x \exists y \exists z (U(x, y, z) \land T(x, x) \land T(y, y) \land T(z, z)) \lor
\exists x \exists y \exists w \exists z (T(x, y) \land T(w, z) \land S(x, w) \land x \neq y \land w \neq z).
$$

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 $\text{CERTAIN-ANSWERS}(\mathcal{M}, Q)$ is $\text{coNP}$-complete.

Membership of $\text{CERTAIN-ANSWERS}(\mathcal{M}, Q)$ in $\text{coNP}$ follows from (Fagin, Kolaitis, Miller, & Popa, 2005). The $\text{coNP}$-hardness is established from a reduction from 3SAT to the complement of $\text{CERTAIN-ANSWERS}(\mathcal{M}, Q)$. More precisely, for every 3CNF propositional formula $\phi$, we construct in polynomial time an instance $I_\phi$ of $S$ such that $\phi$ is satisfiable iff $\text{certain}_\mathcal{M}(Q, I_\phi) = \text{false}$.

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

- The interpretation of $P$ in $I_\phi$ contains the pair $(q, \neg q)$, for each propositional variable $q$ mentioned in $\phi$; and
- the interpretation of $R$ in $I_\phi$ contains all tuples $(\alpha, \beta, \gamma)$ such that for some $1 \leq j \leq m$, $C_j = (\alpha \lor \beta \lor \gamma)$.}

Clearly, $I_\phi$ can be constructed in polynomial time from $\phi$. 

The canonical universal solution $J$ for $I_\phi$ is as follows, where we denote by $\bot_q$ (or $\bot_{\neg q}$) the null generated by applying the std $P(x, y) \rightarrow \exists z(T(x, z) \land T(y, z) \land S(x, y))$ 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 $\phi$;
- the interpretation of the relation $S$ in $J$ is just a copy of the interpretation of the relation $P$ in $I_\phi$; and
- the interpretation of the relation $U$ in $J$ is just a copy of the interpretation of the relation $R$ in $I_\phi$.

We prove now that $\phi$ is satisfiable iff $\text{certain}_M(Q, I_\phi) = \text{false}$.

$(\Rightarrow)$ Assume that $\phi$ is satisfiable, and let $\kappa$ be a truth assignment for the propositional variables of $\phi$ such that $\kappa(\phi) = 1$. From $\kappa$, 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 $I_\phi$ obtained from $J$ by replacing each occurrence of an element $v$ in $J$ by $f(v)$. We show next that $Q(J^*) = \text{false}$, and, thus, that $\text{certain}_M(Q, I_\phi) = \text{false}$.

Assume, for the sake of contradiction, that $Q(J^*) = \text{true}$. Then $Q_1(J^*) = \text{true}$ or $Q_2(J^*) = \text{true}$. Assume first that the latter holds. Then there is a function $h : \{x, y, z, w\} \rightarrow \text{dom}(J^*)$ 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 $\phi$, $h(x) = q$ 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 $Q_1(J^*) = \text{true}$. Then there is a function $h : \{x, y, z\} \rightarrow \text{dom}(J^*)$ 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 $I_\phi$, there exists a clause $(\alpha \lor \beta \lor \gamma)$ in $\phi$ 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 $certain_\mathcal{M}(Q, I_\phi) = \text{false}$. Then there exists a solution $J'$ such that $Q(J') = \text{false}$. Let $h : J \rightarrow J'$ be an homomorphism from $J$ into $J'$, and let $\kappa$ be the following truth assignment for the propositional variables mentioned in $\phi$:

$\kappa(q) = 1$ iff $h(\bot_q) = \neg q$. We show next that $\kappa(C_j) = 1$, for each $1 \leq j \leq m$, and, thus, that $\phi$ is satisfiable.

Consider an arbitrary $j \in [1, m]$, and assume that $C_j = (\alpha \lor \beta \lor \gamma)$. 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 $Q_2(J^*) = \text{false}$ 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 queries has been considered in the literature, none of the results of Theorem 5.2 directly follows from any of the known results for this problem. In particular, Fagin et al. showed in (Fagin, Kolaitis, Miller, & Popa, 2005) a similar result to (1), namely that the problem of computing certain answers is $\text{CO}_\text{NP}$-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 (Madry, 2005) a similar result to (2) and (3), namely that the problem of computing certain answers is \( \text{coNP} \)-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 \( \text{coNP} \)-complete.

We conclude this chapter with a remark about the possibility of using the conditions defined in this chapter to obtain tractability for \( \text{UCQ}^\neq \). As we mentioned above, the notions of constant joins and almost constant inequalities can be extended to \( \text{UCQ}^\neq \) 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 \( \text{UCQ}^\neq \) queries with three inequalities.

**Proposition 5.3.** There exist a \( \text{LAV} \) 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 \( \text{CERTAIN-ANSWERS}(\mathcal{M}, Q) \) is \( \text{coNP} \)-complete.

*Proof:* See appendix A
6. COMBINED COMPLEXITY OF DATA EXCHANGE

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 chapter 2. Let $\mathcal{DE}$ be a class of data exchange settings and $\mathcal{C}$ a class of queries. In this chapter, we study the following problem:

<table>
<thead>
<tr>
<th>PROBLEM:</th>
<th>CERTAIN-ANSWERS($\mathcal{DE}, \mathcal{C}$).</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUT:</td>
<td>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 $I$.</td>
</tr>
<tr>
<td>QUESTION:</td>
<td>Is $\bar{t} \in \text{certain}_{\mathcal{M}}(Q, I)$?</td>
</tr>
</tbody>
</table>

It is worth mentioning that a related study appeared in (Kolaitis et al., 2006). Even though the focus of that paper was the combined complexity of the existence of solutions problem, some of the results in (Kolaitis et al., 2006) 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 chapter we prove stronger lower bounds that consider single conjunctive queries with inequalities, and which cannot be directly proved by using the results of (Kolaitis et al., 2006).

We start by stating the complexity for the case of DATALOG$^C(\neq)$ queries. The study continues by considering some restrictions of DATALOG$^C(\neq)$ 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$^C(\neq)$ (unless PTIME = NP). The results of this chapter are summarized in Table 6.1, where we let $k$-CQ$^\neq$ be the class of CQ$^\neq$ queries with at most $k$ inequalities.

6.1. Combined Complexity of DATALOG$^C(\neq)$ programs

We showed in Proposition 3.1 that the certain answers of a DATALOG$^C(\neq)$ 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\(^C(\neq)\) 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\((\text{GLAV}, \text{DATALOG}\,(\neq))\) is \textsc{Exptime}-complete.

**Proof:** The \textsc{Exptime}-hardness follows directly from Theorems 4.1 and 6.2: The problem of computing certain answers is already \textsc{Exptime}-hard for the class of conjunctive queries with single inequalities (see Theorem 6.2), and it follows from Theorem 4.1 that for each query \(Q\) in this class, one can construct in polynomial time a DATALOG\(^C(\neq)\) program \(\Pi_Q\) such that 
\[\text{certain}_\mathcal{M}(Q,I) = \text{certain}_\mathcal{M}(\Pi_Q, I),\]
for every source instance \(I\).

For membership in \textsc{Exptime}, in chapter 3 we showed that in order to compute the certain answers of a DATALOG\(^C(\neq)\) program \(\Pi\) for a source instance \(I\), it suffices to evaluate \(\Pi\) over \text{CAN}(I). It is well known that every DATALOG program \(\Pi\) with negated atoms can be evaluated over an instance \(D\) in time \(|D|^{|\Pi|}\) (see e.g. (Abiteboul et al., 1995), (Vardi, 1982)). Hence, since for every source instance \(I\), \text{CAN}(I)\ is of size at most \(|I|^{|\Sigma_{st}|}\) (Fagin, Kolaitis, Miller, & Popa, 2005), we obtain an exponential bound for computing the certain answers of a DATALOG\(^C(\neq)\) program.

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. The less obvious but more interesting case is to restrict the class of data exchange settings to be \text{LAV} settings. However, for the case of DATALOG\(^C(\neq)\) 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 6.1.** Certain-Answers\((\text{LAV}, \text{DATALOG}\,(\neq))\) is \textsc{Exptime}-complete.
### Table 6.1. Combined complexity of computing certain answers.

<table>
<thead>
<tr>
<th>Query</th>
<th>GLAV setting</th>
<th>LAV setting</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DATALOG^{C(≠)}</code></td>
<td>EXPTIME-complete</td>
<td>EXPTIME-complete</td>
</tr>
<tr>
<td>1-CQ^{≠}</td>
<td>EXPTIME-complete</td>
<td>NP-complete</td>
</tr>
<tr>
<td><code>k-CQ^{≠}, k ≥ 2</code></td>
<td><code>coNEXPTIME</code></td>
<td><code>Π_{2}^{p}</code>-complete</td>
</tr>
</tbody>
</table>

**Proof:** The membership in EXPTIME follows from the proof of Theorem 6.1. The exponential bound is obtained with the evaluation of a `DATALOG` program `Π` with negated atoms over `CAN(I)`. For the EXPTIME-hardness, we will show a reduction from the problem of checking whether a tuple `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. (Abiteboul et al., 1995), (Vardi, 1982)). It is clear that every `DATALOG` is also a `DATALOG^{C(≠)}` program. Let `M` be a copying data exchange setting, that is, a (LAV) setting `(S, T, Σ_{st})` such that `S` contains all the relation symbols in `I`, `T` is a copy of `S` and `Σ_{st}` consists of dependencies of the form `R(\bar{x}) → R'(\bar{x})` for each relation symbol `R` in `S`, where `R'` is the copy in `T` of the relation `R` in `S`. From the results in chapter 3 and the construction of `CAN(I)`, it is straightforward to prove that `t ∈ Π(I)` if and only if `t ∈ certain_{M}(Q, I)`. □

It was shown in Theorem 4.1 that every conjunctive query with one inequality can be efficiently translated into a `DATALOG^{C(≠)}` program. Hence, the class of 1-CQ^{≠} queries form a subclass of the class of `DATALOG^{C(≠)}` programs. Thus, it is natural to ask whether the EXPTIME lower bound carries 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^{≠}

We leave the `DATALOG^{C(≠)}` queries to concentrate on the analysis of CQ^{≠} queries in data exchange. We first study the class 1-CQ^{≠}, that is, the class of conjunctive queries with only one inequality. It is worth mentioning that an EXPTIME lower bound can be obtained...
from (Kolaitis et al., 2006) for the case of unions of 1-CQ̸ queries. We refine this result to the case of 1-CQ̸ queries, and therefore present a stronger lower bound:

**Theorem 6.2.** Certain-Answers(GLAV, 1-CQ̸) is Exptime-complete.

*Proof(Sketch):* Membership in Exptime can be proved as follows. The certain answers to each union of conjunctive queries for a source instance I, under a setting M, can be computed in polynomial time in the size of CAN(I) (Fagin, Kolaitis, Miller, & Popa, 2005). But as we mentioned before, the size of CAN(I) is at most |I|^|Σst|. It follows that the certain answers to each union of conjunctive queries with respect to a source instance I, under a setting M, can be computed in exponential time in the size of I.

The proof of the lower bound is a refinement of a proof shown in (Kolaitis et al., 2006), where the theorem was proved for a union of two CQ̸ queries. The Exptime-hardness is established from a reduction from the Single Rule Datalog Problem (Gottlob & Papadimitriou, 2003), 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 an empty instance? That is, we ask whether \( \bar{t} \in Π(\emptyset) \). We shall call these programs Single Rule Datalog Programs (sirup). The combined complexity of this problem was shown to be Exptime-complete by Gottlob and Papadimitriou (2003). For the details of this proof, see the appendix A.

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 6.3.** Certain-Answers(GLAV, k-CQ̸) is Conexptime-complete, for every \( k \geq 2 \).

*Proof:* First, we prove the membership in Conexptime. The certain answers to each \( k \)-UCQ̸ query for a source instance I, under a setting M, can be computed in Conp time in
the size of $\text{CAN}(I)$ (Fagin, Kolaitis, Miller, & Popa, 2005). But as we mentioned before, the size of $\text{CAN}(I)$ is at most $|I|^{|\Sigma_{st}|}$. It follows that the certain answers to each $k$-$\text{UCQ}^\neq$ query $Q$ with respect to a source instance $I$, under a setting $\mathcal{M}$, can be computed in coNEXPTIME time in the size of $I$, $\mathcal{M}$ and $Q$.

The coNEXPTIME-hardness will be established with a reduction from the satisfiability problem for the Bernays-Schönfinkel class of Boolean FO formulas, which is known to be NEXPTIME-complete (see, e.g., (Börger, Grädel, & Gurevich., 2001)), to the complement of the problem $\text{CERTAIN-ANSWERS}(\text{GLAV}, 2$-$\text{CQ}^\neq)$. Formally, the Bernays-Schönfinkel class of Boolean FO formulas is defined as the class of all FO formulas of the form $\exists \bar{x} \forall \bar{y} \psi(\bar{x}, \bar{y})$, where $\psi$ is quantifier-free and does not contain function symbols. Along the proof we show the following: For every Boolean FO formula $\phi$ in the Bernays-Schönfinkel class, one can construct in polynomial time a GLAV data exchange setting $\mathcal{M} = (S, T, \Sigma_{st})$, a conjunctive query $Q$ with two inequalities, and an instance $I$ of $S$, such that $\phi$ is satisfiable iff $\text{certain}_\mathcal{M}(Q, I) = \text{false}$.

Fix a Boolean FO formula $\phi \equiv \exists x_1, \ldots, \exists x_p \forall y_1, \ldots, \forall y_m \psi$ in the Bernays-Schönfinkel class. Assume, without loss of generality, that the vocabulary of $\phi$ is constant-free and that $\psi$ does not contain atomic formulas of the form $x = y$. Also, let $\{R_1, \ldots, R_n\}$ be the set of all relation symbols mentioned in $\psi$. 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 $\phi$: Either $\phi$ is unsatisfiable, or it has a model of cardinality at most $p$ (see, e.g., (Börger et al., 2001)).

We split our reduction into two parts:

- First, we construct in polynomial time from $\phi$ a GLAV data exchange setting $\mathcal{M} = (S, T, \Sigma_{st})$, a query $Q$ that is a union of conjunctive queries with at most two inequalities per disjunct, and an instance $I$ of $S$, such that $\phi$ is satisfiable iff $\text{certain}_\mathcal{M}(Q, I) = \text{false}$. Although this is still not sufficient to prove the theorem, because $Q$ belongs to UCQ$^\neq$, the construction helps obtaining intuition for the second part of the proof, which is technically more involved.
Afterwards, using a refinement of the techniques in the first part of the proof, we construct in polynomial time from $\phi$ another G\textsc{lav} data exchange setting $\mathcal{M}' = (S', T', \Sigma_{st})$, a conjunctive query $Q'$ with two inequalities, and an instance $I'$ of $S'$, such that $\phi$ is satisfiable iff certain $\mathcal{M}'(Q', I') = \text{false}$.

