PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE

SCHOOL OF ENGINEERING

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

PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
SCHOOL OF ENGINEERING

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

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$^{\mathbf{C}(\neq)}$ Programs 12
   - 3.1. DATALOG$^{\mathbf{C}(\neq)}$ Programs 13

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

5. Conjunctive Queries with two Inequalities 29

6. Combined Complexity of Data Exchange 46
   - 6.1. Combined Complexity of DATALOG$^{\mathbf{C}(\neq)}$ 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 can be computed in polynomial time. The problem is that both UCQ and DATALOG do not allow negated atoms, as adding an unrestricted form of negation to these languages yields to intractability.

In this work, we propose a language called DATALOG$^{\mathbf{C}(\neq)}$ that extends DATALOG with a restricted form of negation, and study some of its fundamental properties. In particular, we show that the certain answers to a DATALOG$^{\mathbf{C}(\neq)}$ program can be computed in polynomial time (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 DATALOG$^{\mathbf{C}(\neq)}$ program in the context of data exchange. Furthermore, we show that this is also the case for a syntactic restriction of the class of unions of conjunctive queries with at most two inequalities per disjunct. This syntactic restriction is given by two conditions that are optimal, in the sense that computing certain answers becomes intractable if one removes any of them. Finally, we provide a thorough analysis of the combined complexity of computing certain answers to DATALOG$^{\mathbf{C}(\neq)}$ programs and other related query languages. In particular, we show that this problem is EXPTIME-complete for DATALOG$^{\mathbf{C}(\neq)}$, even if one restricts to conjunctive queries with single inequalities, which is a fragment of DATALOG$^{\mathbf{C}(\neq)}$ 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 seer 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$^{\mathbf{C}(\neq)}$, 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$^{\mathbf{C}(\neq)}$ 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$^{\mathbf{C}(\neq)}$. Ms aún, mostramos que este también es el caso para una restricción sintáctica de la clase de uniónes 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$^{\mathbf{C}(\neq)}$ y otros lenguajes de consulta relacionados. En particular, mostramos que este problema es EXPTIME-completo para DATALOG$^{\mathbf{C}(\neq)}$, incluso si se restringe a consultas conjuntivas con una desigualdad, que es un fragmento de DATALOG$^{\mathbf{C}(\neq)}$ 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 $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, where $\mathbf{S}$ is a *source* schema, $\mathbf{T}$ is a *target* schema, and Σ_{st} is a mapping defined as a set of *source-to-target* dependencies (Fagin et al, 2005) that specifies how to map data from $\mathbf{S}$ to $\mathbf{T}$. Given a source instance I, the goal in data exchange is to materialize a target instance J that is a *solution* for I, that is, J together with I must conform to the mapping Σ_{st}.

An important issue in data exchange is that the existing specification languages usually do not completely determine the relationship between source and target data and, thus, there may be many solutions for a given source instance. This immediately raises the question of which solution should be materialized. Initial work on data exchange (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 Σ_{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 $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ and a query Q over $\mathbf{T}$, a tuple $\bar{t}$ is said to be a certain answer to Q over I under $\mathcal{M}$, if $\bar{t}$ belongs to the evaluation of Q over every possible solution J for I under $\mathcal{M}$.

The definition of certain answers is highly non-effective, as it involves computing the intersection of (potentially) infinitely many sets. Thus, it becomes particularly important to understand for which classes of relevant queries, the certain answers can be computed efficiently. In particular, it becomes relevant to understand whether it is possible to compute the certain answers to any of these classes by using some materialized solution. Fagin, Kolaitis, Miller, and Popa (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 Σ_{st} and the query Q are assumed to be fixed.

The good properties of UCQ for data exchange can be completely explained by the fact that unions of conjunctive queries are preserved under homomorphisms. But this is not the only language that satisfies this condition, as queries definable in DATALOG, the recursive extension of UCQ, are also preserved under homomorphisms. Thus, DATALOG is as good as UCQ for data exchange purposes. In particular, the certain answers to a DATALOG program Π over a source instance I can also be computed efficiently by evaluating Π 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$^{\mathbf{C}(\neq)}$ that extends DATALOG with a restricted form of negation, and that has the same good properties for data exchange as DATALOG. In particular, we prove that the certain answers to a DATALOG$^{\mathbf{C}(\neq)}$ program Π over a source instance I can be computed by evaluating Π over the canonical universal solution for I. As a corollary, we obtain that computing certain answers to a DATALOG$^{\mathbf{C}(\neq)}$ program can be done in polynomial time (in terms of data complexity).
- To show that DATALOG$^{\mathbf{C}(\neq)}$ can be used to express interesting queries in the data exchange context, we prove that every union of conjunctive queries with at most one inequality or negated relational atom per disjunct, can be efficiently expressed as a DATALOG$^{\mathbf{C}(\neq)}$ program in the context of data exchange.
- It follows from the previous result that the certain answers to every union of conjunctive queries with at most one inequality or negated relational atom per disjunct can be computed in polynomial time (in terms of data complexity). Although this corollary is not new (it is a simple extension of a result in (Fagin, Kolaitis, Miller, & Popa, 2005)), the use of DATALOG$^{\mathbf{C}(\neq)}$ in the context of data exchange opens the possibility of finding new tractable classes of query languages with negation. In fact, we also use DATALOG$^{\mathbf{C}(\neq)}$ to find a tractable fragment of the class of conjunctive queries with two inequalities.

  It is known that for the class of conjunctive queries with inequalities, the problem of computing certain answers is CONP-complete 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$^{\mathbf{C}(\neq)}$ program in the context of data exchange. It immediately follows that the data complexity of computing certain answers to a query conforming to this restriction is polynomial.