Due to the technical characteristic of this proof, we only give the details for the first part of the reduction. For the second part, the reader is referred to the appendix A.

We need some additional notation. A subformula of $\psi$ is inductively defined as follows:

- If the atomic formula $R_i(\bar{z})$ appears in $\psi$, then $R_i(\bar{z})$ is a subformula of $\psi$;
- If $\xi$ and $\gamma$ are subformulas of $\psi$, then $(\neg \xi)$, $(\xi \lor \gamma)$, and $(\xi \land \gamma)$ are also subformulas of $\psi$.

Notice that in particular, $\psi$ is a subformula of itself. Let $S_1, \ldots, S_\ell$ be an enumeration of all the subformulas of $\psi$. Without loss of generality assume that $S_1 = \psi$.

The intuition of the first part of the reduction is the following. We construct a source instance $I$ such that $\text{dom}(I)$ consists of $p$ elements $a_1, \ldots, a_p$. This is justified by the fact, mentioned above, that if $\phi$ is satisfiable then it has a model of size at most $p$. We then construct a set $\Sigma_{st}$ of stds such that for each tuple $\bar{a}$ of size $r_i$ of elements in $\text{dom}(I)$, $i \in [1, n]$, $R'_i(\bar{a}, \bot)$ belongs to $\text{CAN}(I)$, where $\bot$ is a fresh null value. The target schema $T$ will contain one relation $F_j$ of arity $m + 1$ for each subformula $S_j$ of $\psi$ ($j \in [1, \ell]$), such that for each tuple $\bar{b}$ of size $m$ in $\text{dom}(I)$, $F_j(\bar{b}, \bot)$ belongs to $\text{CAN}(I)$, for every $j \in [1, \ell]$. We are interested in those solutions for $I$ in which each such null value is replaced by either value 0 or 1. With each such solution $J$ we naturally identify structure $A_J$ over the vocabulary $\{R_1, \ldots, R_k\}$ as follows: $\bar{a}$ belongs to the interpretation of the symbol $R_i$ in $A_J$ iff $R'_i(\bar{a}, 1) \in J$. Moreover, for $j \in [1, \ell]$, $F_j(\bar{b}, 1) \in J$ iff $A_J$ satisfies subformula $S_j$ when we respectively assign each element from $\bar{b}$ to variables $y_1, \ldots, y_m$ and elements $a_1, \ldots, a_p$ to variables $x_1, \ldots, x_p$.

We begin now the first part of the reduction.
• The source schema $S$ 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 $\phi$), two ternary relations $C$ and $D$, and one binary relation $E$.

• The target schema $T$ 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 $\ell$ is the number of subformulas of $\psi$, and that $m$ is the number of universally quantified variables of $\phi$).

• 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 of each symbol in $S$ in $I$ is as follows:
  - $B^I = \{a_1, \ldots, a_p\}$;
  - $O^I = \{0\}$ and $U^I = \{1\}$.
  - $C^I = \{(1, 1, 1), (1, 0, 1), (0, 1, 1), (0, 0, 0)\}$
  - $D^I = \{(1, 1, 1), (1, 0, 0), (0, 1, 0), (0, 0, 0)\}$
  - and $E^I = \{(0, 1), (1, 0)\}$.
  - Finally, $V^I_i = \{a_i\}$ for each $i \in [1, p]$.

• The set $\Sigma_{st}$ of source-to-target dependencies is as follows:
We create a copy of every relation $V_i$ into $V_i'$. We also create a copy of $O$, $U$, $C$, $D$ and $E$ into $O'$, $U'$, $C'$, $D'$ and $E'$, respectively:

\[
\begin{align*}
V_1(x) & \rightarrow V_1'(x) \\
\vdots \\
V_p(x) & \rightarrow V_p'(x) \\
O(x) & \rightarrow O'(x) \\
U(x) & \rightarrow U'(x) \\
C(x, y, z) & \rightarrow C'(x, y, z) \\
D(x, y, z) & \rightarrow D'(x, y, z) \\
E(x, y) & \rightarrow E'(x, y)
\end{align*}
\]

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:

\[
\begin{align*}
B(x_1), ..., B(x_{r_1}) & \rightarrow \exists z R_1'(x_1, ..., x_{r_1}, z) \\
\vdots \\
B(x_1), ..., B(x_{r_n}) & \rightarrow \exists z R_n'(x_1, ..., x_{r_n}, z)
\end{align*}
\]

As we mentioned before, we are interested in those solutions for $I$ that replace each such null value with either 0 or 1. Informally, with each such solution $J$ for $I$ 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, 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:

$$B(x_1), ..., B(x_m) \rightarrow \exists z F_1(x_1, ..., x_m, z)$$

$$\vdots$$

$$B(x_1), ..., B(x_m) \rightarrow \exists z F_\ell(x_1, ..., x_m, z)$$

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

It is clear at this point what is the canonical universal solution, $\text{CAN}(I)$, for $I$.

Before presenting the query $Q$, we give an intuition of what $Q$ does: 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 $\bot$ such that the tuple $R_i(\bar{a}, \bot)$ belongs to $\text{CAN}(I)$. Afterwards, for every solution $J$ in which there are only values 1 or 0 in the last position of the tuples in a relation $R_i$, $i \in [1, \ldots, k]$, the query will assign a value 1 (resp. 0) to every null $\bot$ such that the tuple $F_j(\bar{b}, \bot)$ is in $\text{CAN}(I)$ for a relation in $\{F_1, \ldots, F_\ell\}$, and $S_j$ holds (resp. does not hold) in $A_J$ when we respectively assign each element from $\bar{b}$ to variables $y_1, \ldots, y_m$ and elements $a_1, \ldots, a_p$ to variables $x_1, \ldots, x_p$. Finally, the query will ask for a tuple in $J$ of the form $F_1(\bar{c}, 0)$. If there are no such tuples, then for every assignment of the universally quantified variables it will be the case that $S_1$ holds in $A_J$.

The query $Q$ is defined as $Q_\alpha \lor Q_\beta \lor Q_\gamma \lor Q_\delta$, where

- $Q_\alpha$ is $\bigcup_{i \in [1,n]} Q^i_\alpha$, where each $Q^i_\alpha$ is defined as follows:

$$\exists z_1 \ldots \exists z_r \exists n \exists v \exists w (R_i'(z_1, ..., z_r, n) \land O'(v) \land U'(W) \land n \neq v \land n \neq w)$$

Note that if the evaluation of $Q_\alpha$ over a solution is false, then all the nulls in the relations $R_i$ of that solution must have been replaced by 0 or 1.
Let $\Theta \subseteq \{1, \ldots, \ell\}$ be the set of all indexes $j$ such that $S_j$ is an atomic formula. The query $Q_\beta$ is defined as $\bigcup_{j \in \Theta} Q^j_\beta$, 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 $Q^j_\beta$ is as follows:

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

Assume that $S_j$ holds (resp., does not hold) in $A_J$ 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 $x_1, \ldots, x_p$, where $J$ is a solution that falsifies $Q^j_\beta$. If $F_j(a_{i_1}, \ldots, a_{i_m}, \bot)$ belongs to $\text{CAN}(J)$ then $\bot$ must have been replaced by 1 (resp. 0) in $J$.

$Q_\gamma$ is defined as $\bigcup_{k \not\in \Theta} Q^k_\gamma$, where each query $Q^k_\gamma$ is defined as follows:

(i) If $S_k \equiv S_g \lor S_h$, then

$$Q^k_\gamma \equiv \exists y_1 \ldots \exists y_m \exists n \exists v \exists w \exists z \left( F_k(y_1, \ldots, y_m, n) \land F_g(y_1, \ldots, y_m, v) \land F_h(y_1, \ldots, y_m, w) \right. \left. \land D'(v, w, z) \land n \neq z \right).$$

(ii) If $S_k \equiv S_g \land S_h$, then

$$Q^k_\gamma \equiv \exists y_1 \ldots \exists y_m \exists n \exists v \exists w \exists z \left( F_k(y_1, \ldots, y_m, n) \land F_g(y_1, \ldots, y_m, v) \land F_h(y_1, \ldots, y_m, w) \right. \left. \land C'(v, w, z) \land n \neq z \right).$$

(iii) If $S_k \equiv \neg S_g$, then

$$Q^k_\gamma \equiv \exists y_1 \ldots \exists y_m \exists n \exists v \exists w \exists z \left( F_k(y_1, \ldots, y_m, n) \land F_g(y_1, \ldots, y_m, v) \land E'(v, w, z) \land n \neq z \right).$$

The purpose of this query is similar to $Q_\beta$, but here we ensure the correct interpretation of subformulas of $\psi$ that are Boolean combinations of other subformulas. Recall that the tuples in relations $C'$, $D'$ and $E'$ only encode the truth tables of $\land$, $\lor$, and $\neg$, respectively. For example, if $S_k \equiv (S_g \land S_h)$, we will only set the null of $F_k(\bar{a}, \bot)$ to be 1 if there are tuples $F_g(\bar{a}, 1)$ and $F_h(\bar{a}, 1)$. 
Finally, \( Q_\delta \) is defined to be \( \exists y_1 \ldots \exists y_m \exists v (F_1(y_1, \ldots, y_m, v) \land O'(v)) \). This query asks for a tuple of the form \( F_1(\bar{a}, 0) \). That is, this query will be falsified by a solution \( J \) if and only if none of the tuples in the interpretation of \( F_1 \) in \( J \) contains a 0 in the last position.

**Example 6.1.** Let \( \phi \) be the formula \( \exists x_1 \exists x_2 \forall y_1 (R_1(x_1, y_1) \lor (\neg R_1(x_2, y_1))) \). Recall that the source schema \( S \) consists of relations \( B, O, U, C, D, E \) as described above, plus extra relations \( V_1 \) and \( V_2 \). The target schema \( T \) consists of relations \( O', U', C', D', E', R'_1, F_1, F_2, F_3 \) and \( F_4 \) (because \( (R_1(x_1, y_1) \lor (\neg R_1(x_2, y_1))) \) has 4 subformulas). The enumeration of the subformulas is chosen such that \( F_3 \) and \( F_4 \) correspond to the subformulas \( S_3 \equiv R_1(x_1, y_1) \) and \( S_4 \equiv R_1(x_2, y_1) \), respectively, \( F_2 \) corresponds to \( S_2 \equiv (\neg R_1(x_2, y_2)) \) and \( F_1 \) corresponds to \( S_1 \equiv (R_1(x_1, y_1) \lor (\neg R_1(x_2, y_1))) \).

The source-to-target dependencies are:

\[
\begin{align*}
V_1(x) & \rightarrow V'_1(x) \\
O(x) & \rightarrow O'(x) \\
U(x) & \rightarrow U'(x) \\
C(x, y, z) & \rightarrow C'(x, y, z) \\
D(x, y, z) & \rightarrow D'(x, y, z) \\
E(x, y) & \rightarrow E'(x, y) \\
B(x_1), B(x_2) & \rightarrow \exists z R'_1(x_1, x_2, z) \\
B(x_1) & \rightarrow \exists z F_1(x_1, z)
\end{align*}
\]
\[ B(x_1) \rightarrow \exists z F_2(x_1, z) \quad B(x_1) \rightarrow \exists z F_3(x_1, z) \quad B(x_1) \rightarrow \exists z F_4(x_1, z) \]

The instance \( I \) of \( S \) is constructed as follows: \( B^I = \{a_1,a_2\}, 0^I = \{0\} \) and \( U^I = \{1\} \). Furthermore, \( C^I = \{(1,1,1), (1,0,1), (0,1,1), (0,0,0)\} \), \( D^I = \{(1,1,1), (1,0,0), (0,1,0), (0,0,0)\} \), and \( E^I = \{(0,1),(1,0)\} \). Finally, \( V_1^I = \{a_1\}, V_2^I = \{a_2\} \).

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 \( Q_\alpha, Q_\beta, Q_\gamma \) and \( Q_\delta \) in this case are as follows:

- \( Q_\alpha \equiv \exists x_1 \exists x_2 \exists n \exists v \exists w (R'_1(x_1, x_2, n) \land O'(v) \land U'(w) \land n \neq v \land n \neq w) \);
- \( Q_\beta \) is the union of \( Q^3_\beta \) and \( Q^4_\beta \), where:
  \[
  Q^3_\beta \equiv \exists y_1 \exists n \exists v \exists x_1 (F_3(y_1, n) \land R'_1(x_1, y_1, w) \land V'_1(x_1) \land n \neq w) \\
  Q^4_\beta \equiv \exists y_1 \exists n \exists v \exists x_2 (F_4(y_1, n) \land R'_1(x_2, y_1, w) \land V'_2(x_2) \land n \neq w)
  \]
- \( Q_\gamma \) is the union of \( Q^1_\gamma \) and \( Q^2_\gamma \), where:
  \[
  Q^1_\gamma \equiv \exists y_1 \exists n \exists v \exists z (F_1(y_1, n) \land F_3(y_1, v) \land F_2(y_1, w) \land D'(v, w, z) \land n \neq z) \\
  Q^2_\gamma \equiv \exists y_1 \exists n \exists v \exists z (F_2(y_1, n) \land F_4(y_1, v) \land E'(v, z) \land n \neq z)
  \]
\( Q_\delta \equiv \exists y_1 \exists v (F_1(y_1, v) \land O'(v)) \)

This finishes the example.

We now continue with the proof of the theorem. We prove next that \( \phi \) is satisfiable if and only if \( \text{certain}_M(Q, I) = \text{false} \).

\((\Leftarrow)\) Assume first that \( \phi \) is satisfiable. Then, as we mentioned above, it is satisfiable by a structure of cardinality at most \( p \). Let \( A \) be such structure, and assume that the elements of \( A \) are \( \{b_1, \ldots, b_p\} \). Further, assume without loss of generality that \( A \) satisfies \( \forall y_1, \ldots, \forall y_m \psi \) when we assign to each free variable \( x_i \) in \( \psi \) the element \( b_i, i \in [1, p] \). Define a function \( h \) from \( \text{CAN}(I) \) into \( \text{CAN}(I) \) such that for every element \( v \) in \( \text{CAN}(I) \):

- If \( v \) is the constant \( c \), then \( h(v) = v \);
- if \( v \) is the null value \( \bot \) such that the tuple \( R'_i(a_{i_1}, \ldots, a_{i_r}, \bot) \) belongs to \( \text{CAN}(I) \), and the interpretation of \( R_i \) in \( A \) contains the tuple \( (b_{i_1}, \ldots, b_{i_l}) \), then \( h(v) = 1 \);
- if \( v \) is the null value \( \bot \) such that the tuple \( F'_j(a_{i_1}, \ldots, a_{i_m}, \bot) \) belongs to \( \text{CAN}(I) \), and such that the formula \( S_j \) holds in \( A \) whenever we assign to the universally quantified variables \( y_1, \ldots, y_m \) the elements \( b_{i_1}, \ldots, b_{i_m}, b_{i_l} \in \{b_1, \ldots, b_p\}, l \in [1, p] \), and to the existentially quantified variables \( x_1, \ldots, x_p \) the elements \( b_1, \ldots, b_p \), then \( h(v) = 1 \); and
- otherwise, \( h(v) = 0 \)

Let \( J^* \) the solution obtained by replacing each occurrence of element \( v \) in \( \text{CAN}(I) \) by \( h(v) \). We show that the evaluation of \( Q(J^*) = \text{false} \), and, thus, that \( \text{certain}_M(Q, I) = \text{false} \).

Assume for the sake of contradiction that \( Q(J^*) = \text{true} \). Clearly, \( Q_\alpha(J^*) = \text{false} \). Furthermore, \( Q_\delta(J^*) = \text{false} \): Since \( A \) satisfies \( \forall y_1, \ldots, \forall y_p \psi \) when we assign the element \( b_i \) to \( x_i \), for each \( i \in [1, p] \), then it must be the case that \( S_1 \) holds in \( A \) whenever we assign \( b_{i_1}, \ldots, b_{i_m} \) to the universally quantified variables and \( b_1, \ldots, b_p \) to the existentially quantified variables.
Assume first that $Q_\beta(J^*) = \text{true}$, and, in particular, that $Q_j^\prime(J^*) = \text{true}$, for $j \in A$. Further, assume that $Q_j^\prime$ is of the form

$$
\exists y_1, \ldots, \exists y_m \exists n \exists v \exists \bar{x} (F_j(y_1, \ldots, y_m, n) \land R_i'(\bar{x}, \bar{y}, w) \land \bigwedge_{x_k \in \bar{x}} V'_k(x_k) \land n \neq \bar{w}),
$$

where $\bar{x}$ is a tuple of elements in $\{x_1, \ldots, x_p\}$ and $\bar{y}$ is a tuple of elements in $\{y_1, \ldots, y_m\}$. Then there exists a function $f : \{y_1, \ldots, y_m, n, v, \bar{x}\} \rightarrow \text{dom}(J^*)$, such that $F_j(f(y_1), \ldots, f(y_m), f(n), R'_i(f(\bar{x}), f(\bar{y}), f(w))$ and $V'_k(f(x_k))$, for each $k$ such that $x_k \in \bar{x}$, belong to $J^*$. Further, $f(n) \neq f(w)$.

Assume, without loss of generality that $f(n) = 0$ (the case when $f(n) = 1$ is completely symmetrical). Then $\mathcal{A}$ does not satisfy $S_j = R_i(\bar{x}, \bar{y})$, whenever we assign elements $b_1, \ldots, b_p$ to $x_1, \ldots, x_p$ and elements $f(y_1), \ldots, f(y_m)$ to $y_1, \ldots, y_m$. Since $f(n) \neq f(w)$, it must be the case that $f(w) = 1$. But then $R'_i(\bar{x}, \bar{y})$ holds in $\mathcal{A}$ whenever we assign elements $b_1, \ldots, b_p$ to $x_1, \ldots, x_p$ and elements $f(y_1), \ldots, f(y_p)$ to $y_1, \ldots, y_m$. This is a contradiction.

Assume then that $Q_\gamma(J^*) = \text{true}$, and, in particular, that $Q_k^\gamma(J^*) = \text{true}$, for some $k \notin \Theta$. Further, assume without loss of generality that $S_k$ is of the form $S_g \lor S_h$ (the other two cases are completely symmetrical). Also assume $Q_\gamma$ is of the form

$$
\exists y_1 \ldots \exists y_m \exists n \exists v \exists w \exists z (F_k(y_1, \ldots, y_m, n) \land F_g(y_1, \ldots, y_m, v) \land F_h(y_1, \ldots, y_m, w) \land D'(v, w, z) \land n \neq z).
$$

Then there exists a function $f : \{y_1, \ldots, y_n, v, w, z\} \rightarrow \text{dom}(J^*)$, such that $F_k(f(y_1), \ldots, f(y_m), f(n)), F_g(f(y_1), \ldots, f(y_m), f(v)), F_h(f(y_1), \ldots, f(y_m), f(w))$, and $D'(f(v), f(w), f(z))$, belong to $J^*$. Further, $f(n) \neq f(z)$.

Assume, without loss of generality, that $f(n) = 1$ (the case $f(n) = 0$ is completely symmetrical). Then $\mathcal{A}$ satisfies $S_k \equiv (S_g \lor S_h)$, whenever we assign elements $b_1, \ldots, b_p$ to $x_1, \ldots, x_p$ and elements $f(y_1), \ldots, f(y_p)$ to $y_1, \ldots, y_m$. Since $f(n) \neq f(z)$, it must be the case that $f(z) = 0$. Then, from the construction of $D$, it must be the case that $f(v) = 0$.
and \( f(w) = 0 \). But then \( \mathcal{A} \) does not satisfy neither \( S_g \) nor \( S_h \) whenever we assign elements \( b_1, \ldots, b_p \) to \( x_1, \ldots, x_p \) and elements \( f(y_1), \ldots, f(y_p) \) to \( y_1, \ldots, y_m \). This is a contradiction.

\( \Rightarrow \) Assume certain, \( \mathcal{M}(Q, I) = \text{false} \). Then there exists a solution \( J' \) for \( I \) such that \( Q(J') = \text{false} \). Let \( h \) be a homomorphism from \( \text{CAN}(I) \) to \( J' \).

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 \( R_i \) in \( \mathcal{A} \), \( i \in [1, n] \), is the following: The tuple \( \bar{a} \) belongs to the interpretation of \( R_i \) in \( \mathcal{A} \) iff \( R'_i(\bar{a}, 1) \) belongs to \( J' \). We show next that \( \mathcal{A} \) satisfies \( \phi \). In order to do so we start by proving the following: We prove that for every \( j \in [1, \ell] \), if the tuple \( F_j(a_{i_1}, \ldots, a_{i_m}, 1) \) belongs to \( J' \), then \( \mathcal{A} \) satisfies the formula \( S_j \) when we assign \( a_{i_1}, \ldots, a_{i_m} \) 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 \( \psi \).

- For the base case, assume that \( S_j = R_i(\bar{x}, \bar{y}) \), where \( \bar{x} \) is a tuple of variables in \( \{x_1, \ldots, x_p\} \) and \( \bar{y} \) is a tuple of variables in \( \{y_1, \ldots, y_m\} \). Further, assume that \( F_j(b_{i_1}, \ldots, b_{i_m}, 1) \) belongs to \( J' \).

Let \( g : \{x_1, \ldots, x_p, y_1, \ldots, y_m\} \rightarrow \text{dom}(J') \) be a function, such that \( g(x_i) = a_i \), for each \( i \in [1, p] \), and \( g(y_j) = a_{i_j} \), for each \( j \in [1, m] \). We know that \( \text{CAN}(I) \) contains a tuple \( R'_i(g(\bar{x}), g(\bar{y}), \bot) \), for some null value \( \bot \), and, thus, \( J' \) contains the tuple \( R'_i(g(\bar{x}), g(\bar{y}), h(\bot)) \). Further, since \( Q^j_\beta \) is of the form

\[
\exists y_1, \ldots, \exists y_m \exists n \exists v \exists \bar{x} (F_j(y_1, \ldots, y_m, n) \land R'_i(\bar{x}, \bar{y}, w) \land \bigwedge_{x_k \in \bar{x}} V'_k(x_k) \land n \neq w),
\]

and we know that \( Q^j_\beta(J') = \text{false} \), it must be the case that \( h(\bot) = 1 \). It follows that \( R'_i(g(\bar{x}), g(\bar{y}), 1) \) belongs to \( J' \), and, by definition of \( \mathcal{A} \), that \( \mathcal{A} \) satisfies \( S_j \) when we assign \( a_{i_1}, \ldots, a_{i_m} \) to \( y_1, \ldots, y_m \) and \( a_1, \ldots, a_p \) to \( x_1, \ldots, x_p \).

- For the inductive case, assume that \( S_j \equiv (S_j_{11} \land S_j_{22}) \) (the cases where \( S_j \equiv (S_{j_{11}} \lor S_{j_{22}}) \) and \( S_j \equiv (\neg S_{j_{11}}) \) are completely symmetrical). Further, assume that \( F_j(a_{i_1}, \ldots, a_{i_m}, 1) \) belongs to \( J' \). We know there are tuples \( F_{j_{11}}(a_{i_1}, \ldots, a_{i_m}, \bot_{i_1}) \) and \( F_{j_{22}}(a_{i_1}, \ldots, a_{i_m}, \bot_{i_2}) \) in \( \text{CAN}(I) \), where \( \bot_{i_1} \) and \( \bot_{i_2} \) are null values. Therefore, \( J' \) contains the tuples \( F_{g}(a_{i_1}, \ldots, a_{i_m}, h(\bot_{i_1})) \) and \( F_{h}(a_{i_1}, \ldots, a_{i_m}, h(\bot_{i_2})) \).
We know that $Q^k_\gamma$ is of the form

$$\exists y_1 \ldots \exists y_m \exists n \exists v \exists w \exists z (F_k(y_1, \ldots, y_m, n) \land F_g(y_1, \ldots, y_m, v) \land F_h(y_1, \ldots, y_m, w) \land C'(v, w, z) \land n \neq z).$$

Further, $Q^k_\gamma(J') = \text{false}$. Then it must be the case that $h(\perp_{j_1}) = h(\perp_{j_2}) = 1$. It follows that $F_{j_1}(a_{i_1}, \ldots, a_{i_m}, 1)$ and $F_{j_2}(a_{i_1}, \ldots, a_{i_m}, 1)$ belong to $J'$.

By the inductive hypothesis, $\mathcal{A}$ satisfies $S_{j_1}$ and $S_{j_2}$ whenever we assign elements $a_{i_1}, \ldots, a_{i_m}$ to $y_1, \ldots, y_m$ and elements $a_1, \ldots, a_p$ to $x_1, \ldots, x_p$. It follows that $\mathcal{A}$ satisfies $S_j$ whenever we assign elements $a_{i_1}, \ldots, a_{i_m}$ to $y_1, \ldots, y_m$ and elements $a_1, \ldots, a_p$ to $x_1, \ldots, x_p$.

Finally, since $Q^\phi(J') = \text{false}$, for every tuple $a_{i_1}, \ldots, a_{i_m}$ of elements in $\{a_1, \ldots, a_p\}$ it must be the case that $(a_{i_1}, \ldots, a_{i_m}, 1)$ is in the interpretation of $F_1$ in $J'$. From the previous induction, we obtain that $\mathcal{A}$ satisfies $S_1 = \psi$ for every assignment of the variables $y_1, \ldots, y_m$, when we assign $b_1, \ldots, b_p$ to $x_1, \ldots, x_p$. In other words, $\mathcal{A}$ satisfies $\phi$. This concludes the first part of the reduction.

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$^{C(\neq)}$ 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 6.2.** Certain-Answers(LAV, 1-CQ$^\neq$) is NP-complete, and for every $k \geq 2$, Certain-Answers(LAV, k-CQ$^\neq$) is $\Pi^p_2$-complete.
Proof: We only show that \textsc{Certain-Answers}(LAV, k-CQ*) is $\Pi^p_2$-complete. That the problem \textsc{Certain-Answers}(LAV, 1-CQ*) is \textsc{NP}-complete can be proved using techniques in (Fagin, Kolaitis, Miller, & Popa, 2005) for membership, and in (Kolaitis et al., 2006) for hardness.

First we prove the membership in $\Pi^p_2$. The certain answers to each $k$-UCQ* queries for a source instance $I$, under a setting $\mathcal{M}$, can be computed in \textsc{coNP} time in the size of $\text{CAN}(I)$ (Fagin, Kolaitis, Miller, & Popa, 2005). Specifically, the algorithm will use the chase to materialize a solution $J$ for which the certain answers are empty if and only if the evaluation of $Q$ over $J$ is empty. Contrary to Theorem 6.3, for the case of LAV settings, it is clear that $\text{CAN}(I)$ is bounded by $|I|$. However, the results in (Fagin, Kolaitis, Miller, & Popa, 2005) assume the query is fixed. If the query is part of the input, we will need a \textsc{NP} oracle in order to check that the evaluation of $Q$ over $J$ is empty, thus obtaining a $\Pi^p_2$ bound.

The $\Pi^p_2$ hardness is established from a reduction of $\forall \bar{x} \exists \bar{z} \text{3-SAT}(\bar{x}, \bar{z})$, which is the following problem: Given a boolean formula $\phi$ in 3-CNF with variables partitioned into two sets, $\bar{x}$ and $\bar{z}$. Is it true that for all truth assignments of the variables in $\bar{x}$ there is a valuation for the variables in $\bar{z}$ so that $\phi'$ is satisfied with the overall assignment? This problem is widely known to be $\Pi^p_2$-complete.

Let the formula be of the form $\phi \equiv \forall \bar{x} \exists \bar{z} \bigwedge_{1 \leq k \leq \ell} \psi_k$, where each $\psi_k$, $1 \leq k \leq \ell$ is a clause. Let $\bar{x} = \{x_1, ..., x_n\}$ and $\bar{z} = \{z_1, ..., z_m\}$. Based on $\phi$, we will show a LAV data exchange setting $\mathcal{M} = (S, T, \Sigma_{st})$, a query $Q$ and an instance $I$ such that $\text{certain}_{\mathcal{M}}(Q, I) = \text{true}$ if and only if $\phi$ is satisfiable.

- the LAV setting $\mathcal{(S,T,\Sigma_{st})}$ is as follows:

  The Source schema $S$ consists of two unary relations $\hat{I}$, $\hat{O}$, two binary relations $F$, $S$, one tertiary relation $E$ and one octiary relation $\hat{C}$. The Target Schema $G$ consists of two unary relations $I$, $O$, two binary relations $T$, $TV$, one tertiary relation $R$, and one octiary relation $C$.  