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

The study of the complexity of computing certain answers to DATALOG$^{\mathbf{C}(\neq)}$ programs will not be complete if one does not consider the notion of *combined* complexity. Although the notion of data complexity has shown to be very useful in understanding the complexity of evaluating a query language, one should also study the complexity of this problem when none of its parameters is considered to be fixed. This corresponds to the notion of combined complexity introduced in (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 DATALOG$^{\mathbf{C}(\neq)}$ programs is EXPTIME-complete, even if one restricts to the class of conjunctive queries with single inequalities (which is a fragment of DATALOG$^{\mathbf{C}(\neq)}$ by the result mentioned above). This refines a result in (Kolaitis et al., 2006) that shows that the combined complexity of the problem of computing certain answers to *unions* of conjunctive queries with at most one inequality per disjunct is EXPTIME-complete.
- We also consider the class of conjunctive queries with an arbitrary number of inequalities per disjunct. More specifically, we show that the combined complexity of the problem of computing certain answers is coNEXPTIME-complete for the case of conjunctive queries with k inequalities, for every $k \geq 2$.
- One of the reasons for the high combined complexity of the previous problems is the fact that if data exchange settings are not considered to be fixed, then one has to deal with canonical universal solutions of exponential size. A natural way to

reduce the size of these solutions is to focus on the class of LAV data exchange settings (Lenzerini, 2002), which are frequently used in practice.
For the case of DATALOG$^{\mathbf{C}(\neq)}$ programs, the combined complexity is inherently exponential, and thus focusing on LAV settings does not reduce the complexity of computing certain answers. However, we show in 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 Π_2^p-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$^{\mathbf{C}(\neq)}$ programs. In chapter 4, we study some of the fundamental properties of DATALOG$^{\mathbf{C}(\neq)}$ 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$^{\mathbf{C}(\neq)}$ programs and other related query languages. Conclusions are in chapter 7.

2. BACKGROUND

2.1. Preliminaries

A *schema* $\mathbf{R}$ is a finite set $\{R_1, \ldots, R_k\}$ of relation symbols, with each R_i having a fixed arity $n_i > 0$. Let $\mathbf{D}$ be a countably infinite domain. An *instance* I of $\mathbf{R}$ assigns to each relation symbol R_i of $\mathbf{R}$ a finite n_i-ary relation $R_i^I \subseteq \mathbf{D}^{n_i}$. The *domain* 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, CQ$^{\neq}$ is the class of conjunctive queries with inequalities, and 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 $\mathbf{C}$ and $\mathbf{N}$ be infinite and disjoint sets of constants and nulls, respectively, and assume that $\mathbf{D} = \mathbf{C} \cup \mathbf{N}$. If we refer to a schema $\mathbf{S}$ as a *source* schema, then for every instance I of $\mathbf{S}$, it holds that dom$(I) \subseteq \mathbf{C}$. On the other hand, if we refer to a schema $\mathbf{T}$ as a *target* schema, then for every instance J of $\mathbf{T}$, it holds that dom$(J) \subseteq \mathbf{C} \cup \mathbf{N}$. Slightly abusing notation, we use $\mathbf{C}(\cdot)$ to denote a built-in unary predicate such that $\mathbf{C}(a)$ holds if and only if a is a constant, that is $a \in \mathbf{C}$.

A *data exchange setting* is a tuple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, where $\mathbf{S}$ is a source schema, $\mathbf{T}$ is a target schema, $\mathbf{S}$ and $\mathbf{T}$ do not have predicate symbols in common and Σ_{st} is a set of FO-dependencies over $\mathbf{S} \cup \mathbf{T}$ (in (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 Σ_{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 $\mathbf{S}$ and $\psi(\bar{x}, \bar{y})$ is a conjunction of relational atoms over $\mathbf{T}$. A *source* (resp. *target*) instance K for $\mathcal{M}$ is an instance of $\mathbf{S}$ (resp. $\mathbf{T}$). We usually denote source instances by $I, I', I_1, \ldots$, and target instances by $J, J', J_1, \ldots$.

The class of data exchange settings considered in this paper is usually called GLAV (global-&-local-as-view) in the database literature (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 $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, in which every std in Σ_{st} is of the form $\forall \bar{x}\, (S(\bar{x}) \rightarrow \psi(\bar{x}))$, for some $S \in \mathbf{S}$.

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

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

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 $\mathbf{R}$. A *homomorphism* h from K_1 to K_2 is a function $h : \text{dom}(K_1) \to \text{dom}(K_2)$ such that: (1) $h(c) = c$ for every $c \in \mathbf{C} \cap \text{dom}(K_1)$, and (2) for every $R \in \mathbf{R}$ and every tuple $\bar{a} = (a_1, \ldots, a_k) \in R^{K_1}$, it holds that $h(\bar{a}) = (h(a_1), \ldots, h(a_k)) \in R^{K_2}$. Notice that this definition of homomorphism slightly differs from the usual one, as the additional constraint that homomorphisms are the identity on the constants is imposed.

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

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

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 $(\mathbf{S}, \mathbf{T}, \Sigma_{st})$. For each std in Σ_{st} of the form:

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

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

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 $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, we denote by $\textsc{Can}$ the transformation from source instances to target instances, such that $\textsc{Can}(I)$ is the canonical universal solution for I under $\mathcal{M}$.

2.4. Certain answers

Queries in a data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ are posed over the target schema $\mathbf{T}$. Given that there may be (even infinitely) many solutions for a given source instance I with respect to $\mathcal{M}$, the standard approach in the data exchange literature is to define the semantics of the query based on the notion of certain answers (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 $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{true}$ iff Q holds for every J that is a solution for I under $\mathcal{M}$. We write $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$ if it is not the case that $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{true}$.