The set $\Sigma_{st}$ of source-to-target dependencies is:

\begin{align*}
\hat{O}(x) & \rightarrow O(x) \tag{6.1} \\
\hat{I}(x) & \rightarrow I(x) \tag{6.2} \\
F(x, y) & \rightarrow \exists z, w(T(x, z), R(y, x, z), TV(x, z), R(x, w, w), C(w, w, w, w, w, w, w, w)) \tag{6.3} \\
S(x, y) & \rightarrow T(x, y) \tag{6.4} \\
E(x, y, z) & \rightarrow TV(x, z), R(y, x, z) \tag{6.5} \\
\hat{C}(a, b, c, d, e, f, g, h) & \rightarrow C(a, b, c, d, e, f, g, h) \tag{6.6}
\end{align*}

- The elements of the source instance $I$ are $\hat{x}_1, ..., \hat{x}_n, \hat{z}_1, ..., \hat{z}_m$ plus the additional constants $0, 1, a, b$ and $p_1, ..., p_\ell, p_{\ell+1}$. The interpretation of the relations in $I$ are as follows:

  - The interpretations of relations $\hat{I}$ and $\hat{O}$ contain the constants 1 and 0, respectively.
  - The interpretation of the relation $F$ contains the pairs $(\hat{x}_i, a)$, $i \in \{1, ..., n\}$
  - The interpretation of the relation $S$ contains the pair $(a, b)$
  - The interpretation of the relation $E$ contains the triples $(\hat{z}_j, a, 1)$ and $(\hat{z}_j, a, 0)$, $j \in \{1, ..., m\}$
  - The interpretation of the relation $\hat{C}$ contains seven tuples of the form $(\hat{s}_{1k}, \hat{s}_{2k}, \hat{s}_{3k}, u, v, w, p_k, p_{k+1})$ for each clause $\psi_k = (s_{1k} \lor s_{2k} \lor s_{3k})$ in $\phi$ (1 $\leq k \leq \ell$). The variables $u, v, w$ represent the values of satisfying assignments for $\psi_k$ and the two last positions $p_k, p_{k+1}$ are used to denote that a particular tuple corresponds to the k-th clause of $\phi$. For example, if $\phi \equiv (x_1 \lor \neg x_2 \lor z_1)$, then $\hat{C}$ will contain the following seven tuples: $(x_1, x_2, z_1, 0, 0, 0, p_1, p_2)$, $(x_1, x_2, z_1, 0, 0, 1, p_1, p_2)$, $(x_1, x_2, z_1, 0, 1, 1, p_1, p_2)$, $(x_1, x_2, z_1, 1, 0, 0, p_1, p_2)$, $(x_1, x_2, z_1, 1, 0, 1, p_1, p_2)$, $(x_1, x_2, z_1, 1, 1, 0, p_1, p_2)$ and
(x_1, x_2, z_1, 1, 1, 1, p_1, p_2). No tuple (x_1, x_2, z_1, 0, 1, 0, p_1, p_2) occur in S, because (0, 0, 1) is not a satisfying assignment for \( \phi \)

- Finally, the query \( Q \) is as follows:

\[
Q \equiv \exists e \exists f \exists g \exists u \exists x_1, \ldots, x_n \exists v_1, \ldots, v_m \exists z_1, \ldots, z_m \exists w_1, \ldots, w_m \exists y_1, \ldots, y_{\ell + 1} \bigg( \bigwedge_{1 \leq i \leq n} (TV(x_i, v_i) \land R(g, x_i, v_i)) \land \bigwedge_{1 \leq j \leq m} (TV(z_j, w_j) \land R(g, z_j, w_j)) \land \bigwedge_{1 \leq k \leq p} (C(s_1, s_2, s_3, v_1, v_2, v_3, y_k, y_{k + 1})) \land I(e) \land O(f) \land T(g, u) \land u \neq e, u \neq f \bigg)
\]

Let \( \#_i \) and \( \perp_i \) be the null values obtained from the application of std (3) to the tuple \((\hat{x}_i, a)\). Note that the canonical solution \( J \) for \( I \) under \( \mathcal{M} \) is as follows:

- The interpretation of \( I \) and \( O \) in \( J \) contains the constants 1 and 0, respectively
- The interpretation of \( T \) in \( J \) contains the tuple \((a, b)\) and the tuples \((\hat{x}_i, \perp_i)\) for \( i \in \{1, \ldots, n\} \)
- The interpretation of \( TV \) in \( J \) contains tuples \((\#_i, \#_i)\) and \((\hat{x}_i, \perp_i)\) for each universally quantified variable \( \hat{x}_i \) in \( \phi \) and the pairs \((\hat{z}_j, 0)\), \((\hat{z}_j, 1)\) for each existentially quantified variable \( \hat{z}_j \).
- The interpretation of \( R \) in \( J \) contains tuples \((\hat{x}_i, \#_i, \#_i)\) and \((a, \hat{x}_i, \perp_i)\) for each universally quantified variable \( \hat{x}_i \) in \( \phi \) and the pairs \((a, \hat{z}_j, 0)\), \((a, \hat{z}_j, 1)\) for each existentially quantified variable.
- The interpretation of \( C \) contains a copy of the interpretation of \( \hat{C} \) plus one tuple \((\#_i, \#_i, \#_i, \#_i, \#_i, \#_i, \#_i, \#_i)\) for \( i \in \{1, \ldots, n\} \).

We first illustrate the idea of this reduction with an example.

Suppose \( \phi \equiv \forall x_1, x_2 \exists z_1 (x_1 \lor \neg x_2 \lor \neg z_1) \). We then create the following \textsc{Lav} setting \((S, T, \Sigma_{st})\):

- The elements of \( I \) are: \( \hat{x}_1, \hat{x}_2, \hat{z}_1, 0, 1, a, b, p_1, p_2 \)
The relations of $I$ are:
- $\hat{I} = \{(1)\}$, $\hat{O} = \{(0)\}$, $S = \{(a, b)\}$
- $F = \{((\hat{x}_1, a), (\hat{x}_2, a))\}$
- $E = \{((\hat{z}_1, a, 1), (\hat{z}_1, a, 0))\}$
- $\hat{C} = \{((\hat{x}_1, \hat{x}_2, \hat{z}_1, 0, 0, p_1, p_2), (\hat{x}_1, \hat{x}_2, \hat{z}_1, 0, 0, 1, p_1, p_2), (\hat{x}_1, \hat{x}_2, \hat{z}_1, 1, 0, 1, p_1, p_2), (\hat{x}_1, \hat{x}_2, \hat{z}_1, 1, 1, 1, p_1, p_2)\}$

The canonical solution contains four null values: $\#_1$, $\bot_1$ and $\#_2$, $\bot_2$, obtained from the application of std (3) to the tuples $(\hat{x}_1, a)$ and $(\hat{x}_2, a)$ respectively. The relations in the canonical universal solution $J$ are as follows:

- $I = \{(1)\}$, $O = \{(0)\}$
- $T = \{((\hat{x}_1, \bot_1), (\hat{x}_2, \bot_2), (a, b))\}$
- $TV = \{((\hat{x}_1, \bot_1), (\hat{x}_2, \bot_2), (\hat{z}_1, 1), (\hat{z}_1, 0), (\#_1, \#_1), (\#_2, \#_2))\}$
- $R = \{(a, \hat{x}_1, \bot_1), (a, \hat{x}_2, \bot_2), (a, \hat{z}_1, 1), (a, \hat{z}_1, 0), (\hat{x}_1, \#_1, \#_1), (\hat{x}_2, \#_2, \#_2)\}$
- $C = \{((\hat{x}_1, \hat{x}_2, \hat{z}_1, 0, 0, p_1, p_2), (\hat{x}_1, \hat{x}_2, \hat{z}_1, 0, 0, 1, p_1, p_2), (\hat{x}_1, \hat{x}_2, \hat{z}_1, 1, 0, 0, p_1, p_2), (\hat{x}_1, \hat{x}_2, \hat{z}_1, 1, 0, 1, p_1, p_2), (\hat{x}_1, \hat{x}_2, \hat{z}_1, 1, 1, 1, p_1, p_2), (\#_1, \#_1, \#_1, \#_1, \#_1, \#_1, \#_1, \#_1, \#_1, \#_1, \#_1, \#_1, \#_1)\}$

Finally, the Query $Q$ will be the existential closure of:

$$Q' \equiv I(e) \land O(f) \land T(x, u) \land TV(x_1, v_1) \land TV(x_2, v_2) \land R(x, x_1, v_1) \land R(x, x_2, v_2) \land TV(z_1, w_1) \land R(x, z_1, w_1) \land C(x_1, x_2, z_1, v_1, v_2, w_1, v_1, y_1, y_2) \land u \neq e \land u \neq f$$

Intuitively, the relation $TV$ contains the truth value of every variable in $\phi$, that is, a pair $(\hat{q}, 0)$ represents that the valuation of $q$ is false and a pair $(\hat{q}, 1)$ represents a true value for
The variables in $\bar{x}$ will have a pair $TV(\hat{x}_i, \perp_i)$ in the canonical solution for $I$, while the variables in $\bar{z}$ will have both tuples $(\hat{z}_j, 0)$ and $(\hat{z}_j, 1)$.

It is clear that no tuple of the form $T(a, b)$ will belong to the evaluation of $Q$ over the canonical solution. Instead, all of the tuples $T$ witnessing $Q$ will be pairs of the form $(\hat{x}_i, \perp_i)$. Informally, the query will be false only if evaluated in solutions where all the $\perp$’s have been replaced with the values 0 or 1. We are interested only in those solutions, because they represent a particular valuation for the variables in $\bar{x}$. Suppose that a particular solution $J^*$ has all of the $\perp$ replaced by 0 or 1. In order for the evaluation of $Q$ over $A_J$ to be true, there must exist witnesses for all of the relations $TV$ in $Q$. From the construction of the relation $C$, there must be one tuple of the form $(\hat{q}, v)$ for every variable $q$ mentioned in $\phi$. Due to $\Sigma_{st}$, for the case of the universally quantified variables $x_i, i \in \{1, ..., n\}$ there exists only one tuple per variable. On the contrary, for each existentially quantified variable $z_i, i \in \{1, ..., m\}$ there will be two tuples: $TV(\hat{z}_i, 1)$ and $TV(\hat{z}_i, 0)$. If we are able to find all of the witnesses we need, intuitively, from the construction of the tuples in relation $C$, we will be able to find a valuation for the existentially quantified variables, and the evaluation of $Q$ over $J^*$ will be true. Naturally, if we can do this for all the solutions $J$ that represent a valuation of the universally quantified set, then $\phi$ will be $\forall \exists$-satisfiable.

We now prove that $\text{certain}_M(Q, I) = \text{false}$ if and only if $\phi$ is not $\forall \exists$-satisfiable.

$(\Leftarrow)$ If $\phi$ is not $\forall \exists$-satisfiable, that is, there is valuation $\sigma_x$ for the universally quantified variables such that for every valuation $\sigma_z$ for the existentially quantified variables $\phi$ is not satisfiable. Define a function $h$ as follows:

- $h(y) = 1$ if $y = \perp_i$ belongs to a tuple $TV(\hat{x}_i, \perp_i)$, $x_i$ is a propositional variable mentioned in $\phi$ and $\sigma_z(x_i) = 1$
- $h(y) = 0$ if $y = \perp_i$ belongs to a tuple $TV(\hat{x}_i, \perp_i)$, $x_i$ is a propositional variable mentioned in $\phi$ and $\sigma_z(x_i) = 0$
- $h(y) = y$ otherwise.

Let $J^*$ be the solution obtained from the canonical solution $J$ by replacing each element $y$ with $h(y)$. We show that $\text{certain}_M(Q, I) = \text{false}$. 

Assume for the sake of contradiction that there is a tuple in the evaluation of $Q$ over $J^*$. The witness of the relation $T$ in $Q$ cannot be a pair of the form $(\hat{x}_i, \bot_i)$, because then $h(\bot_i) = \bot_i$, contradicting the definition of $h$. The witness to the relation $T$ in $Q$ must then be the pair $(a, b)$. Then, there must exists a set of tuples $(\hat{s}_{11}, \hat{s}_{21}, \hat{s}_{31}, v_{11}, v_{21}, v_{31}, p_1, p_2), \ldots, (\hat{s}_{1\ell}, \hat{s}_{2\ell}, \hat{s}_{3\ell}, v_{1\ell}, v_{2\ell}, v_{3\ell}, p_\ell, p_{\ell+1})$ plus a series of tuples of the form $TV(\hat{x}_i, v_i)$, $TV(\hat{z}_j, v_j)$ for $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$ witnessing all of the relations $C$ and $TV$ in the query. Choose $\sigma_z$ such that for $j \in \{1, \ldots, m\}$, $\sigma_z(z_j) = 1$ if the pair $(\hat{z}_j, 1)$ witness the relation $TV$ in the query and $\sigma_z(z_j) = 0$ if the pair $(\hat{z}_j, 0)$ witness that relation. From the construction of the tuples in $C$, it is easy to see that $\sigma_x$ and $\sigma_z$ will satisfy all of the clauses in $\phi$, which is a contradiction, since $\phi$ is not satisfiable.

$(\Rightarrow)$ Assume that $\text{certain}_M(Q, I) = \text{false}$: there is a solution $J^*$ where the evaluation of $Q$ does not hold. It is clear that $J^*$ cannot have any unreplaced nulls of the type $\bot$; all of the $\bot_i$ must have been replaced with the values 0 or 1 for every $i \in \{1, \ldots, n\}$. Let $h$ be a homomorphism between the canonical solution $J$ and solution $J^*$. Choose the following valuation $\sigma_x$ for the universally quantified variables $x_i \in \{x_1, \ldots, x_n\}$:

- $\sigma_x(x_i) = 1$ if $h(\bot_i) = 1$ and there is a tuple $TV(\hat{x}_i, \bot_i)$ in $J$.
- $\sigma_x(x_i) = 0$ if $h(\bot_i) = 0$ and there is a tuple $TV(\hat{x}_i, \bot_i)$ in $J$.

Assume for the sake of contradiction that $\phi$ is satisfiable: for every valuation of the universally quantified variables there is a valuation for the existentially quantified variables that satisfy $\phi$. Then, in particular, for $\sigma_x$ there must exist a valuation $\sigma_z$ such that $\sigma_x$, $\sigma_z$ satisfy $\phi$. There are tuples $(\hat{s}_{11}, \hat{s}_{21}, \hat{s}_{31}, v_{11}, v_{21}, v_{31}, p_1, p_2), \ldots, (\hat{s}_{1\ell}, \hat{s}_{2\ell}, \hat{s}_{3\ell}, v_{1\ell}, v_{2\ell}, v_{3\ell}, p_\ell, p_{\ell+1})$ in $J^*$, and, since $\phi$ is satisfiable, it is clear from the construction of $C$ that if we choose the pairs $(\hat{z}_j, \sigma_z(z_j))$ and the pairs $(\hat{x}_i, h(\bot_i))$ as witness to the relations $TV$ for the query $Q$, then the evaluation of $Q$ over $J^*$ will be nonempty, and thus $\text{certain}_M(Q, I) = \text{true}$. This is again a contradiction, so we prove that $\phi$ can not be satisfiable.

We conclude this chapter with two remarks. First, notice that fixing data exchange settings has the same effect than restricting to LAV settings. In fact, the lower bounds in Proposition
6.2 remains the same for fixed LAV settings. Second, all the complexity bounds presented in this chapter remain the same if we allow unions of conjunctive queries with inequalities; if $k$-UCQ$\#$ is the class of unions of $k$-CQ$\#$ queries, then

**Proposition 6.3.**

(1) **Certain-Answers**($\text{GLAV}$, 1-UCQ$\#$) is Exptime-complete, and, for every $k \geq 2$, **Certain-Answers**($\text{GLAV}$, $k$-UCQ$\#$) is coNexptime-complete.

(2) **Certain-Answers**($\text{LAV}$, 1-UCQ$\#$) is NP-complete, and, for every $k \geq 2$, the problem **Certain-Answers**($\text{LAV}$, $k$-UCQ$\#$) is $\Pi^p_2$-complete.
7. CONCLUSION AND FUTURE RESEARCH

7.1. General Remarks

In this work, we proposed the language $\text{DATALOG}^{C(\neq)}$ that extends $\text{DATALOG}$ with a restricted form of negation, and studied some of its fundamental properties. In particular, we showed that the certain answers to a $\text{DATALOG}^{C(\neq)}$ 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. We also studied the combined complexity of computing certain answers to $\text{DATALOG}^{C(\neq)}$ programs and other related query languages.

7.2. Future Research Topics

We have studied thoroughly the class of $\text{UCQ}^{\neq}$ queries, but there are many other fragments of $\text{FO}$ that may deserve a detailed study. Just as we have found tractable fragments for the class of $\text{UCQ}^{\neq}$, we plan to make extensive use of $\text{DATALOG}^{C(\neq)}$ programs as a tool to find new tractable classes of queries in data exchange. This knowledge would lead to a better characterization of the problem of query answering in data exchange.

As a part of our future work, we would like to resolve some unsettled fundamental properties of $\text{DATALOG}^{C(\neq)}$ programs. It would be interesting to know, for example, whether is it decidable if the certain answers to a query $Q$ in UCQ can be computed as the certain answers to a $\text{DATALOG}^{C(\neq)}$ program $\Pi_q$. It would also be important to establish an upper bound to the expressive power of $\text{DATALOG}^{C(\neq)}$ programs. In this respect, we would like to know whether there is a setting $\mathcal{M}$ and a query $Q$ in $\text{UCQ}^{\neq}$ such that the problem $\text{certain}(\mathcal{M}, Q)$ is in PTIME, but the certain answers to $Q$ cannot be computed as the certain answers to a $\text{DATALOG}^{C(\neq)}$ program.

Concerning the use of $\text{DATALOG}^{C(\neq)}$ as a query language for data exchange, there is also much to be done. For starters, we would like to see some implementation for this query language in real-life relational data exchange applications.
REFERENCES


APPENDIX A. ADDITIONAL PROOFS

Proof of Proposition 5.3

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

$$
P(x, y) \rightarrow \exists z (T(x, z) \land T(y, z))$$

$$
R(x, y, z) \rightarrow S(x, y, z)
$$

Furthermore, Boolean query $Q$ is defined as:

$$
\exists x_1 \exists y_1 \exists x_2 \exists y_2 \exists x_3 \exists y_3 (S(x_1, x_2, x_3) \land T(x_1, y_1) \land T(x_2, y_2) \land T(x_3, y_3) \land x_1 \neq y_1 \land x_2 \neq y_2 \land x_3 \neq y_3).
$$

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

Membership of $\text{CERTAIN-ANSWERS}(\mathcal{M}, Q)$ in $\text{CONP}$ follows from (Fagin, Kolaitis, Miller, & Popa, 2005). The $\text{CONP}$-hardness is established from a reduction from 3SAT to the complement of $\text{CERTAIN-ANSWERS}(\mathcal{M}, Q)$. More precisely, for every 3CNF propositional formula $\phi$, we construct in polynomial time an instance $I_\phi$ of $\mathcal{S}$ such that $\phi$ is satisfiable iff $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \text{false}$.

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

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

Clearly, $I_\phi$ can be constructed in polynomial time from $\phi$.

The canonical universal solution $J$ for $I_\phi$ is as follows, where we denote by $\bot_q$ (or $\bot_{\neg q}$) the null generated by applying the std $P(x, y) \rightarrow \exists z (T(x, z) \land T(y, z))$ 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 $\phi$; and
- the interpretation of the relation $S$ in $J$ is just a copy of the interpretation of the relation $R$ in $I_\phi$.

We prove now that $\phi$ is satisfiable iff $\text{certain}_M(Q, I_\phi) = \text{false}$.

($\Rightarrow$) Assume that $\phi$ is satisfiable, and let $\kappa$ be a truth assignment for the propositional variables of $\phi$ such that $\kappa(\phi) = 1$. From $\kappa$, define a function $f$ from $J$ into $J$ 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 $I_\phi$ obtained from $J$ by replacing each occurrence of an element $v$ in $J$ by $f(v)$. We show next that $Q(J^*) = \text{false}$, and, thus, that $\text{certain}_M(Q, I_\phi) = \text{false}$.

Assume, for the sake of contradiction, that $Q(J^*) = \text{true}$. 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 $h(x_1) \neq h(y_1), h(x_2) \neq h(y_2), h(x_3) \neq h(y_3)$. Then by definition of $M$ and $I_\phi$, there exists a clause $(\alpha \lor \beta \lor \gamma)$ in $\phi$ such that $h(x_1) = \alpha$, $h(x_2) = \beta$, and $h(x_3) = \gamma$. Since $L(h(x_1), h(y_1)) = L(\alpha, f(\bot_\alpha))$ belongs to $J^*$, and $\alpha = h(x_1) \neq h(y_1) = f(\bot_\alpha)$, 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, $\kappa(\alpha) = 1, \kappa(\beta) = 1, \text{or } \kappa(\gamma) = 1$.

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

Consider an arbitrary $j \in [1, m]$, and assume that $C_j = (\alpha \lor \beta \lor \gamma)$. 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 theorem. □

Proof of Theorem 5.2

The details for the first part of the proof are given in the body. The rest of the proof is as follows:

(2) We now 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 $\text{CERTAIN-ANSWERS}(\mathcal{M}, Q)$ is $\text{CONP}$-complete.

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

$$
\begin{align*}
M(x, y, z) & \rightarrow P(x, y, z) \\
N(x, y) & \rightarrow \exists z \exists u (R(x, z) \land R(y, u) \land S(x, u)) \\
U(x) & \rightarrow S(x, x)
\end{align*}
$$

The Boolean query $Q$ is as follows:

$$
\exists x_1 \exists y_1 \exists x_2 \exists y_2 \exists x_3 \exists y_3 (P(x_1, x_2, x_3) \land R(x_1, y_1) \land S(x_2, y_2) \land R(x_3, y_3) \land y_1 \neq y_2 \land y_2 \neq y_3).
$$
Clearly, \( Q \) has constant joins, but does not have constant inequalities in \( M \). We prove next that the problem \( \text{CERTAIN-ANSWERS}(M, Q) \) is \( \text{coNP} \)-complete.

Membership in \( \text{coNP} \) follows from (Fagin, Kolaitis, Miller, & Popa, 2005). The \( \text{coNP} \)-hardness is established from a reduction from \( \text{POSITIVE-NOT-ALL-EQUAL-3SAT} \), which is the following decision problem: Given a propositional formula \( \phi \) in 3CNF consisting entirely of positive clauses \( (p \lor q \lor r) \), is there a valuation to the propositional variables of \( \phi \) such that for every clause of \( \phi \) 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 (Fagin, Kolaitis, Miller, & Popa, 2005)). More precisely, for every 3CNF propositional formula \( \phi \) consisting entirely of positive clauses, we construct in polynomial time an instance \( I_\phi \) of \( S \) such that \( \phi \) is \( \text{NOT-ALL-EQUAL-satisfiable} \) iff \( \text{certain}_M(Q, I_\phi) = \text{false} \).

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 \( I_\phi \) be the following source instance, where 1 and 0 are constants not mentioned in \( \phi \):

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

Clearly, \( I_\phi \) can be constructed in polynomial from \( \phi \).

The canonical universal solution \( J \) of \( I_\phi \) is as follows, where we denote by \( \bot_q \) and \( \#_q \) the nulls that are generated in order to witness variables \( z \) and \( u \), respectively, when applying the std \( N(x, y) \rightarrow \exists z \exists u (R(x, z) \land R(y, u) \land S(x, u)) \) 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 \( I_\phi \);
• the interpretation of the relation $R$ in $J$ contains the pairs $(q, \perp_q)$ and $(\hat{q}, \#_q)$, for each propositional variable $q$ mentioned in $\phi$; and

• the interpretation of the relation $S$ in $J$ contains the pair $(q, \#_q)$, for each propositional variable $q$ mentioned in $\phi$, and also contains the pairs $(1, 1)$ and $(0, 0)$.

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

($\Rightarrow$) Assume first that $\phi$ is NOT-ALL-EQUAL-satisfiable, and let $\kappa$ be a truth assignment for the propositional variables mentioned in $\phi$, such that for every clause $(p \lor q \lor r)$ in $\phi$, 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 $\kappa$ we construct a function $f$ from $\text{dom}(J)$ into $\text{dom}(J)$ as follows:

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

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

Assume, for the sake of contradiction, that $Q(J^*) = \text{true}$. Then there is a function $h : \{x_1, x_2, x_3, y_1, y_2, y_3\} \rightarrow \text{dom}(J^*)$ such that $P(h(x_1), h(x_2), h(x_3))$, $R(h(x_1), h(y_1))$, $S(h(x_2), h(y_2))$, 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 $h(x_2)$:

(i) 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 $\phi$. Further, $h(y_1) = f(\perp_q)$,
Assume, on the other hand, that $h(y_2) = 1$, 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 $f$.

(ii) Second, $h(x_2) = 0$. This case is similar to the previous one.

(iii) Finally, $h(x_2) = q$, for some propositional variable $q$ mentioned in $\phi$. Then there is a clause $(p \lor q \lor r)$ in $\phi$ 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 $\kappa$ is NOT-ALL-EQUAL.

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

We show next that for each $1 \leq j \leq m$, if $C_j = (p \lor q \lor r)$ then $\kappa(C_j) = 1$, but it is not the case that $\kappa(p) = \kappa(q) = \kappa(r) = 1$. This will show that $\phi$ 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 $\phi$.

Assume first, for the sake of contradiction, that $h(\bot_q) = 0$ and $h(\#q) = 0$, for some propositional variable $q$ mentioned in $\phi$. Consider the function $f : \{x_1, y_1, x_2, y_2, x_3, y_3\} \rightarrow \text{dom}(J')$, 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 $Q(J') = \text{true}$, 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 $\phi$.

Consider now an arbitrary $j \in [1, m]$, and assume that $C_j = (p \lor q \lor r)$. Consider the function $f : \{x_1, y_1, x_2, y_2, x_3, y_3\} \rightarrow \text{dom}(J')$, 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 \( Q(J') = \text{false} \), it must be the case that \( h(\perp_p) = h(\#_q) \) or \( h(\#_q) = h(\perp_r) \). 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.

(3) Third, we prove that there is a LA V 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 \( \text{CERTAIN-ANSWERS}(\mathcal{M}, Q) \) is \( \text{coNP} \)-complete.

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

The set \( \Sigma_{st} \) of source-to-target dependencies is:

\[
\begin{align*}
M(x,y) & \rightarrow \exists z(S(x,y) \land S(y,x) \land V(x,y,z) \land U(z,z) \land S(z,z)) \\
R(x,y,v,w) & \rightarrow \exists z(T(x,z) \land T(y,z) \land V(v,w,x,z) \land V(v,w,y,z)) \\
P(x,y,z) & \rightarrow U(x,y,z) \\
N(x,y) & \rightarrow T(x,y)
\end{align*}
\]

The Boolean query \( Q \) over \( T \) is as follows:

\[
\begin{align*}
\exists x \exists x' \exists y \exists y' \exists z \exists x_1 \exists y_1 \exists x_2 \exists y_2 (T(x_1, y_1) & \land T(x_2, y_2) \\
& \land U(x,y,z) \land S(x,x') \land S(y,y') \land S(z,z') \land \\
& V(x_1, x_2, x', y') \land V(x_1, x_2, y', y') \land V(x_1, x_2, z', y') \land x_1 \neq y_1 \land x_2 \neq y_2).
\end{align*}
\]

Clearly, \( Q \) has almost constant inequalities in \( \mathcal{M} \), but does not have constant joins in \( \mathcal{M} \). We prove next that \( \text{CERTAIN-ANSWERS}(\mathcal{M}, Q) \) is \( \text{coNP} \)-complete.
Membership in \textsc{coNP} follows from (Fagin, Kolaitis, Miller, & Popa, 2005). The \textsc{coNP}-hardness is established from a reduction from 3SAT to the complement of the problem studied, namely \textsc{Certain-Answers}(\mathcal{M}, Q). More precisely, for every 3CNF propositional formula $\phi$, we construct in polynomial time an instance $I_\phi$ of $S$ such that $\phi$ is satisfiable iff $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \text{false}$.

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

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

Clearly, $I_\phi$ can be constructed in polynomial time from $\phi$.

Let $\#_q$ be the null obtained from the application of the std $M(x, y) \rightarrow \exists z(S(x, y) \land S(y, x) \land V(x, y, z, z) \land U(z, z, z) \land S(z, z))$ 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 $R(x, y, v, w) \rightarrow \exists z(T(x, z) \land T(y, z) \land R(v, w, x, z) \land R(v, w, y, z))$ to the tuple $(q, \neg q, a, c)$. The canonical universal solution $J$ for $I_\phi$ 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 $\phi$;
- the interpretation of $T$ in $J$ contains a copy of the interpretation of $N$ in $I_\phi$ and the pairs $(q, \bot_q)$, $(\neg q, \bot_{\neg q})$, for each propositional variable $q$ mentioned in $\phi$;
- the interpretation of \( U \) in \( J \) contains a copy of the interpretation of \( P \) in \( I_\phi \) and the tuple \((\#_q, \#_q, \#_q)\), for each propositional variable \( q \) mentioned in \( \phi \); 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 \( \phi \).

We prove next that \( \text{certain}_M(Q, I_\phi) = \text{false} \) iff \( \phi \) is satisfiable.

\((\Leftarrow)\) Assume that \( \phi \) is satisfiable, and let \( \kappa \) be a truth assignment for the propositional variables mentioned in \( \phi \) such that \( \kappa(\phi) = 1 \). Define a function \( f \) from \( \text{dom}(J) \) into \( \text{dom}(J) \) 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 \( I_\phi \) obtained from \( J \) by replacing each occurrence of an element \( v \) in \( J \) by \( f(v) \). We show next that \( Q(J^*) = \text{false} \), and, thus, that \( \text{certain}_M(Q, I_\phi) = \text{false} \).

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

(i) The first case is that \( h(x_1) = q \) and \( h(x_2) = \neg q \), for some propositional variable \( q \) mentioned in \( \phi \). 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 \( f \).
(ii) The second case is that \( h(x_1) = a \) and \( h(x_2) = c \). But then \( h(x') = q \) or \( h(x') = \neg q \), for some propositional variable \( q \) mentioned in \( \phi \). Since \( S(h(x), h(x')) \) belongs to \( J^* \), it must be the case that for some clause \((\alpha \lor \beta \lor \gamma)\) in \( \phi \), \( h(x) = \alpha \), \( h(y) = \beta \) 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 \( \text{certain}_M(Q, I) = \text{false} \). Then there exists a solution \( J' \) for \( I_\phi \) such that \( Q(J') = \text{false} \). Let \( h \) be a homomorphism from \( J \) to \( J' \). Let us define a truth assignment \( \kappa \) for the propositional variables mentioned in \( \phi \) 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 \( \phi \) is satisfiable.