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

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

Since in the above definition both the setting and the query are fixed, it corresponds (in terms of Vardi's taxonomy (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$^{\mathbf{C}(\neq)}$ 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 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.[1]

Example 3.1. Consider a data exchange setting with $\mathbf{S} = \{E(\cdot,\cdot), A(\cdot), B(\cdot)\}$, $\mathbf{T} = \{G(\cdot,\cdot), P(\cdot), R(\cdot)\}$ and $\Sigma_{st} = \{E(x,y) \rightarrow G(x,y), A(x) \rightarrow P(x), B(x) \rightarrow R(x)\}$. Notice

[1]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 $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$ for every source instance I.

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

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 $\mathcal{M}$ are as in the previous example, then there is no union of conjunctive queries Q' such that $Q'(\text{CAN}(I)) = \text{certain}_{\mathcal{M}}(Q',I) = \text{certain}_{\mathcal{M}}(Q,I)$, for every source instance I. It immediately follows that negated relational atoms add expressive power to the class UCQ in the context of data exchange (see also (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$^{\mathbf{C}(\neq)}$ Programs

Unfortunately, adding an unrestricted form of negation to DATALOG (or even to CQ) not only destroys preservation under homomorphisms, but also easily yields to intractability of the problem of computing certain answers (Abiteboul & Duschka, 1999; Fagin, Kolaitis, Miller, & Popa, 2005). More precisely, there is a setting $\mathcal{M}$ and a query Q in UCQ$^{\neq}$ such that the problem CERTAIN-ANSWERS$(\mathcal{M}, Q)$ cannot be solved in polynomial time (unless PTIME $=$ NP). In particular, the set of certain answers of Q cannot be computed by evaluating Q over a polynomial-time computable universal solution. Next we show that there is a natural way of adding negation to DATALOG while keeping all of the good properties of this language for data exchange. In 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 (DATALOG$^{\mathbf{C}(\neq)}$ programs)**.** *A* constant-inequality Datalog rule *is a rule of the form:*

$$S(\bar{x}) \quad \leftarrow \quad S_1(\bar{x}_1), \ldots, S_\ell(\bar{x}_\ell), \mathbf{C}(y_1), \ldots, \mathbf{C}(y_m), u_1 \neq v_1, \ldots, u_n \neq v_n, \qquad (3.1)$$

where

(a) *S, S_1, …, S_ℓ 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 (DATALOG$^{\mathbf{C}(\neq)}$ *program*) Π *is a finite set of constant-inequality Datalog rules.*

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

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

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

Assume that Π is a DATALOG$^{\mathbf{C}(\neq)}$ program and I is a database instance of the relational schema $\mathit{Pred}(\Pi)$. Then $\mathcal{T}(I)$ is an instance of $\mathit{Pred}(\Pi)$ such that for every $R \in \mathit{Pred}(\Pi)$ and every tuple $\bar{t}$, it holds that $\bar{t} \in R^{\mathcal{T}(I)}$ if and only if there exists a rule $R(\bar{x}) \leftarrow R_1(\bar{x}_1), \ldots, R_\ell(\bar{x}_\ell), \mathbf{C}(y_1), \ldots, \mathbf{C}(y_m), u_1 \neq v_1, \ldots, u_n \neq v_n$ in Π and a variable assignment

σ such that (a) $\sigma(\bar{x}) = \bar{t}$, (b) $\sigma(\bar{x}_i) \in R_i^I$, for every $i \in [1, \ell]$, (c) $\sigma(y_i)$ is a constant, for every $i \in [1, m]$, and (d) $\sigma(u_i) \neq \sigma(v_i)$, for every $i \in [1, n]$. Operator $\mathcal{T}$ is used to define the semantics of constant-inequality Datalog programs. More precisely, define $\mathcal{T}_\Pi^0(I)$ to be I and $\mathcal{T}_\Pi^{n+1}(I)$ to be $\mathcal{T}(\mathcal{T}_\Pi^n(I)) \cup \mathcal{T}_\Pi^n(I)$, for every $n \geq 0$. Then the evaluation of Π over I is defined as $\mathcal{T}_\Pi^\infty(I) = \bigcup_{n \geq 0} \mathcal{T}_\Pi^n(I)$.

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

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

PROPOSITION 3.1. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a data exchange setting, I a source instance, J a universal solution for I under $\mathcal{M}$, and Π a* DATALOG$^{\mathbf{C}(\neq)}$ *program over $\mathbf{T}$. Then for every tuple $\bar{t}$ of constants, $\bar{t} \in \mathrm{certain}_{\mathcal{M}}(\Pi, I)$ iff $\bar{t} \in \Pi(J)$.*

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

4. COMPLEXITY AND EXPRESSIVENESS OF DATALOG$^{\mathbf{C}(\neq)}$ PROGRAMS

We start this chapter by studying the expressive power of DATALOG$^{\mathbf{C}(\neq)}$ programs. In particular, we show that these programs are expressive enough to capture the class of unions of conjunctive queries with at most one negated atom per disjunct. This class has proved to be relevant for data exchange, as its restriction with inequalities 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 $\mathbf{T}$, with at most one inequality or negated relational atom per disjunct. Then there exists a* DATALOG$^{\mathbf{C}(\neq)}$ *program Π_Q over $\mathbf{T}$ such that for every data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ and instance I of $\mathbf{S}$,* $\mathrm{certain}_{\mathcal{M}}(Q, I) = \mathrm{certain}_{\mathcal{M}}(\Pi_Q, I)$. *Moreover, Π_Q can be effectively constructed from Q in polynomial time.*

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

Example 4.1. Let $\mathcal{M}$ be a data exchange setting such that $\mathbf{S} = \{E(\cdot,\cdot), A(\cdot)\}$, $\mathbf{T} = \{G(\cdot,\cdot), P(\cdot)\}$ and