Let clause \( C_j \) be \((\alpha \lor \beta \lor \gamma) \) \((j \in [1, m])\). 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 \( f(x_1) = a, f(y_1) = b, f(x_2) = c, f(y_2) = d, f(x) = \alpha, f(x') = \neg \alpha, f(y) = \beta, f(y') = \neg \beta, f(z) = \gamma, f(z') = \neg \gamma \) 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 \( Q(J') = \text{true} \), which is a contradiction.

We prove second that for each propositional variable \( q \) mentioned in \( \phi \), \( h(\bot_q) = q \) 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 $Q(J') = \text{true}$, 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.

□

Proof of Proposition 6.2

Membership in EXPTIME can be proved as follows. The certain answers to each union of conjunctive queries for a source instance $I$, under a setting $\mathcal{M}$, can be computed in polynomial time in the size of $\text{CAN}(I)$ (Fagin, Kolaitis, Miller, & Popa, 2005). But as we mentioned before, the size of $\text{CAN}(I)$ is at most $|I|^{|\Sigma_{st}|}$. It follows that the certain answers to each union of conjunctive queries with respect to a source instance $I$, under a setting $\mathcal{M}$, can be computed in exponential time in the size of $I$.

The proof of the lower bound is a refinement of a proof shown in (Kolaitis et al., 2006), where the theorem was proved for a union of two CQ queries. The EXPTIME-hardness is established from a reduction from the Single Rule Datalog Problem (Gottlob & Papadimitriou, 2003), which is the following problem: given a DATALOG program $\Pi$ 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 $\Pi$ over an empty instance? That is, we ask whether $\bar{t} \in \Pi(\emptyset)$. We shall call these programs Single Rule Datalog Programs (srup). The combined complexity of this problem was shown to be EXPTIME-complete by Gottlob and Papadimitriou (2003).

As in (Kolaitis et al., 2006), let $\Pi$ 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 $\Pi$ over the empty instance if and only if $\bar{t} \in A^{\Pi(\emptyset)}$. 
The idea of the reduction in (Kolaitis et al., 2006) 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 CQ\# query. A second query is used to check whether $\bar{t} \in \Pi(\emptyset)$. The difficulty in this proof is to show that the bound remains the same, even for a single CQ\# query with one inequality.

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

- The source schema $S$ consists of four unary relations $T$, $V$, $F$, $S$ plus all the extensional database predicates $R_1, \ldots, R_m$, 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 arities in $\Pi$, 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 $T$ 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 $k + 3$.
- We create the source instance $I$ as follows:
  - The interpretation of the relations $R_1, \ldots, R_m$ and $A$ contains all the tuples corresponding to the facts.
  - 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 $\Pi$ 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 for the tuple $\bar{t}$. The relations with a 1 will be used for the simulation of the sirup rule, and the relations with a 2 will be used when checking whether $\bar{t} \in \Pi(\emptyset)$. 
The set $\Sigma_{st}$ of dependencies is defined as follows. We create a copy of the relation $T$ in $T'$:

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

We create a copy of the facts in the program such that they may be used when simulating the sirup rule:

$$F(z), T(y), R_1(x_1, \ldots, x_{l_1}) \rightarrow R'_1(x_1, \ldots, x_{l_1}, y, z, z)$$

$$\vdots$$

$$F(z), T(y), R_m(x_1, \ldots, x_{l_m}) \rightarrow R'_m(x_1, \ldots, x_{l_m}, y, z, z)$$

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 each fact, we populate the target with a series of tuples built using every possible constant value in the source:

$$S(z), T(y), V(x_1), \ldots, V(x_{l_1}) \rightarrow R'_1(x_1, \ldots, x_{l_1}, y, z, z)$$

$$\vdots$$

$$S(z), T(y), V(x_1), \ldots, V(x_{l_m}) \rightarrow R'_m(x_1, \ldots, x_{l_m}, y, z, z)$$

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 $A'$. Intuitively, this represents that every fact in $\Pi$ is already computed into our database. We also add to $A'$ every possible tuple that could be generated with the program. The value $c$ in the position $k + 2$ of the tuples in $A'$ represents a computed tuple, while a null value represent one that has not yet been computed:

$$F(z), S(w), T(y), A(x_1, \ldots, x_k) \rightarrow A'(x_1, \ldots, x_k, y, z, w) \quad \text{(A.1)}$$

$$F(z), S(w), V(x_1), \ldots, V(x_k) \rightarrow \exists n(A'(x_1, \ldots, x_k, n, z, w)) \quad \text{(A.2)}$$

The relations $F$ and $S$ 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 \( A' \) with the values of \( F \) and \( S \), and add again every possible tuple to the relation \( A' \), but with other constants:

\[
S(z), F(y), W(x_1, \ldots, x_k, u) \rightarrow A'(x_1, \ldots, x_k, u, z, y) \tag{A.3}
\]

\[
S(z), T(y), V(x_1), \ldots, V(x_k) \rightarrow A'(x_1, \ldots, x_k, y, z, y) \tag{A.4}
\]

- 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 \( A \). Then query \( Q \) is defined as follows:

\[
Q = \exists u \exists v \exists w \exists w_1 \cdots \exists w_n \exists x_1 \cdots \exists x_k \exists y \exists z \left( A'(x_1, \ldots, x_k, z, u, w) \land T'(y) \land A'(x_1, \ldots, x_k, y, w, v) \land z \neq y \land \bigwedge_{1 \leq i \leq n} Q'_i(\bar{x}_i, y, u, w_i) \right).
\]

Depending on the witness for the first relation \( A' \) in the query, \( Q \) can simulate the sirup rule, or check whether \( \bar{t} \in \Pi(\emptyset) \). We will simulate the sirup rule when the witness for the first symbol \( A' \) is a tuple of the form \((x_1, \ldots, x_k, z, 1, 2)\). In this case, if \( z \) is a null value, then the inequality \( z \neq y \) holds in \( Q \) because \( T'(y) \) forces \( y \) to be witnessed by the constant \( c \). Then to make the inequality false, we must set the null value to \( c \). Informally, this means that the query is simulating an application of the sirup rule to add the tuple \((x_1, \ldots, x_k)\) to \( \Pi(\emptyset) \). On the other hand, we will check whether \( \bar{t} \in \Pi(\emptyset) \) when the witness for the first symbol \( A' \) in \( Q \) is a tuple of the form \((x_1, \ldots, x_k, d, 2, 1)\). In this case, the witness for the second symbol \( A' \) in \( Q \) must be an already computed tuple, that is, a tuple of the form \((x_1, \ldots, x_k, c, 1, 2)\), from which we conclude that the inequality in \( Q \) holds. In fact, from this we will also conclude that \( \text{certain}_M(Q, I) = \text{true} \), as formally shown below.

Before proving that the reduction works properly, we describe the canonical universal solution for \( I \). 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 \( \text{dom}(I) \). 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 (A.1), 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 $\text{dom}(I)$, mapping (A.2) includes in $A'$ a tuple of the form $(\bar{b}, \bot, 1, 2)$, where $\bot$ is a fresh null value. We shall generically describe the null values added by (A.2) as $\bot$. Third, mapping (A.3) copies the relation $W$ and adds the constants $2, 1$ to each of the tuple in $W$. Finally, for every possible tuple $\bar{b}$ generated by using the elements in $\text{dom}(I)$, mapping (A.4) includes in $A'$ the tuple $(\bar{b}, c, 2, c)$.

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

$(\Rightarrow)$ If $\text{certain}_M(Q, I) = \text{true}$, then $Q$ holds in all the possible solutions for $I$ under $M$. 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 $M$.
- 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 $\bot$ in $\bar{t}_i$. Then $J_{i+1}$ is generated from $J_i$ by replacing $\bot$ 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 $M$, and given that $J_m$ is a solution for $I$ and $\text{certain}_M(Q, I) = \text{true}$, 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 $A(\bar{a}_i) \in T_{H_+}^{i+1}(\emptyset)$.

- $i = 0$: 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 $\Pi$, and if $Q'_j = A'$, then $A(\bar{b}_j)$ is a fact in $\Pi$. Therefore, by definition of $\bar{t}_0$, we conclude that $A(\bar{a}_0)$ can be deduced from the facts of $\Pi$ and, thus, $A(\bar{a}_0) \in T_{\Pi}^1(\emptyset)$.

- Inductive step: Assume that for every $q < i$, it holds that $A(\bar{a}_q) \in T_{\Pi}^{q+1}(\emptyset)$, 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 $\Pi$ 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 $\Pi$ and, therefore, $A(\bar{b}_j) \in T_{\Pi}^{q+1}(\emptyset)$. 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 $A(\bar{b}_j) \in T_{\Pi}^{q+1}(\emptyset)$. Therefore, we conclude that $A(\bar{a}_i) \in T_{\Pi}^{i+1}(\emptyset)$.

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

$(\Leftarrow)$ Assume that $\bar{t} \in \Pi(\emptyset)$ and, for the sake of contradiction, assume that $\text{certain}_M(Q, I) = \text{false}$. Moreover, let $\hat{Q}(u, v, w, w_1, \ldots, w_n, x_1, \ldots, x_k, y, z)$ be a query obtained by removing the existential quantifiers from $Q$:

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

We build 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 $M$, 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 $\bot$ in a tuple of $T_i$ by the constant $c$, if $\bot$ 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 $\text{certain}_M(Q, I) = \text{false}$, one can deduce that $Q(J_m) = \text{false}$. Let $\psi$ be the following dependency:

$$\forall u \forall v \forall w_1 \cdots \forall w_n \forall x_1 \cdots \forall x_k \forall y \forall z \left( (A'(x_1, \ldots, x_k, z, u, w) \land T'(y) \land A'(x_1, \ldots, x_k, y, w, v) \land \bigwedge_{1 \leq i \leq n} Q_i'(\bar{x}_i, y, u, w_i)) \rightarrow (z = y) \right).$$

It is easy to see that solution $J_m$ can be obtained from $J_0$ by repeatedly chasing with $\psi$. Thus, it follows from (Fagin, Kolaitis, Miller, & Popa, 2005) that $\text{certain}_M(Q, I) = \text{false}$ if and only if $Q(J_m) = \text{false}$ and, therefore, $Q(J_m) = \text{false}$.

We now show that the fact that $Q(J_m) = \text{false}$ leads to a contradiction. Consider the program evaluation sequence $T^0_\Pi(\emptyset), \ldots, T^{m-1}_\Pi(\emptyset)$. Next we show by induction on $i$ that if $A(\bar{a})$ holds in $T^i_\Pi(\emptyset)$, then tuple $A'(\bar{a}, c, 1, 2)$ holds in $J_{i+1}$.

- $i = 0$: Assume that $A(\bar{a})$ holds in $T^0_\Pi(\emptyset)$. Then $A(\bar{a})$ is a fact in $\Pi$ or it can be deduced from the facts of $\Pi$ by using the only rule in this program. Thus, by the definition of $\Pi$, it is easy to see that in both cases $A'(\bar{a}, c, 1, 2)$ holds in $J_1$, since every fact in $\Pi$ of the form $A(\bar{b})$ appears in $J_0$ as $A'(\bar{b}, c, 1, 2)$, and every fact in $\Pi$ of the form $R_p(\bar{b})$ appears in $J_0$ as $R'_p(\bar{b}, c, 1, 1)$.

- Inductive step: Assume that the property holds for every $j < i$ and that $A(\bar{a})$ holds in $T^i_\Pi(\emptyset)$. If $A'(\bar{a}, c, 1, 2)$ holds in $J_i$, then by definition of the sequence $J_0, \ldots, J_m$, we have that $A'(\bar{a}, c, 1, 2)$ holds in $J_{i+1}$. Thus, assume that $A'(\bar{a}, c, 1, 2)$ does not hold in $J_i$, and notice this implies that $A'(\bar{a}, \bot, 1, 2)$ holds in $J_i$, where $\bot$ is a null value, and that $A(\bar{a})$ does not hold in $T^{i-1}_\Pi(\emptyset)$ (otherwise by induction hypothesis we obtain that $A'(\bar{a}, c, 1, 2)$ holds in $J_i$). But then we have that $A(\bar{a})$
can be deduced from $T_{\Pi}^{i-1}(\emptyset)$ by using the only rule in $\Pi$. Thus, there exists an instantiation $A(\bar{a}) \leftarrow Q_1(\bar{a}_1), \ldots, Q_n(\bar{a}_n)$ of the rule of $\Pi$ such that $Q_1(\bar{a}_1), \ldots, Q_n(\bar{a}_n)$ belong to $T_{\Pi}^{i-1}(\emptyset)$. 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 $A'(\bar{a}, \bot, 1, 2)$ and $A'(\bar{a}, c, 1, 2)$ hold in $J_i$, we conclude that one of the tuples of $T_i$ has $\bot$ as a witness for the inequality of $\hat{Q}$. This implies that $A'(\bar{a}, c, 1, 2)$ holds in $J_{i+1}$ since $\bot$ is replaced by $c$ to obtain $J_{i+1}$ from $J_i$.

By the definitions of the sequence $J_0, \ldots, J_m$ and the data exchange setting $M$, it is straightforward to prove that $T_{\Pi}^{m-1}(\emptyset) = T_{\Pi}^{m}(\emptyset)$. Thus, given that $\bar{t} \in \Pi(\emptyset)$, we have that $A(\bar{t})$ holds in $T_{\Pi}^{m-1}(\emptyset)$. Therefore, by the property shown above, we have that $A'(\bar{t}, c, 1, 2)$ holds in $J_m$. But this implies that $Q(J_m) = \text{true}$ since $A'(\bar{t}, d, 2, 1)$ holds in $J_m$, $A'(\bar{b}, c, 2, c)$ holds in $J_m$ for every tuple $\bar{b}$ in $\text{dom}(I)^k$, and $R_t(\bar{b}_i, c, 2, 2)$ holds in $J_m$ for every tuple $\bar{b}_i$ in $\text{dom}(I)^i$ ($i \in \{1, \ldots, m\}$). But this contradicts our initial assumption.


\textbf{Proof of Proposition 6.3}

As we mentioned before, the problem with the previous query $Q$ is that it is a \textit{union} of conjunctive queries with at most two inequalities per disjunct. Fix a FO formula $\phi$ that belongs to the Bernays-Schofinkel class. Let $M = (S, T, \Sigma_{st})$, $Q$ and $I$ be the setting, query and instance constructed as previously presented. For $\phi$, we construct next a second data exchange setting, $M' = (S', T', \Sigma'_{st})$, an instance $I'$ of $S'$ and a conjunctive query $Q'$ with two inequalities, such that $\text{certain}_M(Q, I) = \text{certain}_{M'}(Q', I')$ (that is, $\phi$ is satisfiable iff $\text{certain}_{M'}(Q', I') = \text{false}$).

We will need some additional notation. Let again $\ell$ be the number of subformulas, 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 $\Theta$, that is, the number of atomic subformulas of $\phi$. We will assume that $\Theta$ is ordered, and we will use a function $\tau : \Theta \rightarrow \{1, \ldots, |\Theta|\}$ such that $\tau(j) = m$ if $j$ is the $m$-th element of $\Theta$. 
Now we show the data exchange setting $\mathcal{M}'$:

- The source schema $\mathcal{S}'$ consists of six unary relations $D, E, B, C, O$ 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 $\phi$), a binary relation $G$, a relation $Z$ of arity 4, and a relation $A$ of arity equal to $|\Theta| + 5$.

- The target schema $\mathcal{T}'$ consists of a binary relation $G'$, one 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 $\phi$), and a set of binary relations \{$_{V'_1, \ldots, V'_p}$\}.

- The instance $I'$ is as follows. The domain of $I'$ contains the elements $a_1, \ldots, a_p$, and elements $s_0, \ldots, s_\ell, s_f, c_1, \ldots, c_n, d,$ and 0. The interpretation of each symbol in $\mathcal{S}$ in $I'$ is as follows:
  - $B' = \{a_1, \ldots, a_p\}$.
  - $O' = \{0\}, U' = \{1\}$ and $D' = \{d\}$.
  - $E' = \{s_0, s_f\}$ and $C' = \{s_1, \ldots, s_\ell\}$.
  - $Q''_k = \{c_k\}$ for every $k \in [1, n]$.
  - $V''_i = \{a_i\}$ for every $i \in [1, p]$.
  - $G'' = \{(s_0, 0), (s_f, 0)\} \cup \{(s_i, 1), (s_i, 0) \mid i \in [1, \ell]\}$.
  - $A'$ is as follows:
    * It contains a tuple with only $d$'s, except for a $s_f$ in its first position and values $s_1$, 0 in the last two positions. For example, if the arity of $A$ is 8, we would create the following tuple: $A(s_f, d, d, d, d, s_1, 0)$.
    * For each relation $i \in [1, n]$, $A'$ 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 each subformula $S_j$ of $\psi$ of the form $R_i(\bar{z})$, $A'$ contains two tuples with only $d$’s, except for a $s_j$ in the first position, a $c_i$ in the $(\tau(j) + 2)th$
position and either a value 0 or a value 1 in the last position. For example, if if the arity of $A$ is 8, for $S_2 \equiv R_2$ and such that $\tau(2) = 1$, we would create the tuples: $A(s_2, d, c_2, d, d, d, 0)$ and $A(s_2, d, c_2, d, d, d, d, 1)$.

* Then for each subformula $S_j$, $j \not\in \Theta$, such that $S_j \equiv (S_g \land S_h)$ or $S_j \equiv (S_g \lor S_h)$, $A''$ contains two tuples, both with only $d$'s except for a $s_j$ in the first position, and elements $s_g$, $s_h$, and either a value 0 or a value 1 in the last three positions. Continuing with the example, if if the arity of $A$ is 7 and if $S_1 \equiv (S_2 \land S_3)$ then we create tuples $A(s_1, d, d, d, s_2, s_3, 0)$ and $A(s_1, d, d, d, s_2, s_3, 1)$.

* For each subformula $S_j$, $j \not\in \Theta$, such that $S_j \equiv \neg S_g$, $A''$ contains two tuples, both with only $d$'s except for a $s_j$ in the first position, and elements $s_g$ and either a value 0 or a value 1 in the last two positions. Continuing with the example, if if the arity of $A$ is 7 and if $S_1 \equiv \neg S_2$ then we create tuples $A(s_1, d, d, d, s_2, 0)$ and $A(s_1, d, d, d, s_2, 1)$.

– Finally, we construct $Z''$ as follows:

* The tuple $(s_f, 0, 0, 1)$ belongs to $Z''$.

* For each subformula $S_j$, $j \in \Theta$, the following tuples belong to $Z''$: $(s_j, 0, 0, 1)$ and $(s_j, 0, 0, 0)$.

* For each subformula $S_j$, $j \not\in \Theta$, such that $S_j \equiv S_g \lor S_h$, the following tuples belong to $Z''$: $(s_j, 1, 1, 1)$, $(s_j, 0, 1, 1)$, $(s_j, 1, 0, 1)$ and $(s_j, 0, 0, 0)$.

* For each subformula $S_j$, $j \not\in \Theta$, such that $S_j \equiv S_g \land S_h$, the following tuples belong to $Z''$: $(s_j, 1, 1, 1)$, $(s_j, 0, 1, 0)$, $(s_j, 1, 0, 0)$ and $(s_j, 0, 0, 0)$.

* For each subformula $S_j$, $j \not\in \Theta$, such that $S_j \equiv \neg S_g$, the following tuples belong to $Z''$: $(s_j, 0, 1, 0)$ and $(s_j, 0, 0, 1)$.

This finishes the definition of $I''$.

• The set $\Sigma'_{st}$ of source-to-target dependencies is as follows:
– We create a copy of $G$, $A$ and $Z$ into $G'$, $A'$ and $Z'$, respectively:

\[
G(x, y) \rightarrow G'(x, y) \quad (A.5)
\]

\[
A(\bar{x}) \rightarrow A'(\bar{x}) \quad (A.6)
\]

\[
Z(x, y, z, w) \rightarrow Z'(x, y, z, w) \quad (A.7)
\]

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

\[
V_1(x), C(y) \rightarrow V_1'(y, x) \quad (A.8)
\]

\[
\vdots
\]

\[
V_p(x), C(y) \rightarrow V_p'(y, x)
\]

– For each $i \in [1, p]$, we copy every pair of the form $(a_j, s_0)$ and $(a_j, s_f)$, $j \in [1, p]$, into $V_i'$:

\[
B(x), E(z) \rightarrow V_1'(x, z) \quad (A.9)
\]

\[
\vdots
\]

\[
B(x), E(z) \rightarrow V_p'(x, z)
\]

– Let $s = \max_{i \in [1,n]} r_i$. We add the following stds to $\Sigma_{st}$:

\[
Q_1(y), E(z), D(w), O(v),
\]

\[
B(x_1), \ldots, B(x_{r_1}), D(x_{r_1+1}), \ldots, D(x_s) \rightarrow \exists n(R'(y, x_1, \ldots, x_s, n),
\]

\[
R'(w, x_1, \ldots, x_1, n), Z'(z, v, v, n))
\]

\[
\vdots
\]

\[
Q_n(y), E(z), D(w), O(v),
\]

\[
B(x_1), \ldots, B(x_{r_n}), D(x_{r_n+1}), \ldots, D(x_s) \rightarrow \exists n(R'(y, x_1, \ldots, x_s, n),
\]

\[
R'(w, x_1, \ldots, x_1, n), Z'(z, v, v, n))
\]
The idea is the following. 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 \( \text{CAN}(I') \): \((s_i, a_{j_1}, \ldots, a_{j_{r_i}}, d, \ldots, d, \bot)\) and \((d, a_{j_1}, \ldots, a_{j_{r_i}}, \bot)\), where \( \bot \) is a fresh null value. In such case, we also add the tuples \((s_0, 0, 0, \bot)\) and \((s_f, 0, 0, \bot)\) to the interpretation of \( Z' \) in \( \text{CAN}(I') \).

- The following stds are also in \( \Sigma_{st} \):

\[
C(y), B(x_1), \ldots, B(x_m) \rightarrow \exists n F(y, x_1, \ldots, x_m, n) \tag{A.11}
\]

The idea is that the interpretation of \( F \) in \( \text{CAN}(I') \) contains for every \( j \in [1, \ell] \) and every tuple of elements \( a_{i_1}, \ldots, a_{i_m} \) in \( \{a_1, \ldots, a_p\} \), a tuple \((s_j, a_{i_1}, \ldots, a_{i_m}, \bot)\), where \( \bot \) is a fresh null value.

- The following are also in \( \Sigma_{st} \):

\[
D(y), O(z), B(x_1), \ldots, B(x_s) \rightarrow R'(y, x_1, \ldots, x_s, z) \tag{A.12}
\]

\[
D(y), U(z), B(x_1), \ldots, B(x_s) \rightarrow R'(y, x_1, \ldots, x_s, z) \tag{A.13}
\]

That is, every tuple of the form \((d, a_{i_1}, \ldots, a_{i_s}, 0)\) and \((d, a_{i_1}, \ldots, a_{i_s}, 1)\), where \( a_{i_1}, \ldots, a_{i_s} \) 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 stds to \( \Sigma_{st} \):

\[
D(y), O(z), B(x_1), \ldots, B(x_m) \rightarrow F(y, x_1, \ldots, x_m, z) \tag{A.14}
\]

\[
D(y), U(z), B(x_1), \ldots, B(x_m) \rightarrow F(y, x_1, \ldots, x_m, z) \tag{A.15}
\]

\[
E(y), O(z), B(x_1), \ldots, B(x_m) \rightarrow F(y, x_1, \ldots, x_m, z) \tag{A.16}
\]

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 \((s_0, a_{i_1}, \ldots, a_{i_m}, 0)\) 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 $\Sigma_{st}$.

Informally, we code in $\text{CAN}(I')$ the interpretation of each relation $R'_i$ in $\text{CAN}(I)$ using the symbol $R'$: $R'_i(\bar{a}, \bot)$ belongs to $\text{CAN}(I)$ iff $R'(c, \bar{a}, d, \ldots, d, \bot)$ belongs to $\text{CAN}(I')$. The interpretation of each $F_j$ in $\text{CAN}(I)$ is coded in the same way using $F'$: $F_j(\bar{b}, \bot)$ belongs to $\text{CAN}(I)$ iff $F(s, \bar{b}, \bot)$ belongs to $\text{CAN}(I')$. Finally, we use the relation $A'$ as a controller for the query, such that $Q'$ correctly simulates the queries $Q_\alpha, Q_\beta, Q_\gamma$ and $Q_\delta$ that where used in the first part of the reduction.

This is better explained when we describe the query.

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 $S_j = R_i(\bar{x}_i, \bar{y}_i)$. The query $Q$ is as follows:

$$Q' \equiv \exists x_1, \ldots, x_p \exists y_1, \ldots, y_m \forall t_0, t_1, \ldots, t_{|\Theta|}$$

$$\exists z_1, \ldots, z_{r_{\max}} \exists q \exists r \exists k, k' \exists u \exists w \exists w' \exists h_1, \ldots, h_{|\Theta|}$$

$$A'(q, t_0, t_1, \ldots, t_{i\kappa}, k, k', u) \land R'(t_0, z_1, \ldots, z_{r_{\max}}, v) \land$$

$$\bigwedge_{j \in \Theta} (R'(t_{\tau(j)}, \bar{x}_{\kappa(j)}, \bar{y}_{\kappa(j)}, \bar{h}_{\tau(j)}, v)) \land \bigwedge_{i \in [1, p]} V'_i(q, x_i) \land$$

$$F'(q, y_1, \ldots, y_m, n) \land F'(k, y_1, \ldots, y_m, w) \land F'(k', y_1, \ldots, y_m, w') \land$$

$$Z'(q, w, w', v) \land G(q, w') \land n \neq v \land u \neq v$$

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

All that remains to show is that $\text{certain}_M(Q, I) = \text{false}$ iff $\text{certain}_M'(Q', I') = \text{false}$. We only prove the direction from right to left, the other direction being analogous.

$(\Leftarrow)$ Let $\hat{J}'$ be the a solution for which $Q'(\hat{J}') = \text{false}$. Build the following function $h : \text{CAN}(I) \rightarrow \text{CAN}(I)$:
\[ h(n) = 1 \text{ if } n \text{ is a null value in a tuple of the form } (b_1, \ldots, b_{r_i}, n) \in R'_i, \quad i \in [1, n] \] and there is a tuple containing \((c_i, b_1, \ldots, b_{r_i})\) in its first \(r_i + 1\) positions and a 1 in the last position in \(R'_i\) in \(J'\).

\[ h(n) = 0 \text{ if } n \text{ is a null value in a tuple of the form } (b_1, \ldots, b_{r_i}, n) \in R'_i, \quad i \in [1, n] \] and there is a tuple containing \((c_i, b_1, \ldots, b_{r_i})\) in its first \(r_i + 1\) positions and a 0 in the last position in \(R'_i\) in \(J'\).

\[ h(n) = 1 \text{ if } n \text{ is a null value in a tuple of the form } (b_1, \ldots, b_{m}, n) \in F'_j, \quad j \in [1, \ell] \] and there is a tuple of the form \((s_j, b_1, \ldots, b_{m}, 1)\) in \(F'_j\) in \(J'\).

\[ h(n) = 0 \text{ if } n \text{ is a null value in a tuple of the form } (b_1, \ldots, b_{m}, n) \in F'_j, \quad j \in [1, \ell] \] and there is a tuple of the form \((s_j, b_1, \ldots, b_{m}, 0)\) in \(F'_j\) in \(J'\).

\[ h(n) = n \text{ otherwise.} \]

We prove that \(Q(h(CAN(I))) = \text{false}\). Assume for the sake of contradiction that \(Q_{\alpha}(h(CAN(I))) = \text{true}\). From the definition of \(Q_{\alpha}\), then there is an \(i \in [1, n]\), a tuple \((b_1, \ldots, b_{r_i})\) and an element \(v, v \neq 0, v \neq 1\) such that the tuple \((b_1, \ldots, b_{r_i}, v)\) belongs to the interpretation of \(R'_i\) in \(h(CAN(I))\). Let \(g\) be an homomorphism such that \(g(CAN(I')) = J'\).

We let \(\perp\) represent a null value such that \(R'(c_i, b_1, \ldots, b_{r_i}, \ldots, g(\perp))\) belongs to \(J'\). It then follows from the construction of the function \(h\) that \(g(\perp) \neq 0\) and \(g(\perp) \neq 1\). In order to show that \(Q(J') = \text{true}\), we build a function \(f : \{ x_1, \ldots, x_p, y_1, \ldots, y_m, t_0, t_1, \ldots, t_{|\Theta|}, z_1, \ldots, z_{r_{\max}}, q, r, k, k', u, v, w, w', h_1, \ldots, h_{|\Theta|}\} \rightarrow \text{dom}(J'),\) and then show that the tuples \(A(f(q), f(t_0), f(t_1), \ldots, f(t_{|\Theta|}), f(k), f(k'), f(u)), R'(f(t_0), f(z_1), \ldots, f(z_{r_{\max}}), f(v)), F'(f(q), f(y_1), \ldots, f(y_m), f(n)), F'(f(k'), f(y_1), \ldots, f(y_m), f(w')), Z'(f(q), f(w), f(w'), f(v)), G(f(q), f(w')); R'(f(t_{\tau(j)}), f(x_{\tau(j)}), f(g_{\tau(j)}), f(h_{\tau(j)}), f(v))\) for \(j \in \Theta\) and \(V_i'(f(q), f(x_i))\) for \(i \in [1, p]\) belong to \(J'\).