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

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

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

We build a DATALOG$^{\mathbf{C}(\neq)}$ program Π_Q such that $\mathrm{certain}_{\mathcal{M}}(Q, I) = \mathrm{certain}_{\mathcal{M}}(\Pi_Q, I)$. The set of intensional predicates of the DATALOG$^{\mathbf{C}(\neq)}$ program Π_Q is $\{U_1(\cdot,\cdot,\cdot)$, $U_2(\cdot,\cdot)$, $\mathrm{dom}(\cdot)$, EQUAL$(\cdot,\cdot,\cdot)$, ANSWER$(\cdot)\}$. The program Π_Q over $\mathbf{T}$ is defined as follows:

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

$$\text{dom}(x) \leftarrow G(x,z) \tag{4.1}$$

$$\text{dom}(x) \leftarrow G(z,x) \tag{4.2}$$

$$\text{dom}(x) \leftarrow P(x) \tag{4.3}$$

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

$$\text{E}\textsc{qual}(x,x,z) \leftarrow \text{dom}(x), \text{dom}(z) \tag{4.4}$$

$$\text{E}\textsc{qual}(x,y,z) \leftarrow \text{E}\textsc{qual}(y,x,z) \tag{4.5}$$

$$\text{E}\textsc{qual}(x,y,z) \leftarrow \text{E}\textsc{qual}(x,w,z), \text{E}\textsc{qual}(w,y,z) \tag{4.6}$$

  Predicate $\text{E}\textsc{qual}$ 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 $\text{E}\textsc{qual}(x,x,z) \leftarrow$ to say that $\text{E}\textsc{qual}$ is reflexive, as $\text{D}\textsc{atalog}^{\mathbf{C}(\neq)}$ programs are *safe*, i.e. every variable that appears in the head of a rule also has to appear in its body.

- Third, Π_Q includes the rules:

$$U_1(x,y,z) \leftarrow G(x,y), \text{dom}(z) \tag{4.7}$$

$$U_2(x,z) \leftarrow P(x), \text{dom}(z) \tag{4.8}$$

$$U_1(x,y,z) \leftarrow U_1(u,v,z), \text{E}\textsc{qual}(u,x,z), \text{E}\textsc{qual}(v,y,z) \tag{4.9}$$

$$U_2(x,z) \leftarrow U_2(u,z), \text{E}\textsc{qual}(u,x,z) \tag{4.10}$$

  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 Π_Q is evaluated. The last two rules allow to replace equal elements in the interpretation of U_1 and U_2.

- Fourth, Π_Q includes the following rule for the third disjunct of $Q(x)$:

$$U_1(x, y, x) \quad \leftarrow \quad U_1(x, z, x), U_1(z, y, x) \tag{4.11}$$

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

$$\textsc{Equal}(x, y, x) \quad \leftarrow \quad U_1(x, y, x) \tag{4.12}$$

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

$$\textsc{Answer}(x) \leftarrow U_2(x, x), U_1(x, x, x), \mathbf{C}(x) \tag{4.13}$$

$$\textsc{Answer}(x) \leftarrow \textsc{Equal}(y, z, x), \mathbf{C}(y), \mathbf{C}(z), y \neq z \tag{4.14}$$

  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 Π_Q with parameter a, two distinct constants b and c are declared to be equal ($\textsc{Equal}(b, c, a)$ 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 $\textsc{Can}(I)$. Thus, we conclude by applying rule (4.7) that $U_1(a, n, a)$ and $U_1(n, b, a)$ hold in $\textsc{Can}(I)$ 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 Π_Q, for every source instance I. □

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

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

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

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

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

- Π_Q includes the following rules for predicate EQUAL:

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

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

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

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

$$\textsc{Answer}(\bar{x}) \leftarrow U_{p_1}(\bar{u}_1, \bar{x}), \ldots, U_{p_n}(\bar{u}_n, \bar{x}), \mathbf{C}(x_1), \ldots, \mathbf{C}(x_m) \qquad (4.15)$$

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

$j \in [1, n+1]$. In this case, program Π_Q includes the following rule:

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

This rule is well defined since $\exists \bar{u}\, (T_{p_1}(\bar{u}_1) \wedge \cdots \wedge T_{p_n}(\bar{u}_n) \wedge \neg T_{p_{n+1}}(\bar{u}_{n+1}))$ is a safe query. Finally, assume that $Q_i(\bar{x})$ contains an inequality. Then $Q_i(\bar{x})$ is equal to $\exists \bar{u}\, (T_{p_1}(\bar{u}_1) \wedge \cdots \wedge T_{p_n}(\bar{u}_n) \wedge v_1 \neq v_2)$, where $p_j \in [1, k]$ and every variable in $\bar{u}_j$ is mentioned in either $\bar{u}$ or $\bar{x}$, for every $j \in [1, n]$, and v_1, v_2 are mentioned in $\bar{u}$ or $\bar{x}$. In this case, program Π_Q includes the following rules:

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

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

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

Let $\bar{a}$ be a tuple of elements from the domain of a source instance I. Each predicate U_i in Π_Q is used as a copy of T_i but with m extra arguments that store tuple $\bar{a}$. These predicates are used when testing whether $\bar{a}$ is a certain answer for Q over I. More specifically, the rules of Π_Q try to construct from $\text{CAN}(I)$ a solution J for I such that $\bar{a} \notin Q(J)$. Thus, if in a solution J for I, it holds that $\bar{a} \in Q(J)$ because $\bar{a} \in Q_i(J)$, where $Q_i(\bar{x})$ is equal to $\exists \bar{u}\, (T_{p_1}(\bar{u}_1) \wedge \cdots \wedge T_{p_n}(\bar{u}_n) \wedge \neg T_{p_{n+1}}(\bar{u}_{n+1}))$, then Π_Q uses rule (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 $\exists \bar{u}\, (T_{p_1}(\bar{u}_1) \wedge \cdots \wedge T_{p_n}(\bar{u}_n) \wedge v_1 \neq v_2)$, then Π_Q uses rule (4.17) to create a new solution where the values assigned to v_1 and v_2 are equal (predicate EQUAL is used to store this fact). If v_1 or v_2 is assigned a null value, then it is possible to create a solution where the values assigned to these variables are the same. But this is not possible if both v_1 and v_2 are assigned constant values. In fact, it follows from (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 $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ and for every instance I of $\mathbf{S}$, it is the case that $\text{certain}_{\mathcal{M}}(Q, I) = \text{certain}_{\mathcal{M}}(\Pi_Q, I)$. This concludes the proof of the theorem. $\square$

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

It was shown in (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 $Q^\star$ such that for every source instance I, it holds that $\text{certain}_{\mathcal{M}}(Q, I) = Q^\star(\text{CAN}(I))$. Thus, given that a non-recursive DATALOG$^{\mathbf{C}(\neq)}$ 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$^{\mathbf{C}(\neq)}$ *program Π, 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$^{\mathbf{C}(\neq)}$ programs without inequalities that are not expressible in first-order logic.

PROPOSITION 4.2. *There exist a data exchange setting $\mathcal{M}$ and a Boolean conjunctive query Q with a single inequality such that for every* DATALOG$^{\mathbf{C}(\neq)}$ *program* Π *without inequalities,* $\mathrm{certain}_{\mathcal{M}}(Q, I) \neq \mathrm{certain}_{\mathcal{M}}(\Pi, I)$ *for some source instance* I.

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

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

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

For the sake of contradiction, assume that there exists a DATALOG$^{\mathbf{C}(\neq)}$ program Π_0 without inequalities such that for every source instance I $\mathrm{certain}_{\mathcal{M}}(Q, I) = \mathrm{certain}_{\mathcal{M}}(\Pi_0, I)$ holds, and let $I_1 = \{M(a, b)\}$ and $I_2 = \{M(c, c)\}$. It is not hard to see that $\mathrm{certain}_{\mathcal{M}}(Q, I_1) =$ `true` and $\mathrm{certain}_{\mathcal{M}}(Q, I_2) =$ `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 $\mathrm{certain}_{\mathcal{M}}(Q, I_1) =$ `true`, we have that $\Pi_0(J_1) =$ `true`. Let h be a function from $\mathrm{dom}(J_1)$ to $\mathrm{dom}(J_2)$ defined as $h(a) = h(b) = c$. Since Π_0 is a DATALOG$^{\mathbf{C}(\neq)}$ program without inequalities, it must be preserved under h (because h maps constants to constants, and maps the pair $(a, b) \in N^{J_1}$ into the pair $(h(a), h(b)) = (c, c) \in N^{J_2}$). We conclude that $\Pi_0(J_2) =$ `true`. Hence, given that J_2 is a universal solution for I_2, we conclude from Proposition 3.1 that $\mathrm{certain}_{\mathcal{M}}(\Pi_0, I_2) =$ `true`. But we assume that $\mathrm{certain}_{\mathcal{M}}(Q, I_2) = \mathrm{certain}_{\mathcal{M}}(\Pi_0, I_2)$ and, therefore, we obtain a contradiction since $\mathrm{certain}_{\mathcal{M}}(Q, I_2) =$ `false`. $\square$

Notice that as a corollary of Proposition 4.2 and Theorem 4.1, we obtain that DATALOG$^{\mathbf{C}(\neq)}$ programs are strictly more expressive than DATALOG$^{\mathbf{C}(\neq)}$ programs without inequalities.

We conclude this chapter by studying the complexity of the problem of computing certain answers to DATALOG$^{\mathbf{C}(\neq)}$ programs. It was shown in Proposition 3.1 that the certain answers of a DATALOG$^{\mathbf{C}(\neq)}$ program Π can be computed by directly posing Π over CAN(I). This implies that for each data exchange setting $\mathcal{M}$, the problem
CERTAIN-ANSWERS$(\mathcal{M}, \Pi)$ can be solved in polynomial time if Π is a DATALOG$^{\mathbf{C}(\neq)}$ program (since CAN(I) can be computed in polynomial time and Π has polynomial time data complexity).

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

From the previous proposition and Theorem 4.1, we conclude that the certain answers to a union of conjunctive queries with at most one negated atom per disjunct can also be computed in polynomial time. We note that this slightly generalizes one of the polynomial time results in (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 DATALOG$^{\mathbf{C}(\neq)}$ programs can also be used to express (some) unions of conjunctive queries with two inequalities per disjunct.

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

PROPOSITION 4.4. *There exist a* LAV *data exchange setting* $\mathcal{M}$ *and a Boolean conjunctive query* Q *with one inequality such that* CERTAIN-ANSWERS$(\mathcal{M}, Q)$ *is* PTIME-*complete.*

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

$$V(x) \rightarrow \exists y\, T(x, y) \tag{4.19}$$

$$S(x, y) \rightarrow T(x, y) \tag{4.20}$$

$$P(x, y, w, z) \rightarrow R(x, y, w, z) \tag{4.21}$$

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

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

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

Membership of $\textsc{Certain-Answers}(\mathcal{M}, Q)$ in PTIME follows from (Fagin, Kolaitis, Miller, & Popa, 2005). PTIME-hardness is established from a LOGSPACE reduction from Horn-3CNF to the complement of $\textsc{Certain-Answers}(\mathcal{M}, Q)$, 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 ϕ, we construct in logarithmic space an instance I_ϕ of $\mathbf{S}$ such that ϕ is satisfiable if and only if $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$.