We define \(f\) as follows:

\[ f(\mu) = b_j \text{ if } \mu = z_j \text{ for } j \in [1, r_{\max}] \]

\[ f(\mu) = b_1 \text{ if } \mu = x_j \text{ for } j \in [1, p] \]

\[ f(\mu) = b_1 \text{ if } \mu = y_j \text{ for } j \in [1, m] \]
• \( f(\mu) = b_1 \) if \( \mu \in \bar{h}_j \) for \( j \in [1, |\Theta|] \)
• \( f(\mu) = d \) if \( \mu = k, \mu = k' \) or \( \mu = t_j \) for \( j \in [1, |\Theta|] \)
• \( f(t_0) = c_i, f(q) = s_0, f(w) = 1, f(n) = 0 \)
• \( f(w) = f(w') = 0 \)
• \( f(v) = g(\bot) \)

Note that \( f(v) \neq f(n) \) and \( f(v) \neq f(u) \). In ad \( R(f(t_0), f(z_1), \ldots, f(z_{\max}), f(v)) = R(c_i, b_1, \ldots, b_{r_{\lambda}}, \ldots, g(\bot)). \) This leads to a contradiction: it follows that \( Q'(\bar{J}') = \text{true} \) since the following tuples also belong to \( g(\text{CAN}(I')) = \bar{J}' \):

• \( A(s_0, c_i, d, \ldots, d, 1) \), obtained with (A.6).
• \( R'(d, b_1, \ldots, b_1, g(\bot)) \), obtained with (A.10).
• \( V_1(s_0, b_1), \ldots, V_p(s_0, b_1) \), obtained with (A.9).
• \( F(s_0, b_1, \ldots, b_1, 0) \), obtained with (A.16).
• \( F(d, b_1, \ldots, b_1, 0) \), obtained with (A.14).
• \( Z(s_0, 0, 0, g(\bot)) \), obtained with (A.10).
• \( G(s_0, 0) \), obtained with (A.5).

Next, assume for the sake of contradiction that \( Q(\text{h(\text{CAN}(I))}) = \text{true} \).

From the definition of \( Q_{\beta} \), then there are \( i, j, i \in [1, n] \) and \( j \in \Theta \) tuples \( (a_{i_1}, \ldots, a_{i_m}) \) and \( (\bar{a}_x, \bar{a}_y) \), and elements \( n \) and \( w, n \neq w \), such that subformula \( S_j \) is \( R_i(\bar{x}, \bar{y}) \), the tuple \( (a_{i_1}, \ldots, a_{i_m}, n) \) belongs to the interpretation of \( F' \) in \( \text{h(\text{CAN}(I))} \) and the tuple \( (\bar{a}_x, \bar{a}_y, w) \) belongs to the interpretation of \( R'_i \) in \( \text{h(\text{CAN}(I))} \). Since \( Q(\text{h(\text{CAN}(I))}) = \text{false} \), it is clear that \( w \) must be equal to 0 or to 1. We will show the case when \( w = 1 \), so that \( n \neq 1 \); the other case is analogous.

Since the tuple \( R_i(\bar{a}_x, \bar{a}_y, 1) \) belongs to \( \text{h(\text{CAN}(I))} \), there must be a tuple of the form \( R'(c_i, \bar{a}_x, \bar{a}_y, \ldots, 1) \) in \( \bar{J}' \), because otherwise it would contradict the definition of \( h \). Let \( g \) be an homomorism such that \( g(\text{CAN}(I')) = \bar{J}' \). In addition, it is easy to see that there is a null value \( \bot \) and a tuple \( F'(s_j, a_{i_1}, \ldots, a_{i_m}, \bot) \) in \( \text{CAN}(I') \). Again, by the definition of \( h \), we obtain that \( g(\bot) \neq 1 \).
In order to show that \( Q(\tilde{J}') = \text{true} \), we build again a function

\[
f : \{ x_1, \ldots, x_p, y_1, \ldots, y_m, t_0, t_1, \ldots, t_{|\Theta|}, z_1, \ldots, z_{r_{\text{max}}}, q, r, k, k', u, v, w, w', h_1, \ldots, h_{|\Theta|} \} \rightarrow \text{dom}(\tilde{J}')
\]

Such that the tuples \( A'(f(q), f(t_0), f(t_1), \ldots, f(t_{|\Theta|}), f(k), f(k'), f(u)), R'(f(t_0), f(z_1), \ldots, f(z_{r_{\text{max}}}), f(v)), F'(f(q), f(y_1), \ldots, f(y_m), f(n)), F'(f(q), f(y_1), \ldots, f(y_m), f(n)), F'(f(k'), f(y_1), \ldots, f(y_m), f(w')), Z'(f(q), f(w), f(w'), f(v)), G(f(q), f(w')), R'(f(t_\tau(j)), f(\bar{x}_\kappa(j)), f(\bar{y}_\kappa(j)), f(\bar{h}_\tau(j)), f(v)) \) for \( j \in \Theta \); and \( V'_i(f(q), f(x_i)) \) for \( i \in [1, p] \) belong to \( \tilde{J}' \).

We define \( f \) as follows:

- \( f(\mu) = a_i \) if \( \mu = y_k \) for \( k \in [1, m] \)
- \( f(\mu) = a_k \) if \( \mu = x_k \) for \( k \in [1, p] \)
- \( f(\mu) = a_1 \) if \( \mu = z_k \) for \( k \in [1, r_{\text{max}}] \)
- \( f(\mu) = a_1 \) if \( \mu \in \bar{h}_k \) for \( k \in [1, |\Theta|], k \neq \tau(j) \)
- \( f(\mu) = d \) if \( \mu \in \bar{h}_\tau(j) \)
- \( f(\mu) = d \) if \( \mu k, \mu = k' \) or \( \mu = t_k \) for \( k \in [1, |\Theta|] \) such that \( k \neq \tau(j) \)
- \( f(t_\tau(j)) = c_i \)
- \( f(t_0) = d, f(q) = s_j, f(u) = 1, f(v) = 1 \)
- \( f(w) = f(w') = 0 \)
- \( f(n) = g(\bot) \)

Note that \( f(v) \neq f(n) \) and \( f(v) \neq f(u) \). In addition, from the definition of \( f \) we obtain that \( F'(f(q), f(y_1), \ldots, f(y_m), f(n)) = F'(s_j, a_{i_1}, \ldots, a_{i_m}, g(\bot)) \). It is also the case that

\[
R'(f(t_\tau(j)), f(x_\kappa(j)_1), \ldots, f(x_\kappa(j)_1), f(y_\kappa(j)_1), \ldots, f(y_\kappa(j)_1), f(h_\tau(j)_1), \ldots, f(h_\tau(j)_1), f(v)) = R'(s_j, a_x, a_y, d, \ldots, d, 1).
\]

It follows that \( Q'(\tilde{J}') = \text{true} \) since the following tuples also belong to \( g(\text{CAN}(I')) = \tilde{J}' \):

- \( A(s_j, d, d, \ldots, d, t_\tau(j), d, \ldots, d, 1) \), obtained with (A.6).
- \( R'(d, a_1, \ldots, a_1, 1) \), obtained with (A.13).
- \( R'(d, f(y_\kappa(j)_1), \ldots, f(y_\kappa(j)_1), f(h_\tau(j)_1'), \ldots, f(h_\tau(j)_1'), 1)) \) for every \( j' \neq j \), obtained with (A.10).
\begin{itemize}
  \item $V_1(s_j, b_1), \ldots, V_p(s_j, b_p)$, obtained with (A.8).
  \item $F(s_j, a_{i_1}, \ldots, a_{i_m}, g(\bot_1))$, obtained with (A.11).
  \item $F(d_i, a_{i_1}, \ldots, a_{i_m}, 0)$, obtained with (A.14).
  \item $z(s_j, 0, 0, 1)$, obtained with (A.7).
  \item $G(s_j, 0)$, obtained with (A.5).
\end{itemize}

Next, assume that $Q_\gamma(h(\text{CAN}(I))) = \text{true}$. Let $k$ be the index such that the evaluation of $Q_k^g$ is true in $h(\text{CAN}(I))$. Assume without loss of generality that the subformula $S_k = \neg S_g$ for some $g \in [1, \ell]$. (The other two cases are completely symmetrical).

If $Q_k^g$ is true in $h(\text{CAN}(I))$, then there must be a tuple of elements $(a_{k_1}, \ldots, a_{k_m})$, such that $F'_k(a_{k_1}, \ldots, a_{k_m}, v_1)$ and $F'_g(a_{k_1}, \ldots, a_{k_m}, v_2)$, $E'(v_2, v_3)$ belong to $h(\text{CAN}(I))$, and $v_1 \neq v_3$. The only tuples that belong to the interpretation of $E'$ over CAN(I) are $(0, 1)$ and $(1, 0)$. We assume without loss of generality that $v_2 = 1$ (The case when $v_2 = 0$ is symmetrical). We obtain that $v_1 \neq 0$.

The proofs follows using the same argument as in the case of $Q_\alpha$ and $Q_\beta$: Notice that the only tuples in the interpretation of $F'_k$ and $F'_g$ in CAN(I) that contain $(a_{k_1}, \ldots, a_{k_m})$ are $F'_k(a_{k_1}, \ldots, a_{k_m}, \bot_1)$ and $F'_g(a_{k_1}, \ldots, a_{k_m}, \bot_2)$, where $\bot_1$ and $\bot_2$ represent null values. It must be the case then that $h(\bot_2) = 1$, and $h(\bot_1) \neq 0$.

From the definition of $h$, we obtain that there is a tuple $F'(s_g, a_{k_1}, \ldots, a_{k_m}, 1)$ in $\hat{J}'$. In addition, it is easy to see that CAN(I') contains a tuple of the form $F'(s_k, a_{k_1}, \ldots, a_{k_m}, \bot_3)$. Let $g$ be a homomorphism such that $g(\text{CAN}(I')) = \hat{J}'$. From the definition of $h$, we obtain that $g(\bot_3) \neq 0$.

We proceed in a similar fashion and define a function $f$ to show $Q'(\hat{J}') = \text{true}$.

The function $f$ is defined as follows:

\begin{itemize}
  \item $f(\mu) = a_{k_i}$ if $\mu = y_i$ for $i \in [1, m]$
  \item $f(\mu) = a_i$ if $\mu = x_i$ for $i \in [1, p]$
  \item $f(\mu) = a_1$ if $\mu = z_i$ for $i \in [1, r_{\text{max}}]$
  \item $f(\mu) = a_1$ if $\mu \in \bar{\Theta}$ for $i \in [1, |\Theta|]$
\end{itemize}
• \( f(\mu) = d \) if \( \mu = k \) or \( \mu = t_i \) for \( i \in [0, |\Theta|] \)
• \( f(q) = s_j, f(u) = 1, f(v) = 0 \)
• \( f(w) = 0, f(w') = 1 \)
• \( f(n) = g(\bot_3) \)

Note that \( f(n) \neq f(v) \) and \( f(u) \neq f(v) \). It follows that \( Q'(\hat{J}') = \text{true} \) since the following tuples also belong to \( g(\text{CAN}(I')) = \hat{J}' \):

• \( A(s_j, d, d, s_k, 0) \), obtained with \( (A.6) \).
• \( R'(d, a_{i1}, \ldots, a_{i1}, 0) \), obtained with \( (A.12) \).
• \( R'(d, f(y_{k(j')}), \ldots, f(y_{n(j')}), f(h_{\tau(j')}), \ldots, f(h_{\tau(j')}), 1) \) for every \( j' \in [1, |\Theta|] \), obtained with \( (A.10) \).
• \( V_1(s_j, b_1), \ldots, V_p(s_j, b_p) \), obtained with \( (A.8) \).
• \( F(s_k, a_{i1}, \ldots, a_{i1}, g(\bot_3)) \), obtained with \( (A.11) \).
• \( F(s_g, a_{i1}, \ldots, a_{i1}, 1) \), from the definition of \( h \).
• \( F(d, a_{i1}, \ldots, a_{i1}, 0) \), obtained with \( (A.14) \).
• \( z(s_j, 0, 1, 0) \), obtained with \( (A.7) \).
• \( G(s_j, 1) \), obtained with \( (A.5) \).

Finally, assume for the sake of contradiction that \( Q_\delta(h(\text{CAN}(I))) = \text{true} \). By the definition of \( Q_\delta \), there must be a tuple \( (a_{i1}, \ldots, a_{i1_m}) \) such that \( F'(a_{i1}, \ldots, a_{i1_m}, 0) \) belongs to \( h(\text{CAN}(I)) \). Then, by the definition of \( h \), there is a tuple \( F'(s_1, a_{i1}, \ldots, a_{i1_m}, 0) \) in \( \hat{J}' \).

We will use a function \( f \) again to prove that \( Q'(\hat{J}') = \text{false} \). It is defined as follows:

• \( f(\mu) = a_{1k} \) if \( \mu = y_k \) for \( k \in [1, m] \)
• \( f(\mu) = a_1 \) if \( \mu = x_k \) for \( k \in [1, p] \)
• \( f(\mu) = a_1 \) if \( \mu = z_k \) for \( k \in [1, r_{\text{max}}] \)
• \( f(\mu) = a_1 \) if \( \mu \in \bar{h}_k \) for \( k \in [1, |\Theta|] \)
• \( f(\mu) = d \) if \( \mu \neq k \) or \( \mu = t_k \) for \( k \in [0, |\Theta|] \)
• \( f(q) = s_f, f(u) = 0, f(v) = 1 \)
• \( f(w) = 0, f(w') = 0 \)
• $f(n) = 0$

Note that $f(n) \neq f(v)$ and $f(u) \neq f(v)$. It follows that $Q'(\hat{J}') = \text{true}$ since the following tuples also belong to $g(\text{CAN}(I')) = \hat{J}'$:

• $A(s_f, d, d, s_1, 0)$, obtained with (A.6).
• $R'(d, a_1, \ldots, a_1, 1)$, obtained with (A.13).
• $V_1(s_f, a_1), \ldots, V_p(s_f, a_1)$, obtained with (A.9).
• $F(s_1, a_{1_1}, \ldots, a_{1_m}, 0)$, from the definition of $h$.
• $F(d, a_{1_1}, \ldots, a_{1_m}, 0)$, obtained with (A.14).
• $z(s_f, 0, 0, 1)$, obtained with (A.7).
• $G(s_f, 0)$, obtained with (A.5).

Finally, since it must be the case that $Q_\alpha(h(\text{CAN}(I'))) = \text{false}$, $Q_\beta(h(\text{CAN}(I'))) = \text{false}$, $Q_\gamma(h(\text{CAN}(I'))) = \text{false}$ and $Q_\delta(h(\text{CAN}(I'))) = \text{false}$, we obtain that the evaluation of $Q$ over $h(\text{CAN}(I))$, and then $\text{certain}_\mathcal{M}(Q, I) = \text{false}$. \hfill \square