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

- The interpretation of unary relation V in I_ϕ is the set of propositional variables mentioned in ϕ.

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

Clearly, I_ϕ can be constructed in logarithmic space from ϕ.
Next, we show that $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$ if and only if ϕ is satisfiable.

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

In the setting $\mathcal{M}$, the canonical universal solution $\text{CAN}(I_\phi)$ for I_ϕ is as follows. Assume that $\bot_q$ is the null generated by applying rule (4.20) to each atom $V(q)$ in I_ϕ. Then the interpretation of R in $\text{CAN}(I)$ is equal to the interpretation of P in I, and the interpretation of T in $\text{CAN}(I)$ contains tuples $(\texttt{b}, \texttt{b})$, $(\texttt{h}, \texttt{f})$ and $(q, \bot_q)$ for every propositional variable q mentioned in ϕ.

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

- Assume that $C_i = p$. Since $R(p, \texttt{b}, \texttt{b}, \texttt{b})$ belongs to J, and also $T(\texttt{b}, \texttt{b})$ belongs to J, it must be the case that $h(\bot_p) = p$ since $Q(J) = \texttt{false}$ 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 $C_i = p \vee \neg q \vee \neg r$ 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

interpretation of T in J. Thus, given that $R(p, q, r, \mathtt{b})$ and $T(\mathtt{b}, \mathtt{b})$ belong to J, it must be the case that $h(\bot_p) = p$ since $Q(J) = \mathtt{false}$ and $(p, \bot_p)$ belongs to the interpretation of T in $\textsc{Can}(I)$. We conclude that $\sigma(p) = 1$ and, hence, $\sigma(C_i) = 1$.

- Assume that $C_i = \neg p \vee \neg q \vee \neg r$. For the sake of contradiction, assume that $\sigma(p) = \sigma(q) = \sigma(r) = 1$. Then by definition of h, we have that $h(\bot_p) = p$, $h(\bot_q) = q$ 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 $R(\mathtt{h}, p, q, r)$, $T(\mathtt{h}, \mathtt{f})$ belong to J and $\mathtt{h} \neq \mathtt{f}$ holds, we conclude that $Q(J) = \mathtt{true}$, which contradicts our initial assumption. We conclude that $\sigma(p) = 0$ or $\sigma(q) = 0$ or $\sigma(r) = 0$, which implies that $\sigma(C_i) = 1$.

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

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

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

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

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

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

This concludes the proof of the proposition. □

It is worth mentioning that it follows from Proposition 3.1 in (Kolaitis et al., 2006) that there exist a data exchange setting $\mathcal{M}$ containing some *target* dependencies and a conjunctive query Q with one inequality such that the problem CERTAIN-ANSWERS$(\mathcal{M}, Q)$ 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 LAV setting $\mathcal{M}$ and a Boolean conjunctive query Q with two inequalities such that CERTAIN-ANSWERS$(\mathcal{M}, Q)$ is CONP-complete (Madry, 2005). Therefore, unless PTIME $=$ NP, Theorem 4.1 is no longer valid if we remove the restriction that every disjunct of Q must contain at most one inequality.

The intractability for conjunctive queries with two inequalities is tightly related with the use of null values when joining relations and checking inequalities. In this 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$^{\mathbf{C}(\neq)}$ program Π_Q such that $\text{certain}_{\mathcal{M}}(Q, I) = \text{certain}_{\mathcal{M}}(\Pi_Q, I)$ for every source instance I. It should be noticed that DATALOG$^{\mathbf{C}(\neq)}$ 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 $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a data exchange setting. Then for every n-ary relation symbol T in $\mathbf{T}$, we say that the i-th attribute of T $(1 \leq i \leq n)$ *can be nullified* under $\mathcal{M}$, if there is an std α in Σ_{st} such that the i-th attribute of T is existentially quantified in the right hand side of α. Notice that for each setting $\mathcal{M}$ and source instance I, if the i-th attribute of T cannot be nullified under $\mathcal{M}$, then for every tuple $(c_1, \ldots, c_n)$ that belongs to T in the canonical universal solution for I, it holds that c_i is a constant. Moreover, if Q is a UCQ$^{\neq}$ query over $\mathbf{T}$ and x is a variable in Q, then we say that x *can be nullified* under Q and $\mathcal{M}$, if x appears in Q as the i-th attribute of a target relation T, and the i-th attribute of T can be nullified under $\mathcal{M}$.

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

Let Q be a conjunctive query with two inequalities over a target schema $\mathbf{T}$. Assume that the quantifier free part of Q is of the form $\phi(x_1, \ldots, x_m) \wedge u_1 \neq v_1 \wedge u_2 \neq v_2$, where ϕ is a conjunction of relational atoms over $\mathbf{T}$ and u_1, v_1, u_2 and v_2 are all mentioned in the set of variables x_1, …, 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 $\mathcal{M}$, if u_1 or v_1 cannot be nullified under Q and $\mathcal{M}$, and u_2 or v_2 cannot be nullified under Q and $\mathcal{M}$. Intuitively, this means that to satisfy Q in the canonical universal solution of a source instance, one can only make comparisons of the form $c \neq \perp$ and $c \neq c'$, where c, c' are constants and $\perp$ is a null value. Moreover, we say that Q has *constant joins* under $\mathcal{M}$, if for every variable x that appears at least twice in ϕ, x cannot be nullified under Q and $\mathcal{M}$. Intuitively, this means that to satisfy Q in the canonical universal solution of a source instance, one can only use constant values when joining relations.

Example 5.1. Let $\mathcal{M}$ be a data exchange setting 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 $\mathcal{M}$. On the other hand, the second attribute of U can be nullified under $\mathcal{M}$.

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

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

Proof: Assume that $\mathbf{T} = \{T_1, \ldots, T_k\}$, where each T_i has arity $n_i > 0$, and that $Q(\bar{x}) = Q_1(\bar{x}) \vee \cdots \vee Q_\ell(\bar{x})$, where $\bar{x} = (x_1, \ldots, x_m)$ and each $Q_i(\bar{x})$ is 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$^{\mathbf{C}(\neq)}$ program Π_Q is

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

and the arity of each predicate is defined as follows:

- each U_i, for $i \in [1, k]$, has arity $n_i + m$;

- $\textsc{Dom}$ has arity 1;
- $\textsc{Equal}$ has arity $2 + m$;
- each predicate of the form $\textsc{Equal}^{\textbf{left},j}$ or $\textsc{Equal}^{\textbf{right},j}$, for $j \in W$, has arity $2 + p_j + m$;
- $\textsc{Answer}$ has arity m; and
- each predicate of the form $\textsc{Answer}^{\textbf{left},j}$ or $\textsc{Answer}^{\textbf{right},j}$, for $j \in W$, has arity $p_j + m$.

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

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

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

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

- Π_Q includes the following rules for predicate $\textsc{Equal}$:

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

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

$$\begin{aligned}
\text{EQUAL}^{\textbf{left},j}(x, x, \bar{u}, \bar{z}) &\leftarrow \text{DOM}(x), \text{DOM}(\bar{u}), \text{DOM}(\bar{z}) \\
\text{EQUAL}^{\textbf{left},j}(x, y, \bar{u}, \bar{z}) &\leftarrow \text{EQUAL}^{\textbf{left},j}(y, x, \bar{u}, \bar{z}) \\
\text{EQUAL}^{\textbf{left},j}(x, y, \bar{u}, \bar{z}) &\leftarrow \text{EQUAL}^{\textbf{left},j}(x, w, \bar{u}, \bar{z}), \text{EQUAL}^{\textbf{left},j}(w, y, \bar{u}, \bar{z})
\end{aligned}$$

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

$$\begin{aligned}
\text{EQUAL}^{\textbf{right},j}(x, x, \bar{u}, \bar{z}) &\leftarrow \text{DOM}(x), \text{DOM}(\bar{u}), \text{DOM}(\bar{z}) \\
\text{EQUAL}^{\textbf{right},j}(x, y, \bar{u}, \bar{z}) &\leftarrow \text{EQUAL}^{\textbf{right},j}(y, x, \bar{u}, \bar{z}) \\
\text{EQUAL}^{\textbf{right},j}(x, y, \bar{u}, \bar{z}) &\leftarrow \text{EQUAL}^{\textbf{right},j}(x, w, \bar{u}, \bar{z}), \text{EQUAL}^{\textbf{right},j}(w, y, \bar{u}, \bar{z})
\end{aligned}$$

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

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

- 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 $\exists \bar{u}\, (T_{s_1}(\bar{u}_1) \wedge \cdots \wedge T_{s_n}(\bar{u}_n))$, where $s_j \in [1, k]$ and every variable in $\bar{u}_j$ is mentioned in either $\bar{u}$ or $\bar{x}$, for every $j \in [1, n]$. In this case, program Π_Q includes the following rule:

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

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

  Second, assume that $Q_i(\bar{x})$ contains an inequality. Then $Q_i(\bar{x})$ is equal to the formula $\exists \bar{u}\, (T_{s_1}(\bar{u}_1) \wedge \cdots \wedge T_{s_n}(\bar{u}_n) \wedge v_1 \neq v_2)$, where $s_i \in [1, k]$ and every variable in $\bar{u}_i$ is mentioned in either $\bar{u}$ or $\bar{x}$, for every $i \in [1, n]$, and v_1, v_2 are mentioned in

$\bar{u}$ or $\bar{x}$. In this case, program Π_Q includes the following rules:

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

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

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

We note that the rules above are also well defined since $\exists \bar{u}\,(T_{s_1}(\bar{u}_1) \wedge \cdots \wedge T_{s_n}(\bar{u}_n) \wedge v_1 \neq v_2)$ is a safe query.

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

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

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

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

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

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

It can be proved that for every data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ and instance I of $\mathbf{S}$, $\text{certain}_{\mathcal{M}}(Q, I) = \text{certain}_{\mathcal{M}}(\Pi_Q, I)$. This concludes the proof of the theorem. $\square$

It immediately follows from Proposition 4.3 that if a data exchange setting $\mathcal{M}$ and a UCQ$^{\neq}$ query Q satisfy the conditions mentioned in Theorem 5.1, then the problem denoted by CERTAIN-ANSWERS$(\mathcal{M}, Q)$ is in PTIME. Furthermore, it can also be shown that the properties of having constant joins and almost constant inequalities are helpful in reducing the complexity of computing certain answers to unions of conjunctive queries with at most one inequality per disjunct.

PROPOSITION 5.1. *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 $\mathcal{M}$, then* CERTAIN-ANSWERS$(\mathcal{M}, Q)$ *is in* NLOGSPACE*, and if in addition every disjunct of Q has almost constant inequalities under $\mathcal{M}$, then* CERTAIN-ANSWERS$(\mathcal{M}, Q)$ *is in* LOGSPACE.

Proof: Before proving the proposition, we mention a couple of remarks that will be useful in the proof. First, it is immediate from the definition of canonical universal solution that 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 CAN(I) for an arbitrary source instance I under $\mathcal{M}$, and the i-th attribute of T $(1 \leq i \leq n)$ is not existentially quantified in $\mathcal{M}$, then p_i has to be a constant.

We now prove the proposition, and start with part (1). Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a data exchange setting and Q a query that is the union of conjunctive queries, with at most one inequality per disjunct and without negated relational atoms, and such that each disjunct of Q has constant joins. We prove next that there exists a query Q', such that the data complexity of Q' is in NLOGSPACE, and $\text{certain}_{\mathcal{M}}(Q, I) = Q'(\text{CAN}(I))$, for every source instance I. It will immediately follow that CERTAIN-ANSWERS$(\mathcal{M}, Q)$ is in NLOGSPACE (the algorithm first computes CAN(I) in LOGSPACE, and then evaluates Q' over 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 $Q_1(\bar{x}) \vee \cdots \vee Q_\ell(\bar{x})$, where $\bar{x} = (x_1, \ldots, x_m)$, $m \geq 0$. Let I be an arbitrary source instance and $\bar{t} = (t_1, \ldots, t_m)$ a tuple of constants from I. We construct an undirected graph $G(Q, I, \bar{t})$ as follows:

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

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 $c \in \text{dom}(\textsc{Can}(I))$ to a different constant $c' \in \text{dom}(\textsc{Can}(I))$, or an edge between μ and ν. Next claim shows that checking whether a tuple $\bar{t}$ of constants from $\textsc{Can}(I)$ 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*

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

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 $c \in \text{dom}(\textsc{Can}(I))$ to some different constant

$c' \in \mathrm{dom}(\textsc{Can}(I))$, and there is no edge in $G(Q, I, \bar{t})$ between μ and ν. Thus, each connected component of $G(Q, I, \bar{t})$ with elements in $\mathrm{dom}(\textsc{Can}(I))$ consists of at most one constant and zero or more null values. With each connected component M with elements in $\mathrm{dom}(\textsc{Can}(I))$, we identify an element $id(M) \in M$ as follows: $id(M) = c$, if M contains a constant c, and $id(M) = n$ for an arbitrary null value $n \in M$, otherwise. Let J^* be the solution obtained from $\textsc{Can}(I)$ by replacing each occurrence of an element $p \in \mathrm{dom}(\textsc{Can}(I))$ in connected component M with the element $id(M)$. It is not hard to see that every element in $\bar{t}$ belongs to J^*. We prove that $J^* \not\models Q(\bar{t})$, and, thus, that $\mathrm{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$.

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 $\exists \bar{y}_1, \ldots, \bar{y}_n (T_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge T_n(\bar{x}_n, \bar{y}_n) \wedge u \neq v)$, where $\{T_1, \ldots, T_n\} \subseteq \mathbf{T}$, $\bar{x} = \{\bar{x}_1\} \cup \cdots \cup \{\bar{x}_n\}$, 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 $\textsc{Can}(I)$ and an assignment $\sigma : \{\bar{x}\} \cup \{\bar{y}_1\} \cup \cdots \cup \{\bar{y}_n\} \rightarrow \mathrm{dom}(J^*)$ defined by $\sigma(\bar{x}) = \bar{t}$ and $\sigma(\bar{y}_j) = \bar{p}_j$, for every $1 \leq j \leq n$, such that $(J^*, \sigma) \models T_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge T_n(\bar{x}_n, \bar{y}_n) \wedge u \neq v$. 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 $T_j(\bar{t}_j, \bar{r}_j)$ in J by replacing each element $r \in \bar{r}_j$ in the connected component M of $G(Q, I, \bar{t})$ by $id(M)$. Let us define an assignment $\sigma' : \{\bar{x}\} \cup \{\bar{y}_1\} \cup \cdots \cup \{\bar{y}_n\} \rightarrow \mathrm{dom}(\textsc{Can}(I))$ 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 σ' is well-defined. Assume that z is a variable that appears in at least two different positions in $(\bar{y}_1, \ldots, \bar{y}_n)$. We show that σ' 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 $T_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge T_n(\bar{x}_n, \bar{y}_n)$. 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 $\textsc{Can}(I)$ 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 σ'.

We conclude that $(\textsc{Can}(I), \sigma') \models T_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge T_n(\bar{x}_n, \bar{y}_n)$. 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 $\exists \bar{y}_1, \ldots, \bar{y}_n (T_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge T_n(\bar{x}_n, \bar{y}_n))$, where $\{T_1, \ldots, T_n\} \subseteq \mathbf{T}$ and $\bar{x} = \{\bar{x}_1\} \cup \cdots \cup \{\bar{x}_n\}$. Following the same reasoning it is possible to show that $(\text{CAN}(I), \sigma') \models T_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge T_n(\bar{x}_n, \bar{y}_n)$. Thus, $\text{CAN}(I) \models Q_i(\bar{t})$, which implies that there is an edge between μ and ν 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 μ and ν 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 $\exists \bar{y}(\phi(\bar{x}, \bar{y}) \wedge u \neq v)$, where $\phi(\bar{x}, \bar{y})$ is a conjunction of relational atoms over $\mathbf{T}$, and $u, v \in \{\bar{x}, \bar{y}\}$, and (2) there is an assignment $\sigma : \{\bar{x}, \bar{y}\} \to \text{dom}(\text{CAN}(I))$, such that $\sigma(\bar{x}) = \bar{t}$, $(\text{CAN}(I), \sigma) \models \phi(\bar{x}, \bar{y}) \wedge u \neq v$, $\sigma(u) = p$ and $\sigma(v) = p'$. Then $J^* \models \phi(h(\bar{t}), h(\sigma(\bar{y}))) \wedge h(\sigma(u)) \neq h(\sigma(v))$, because conjunctive queries are preserved under homomorphisms and $h(p) = h(\sigma(u)) \neq h(\sigma(v)) = h(p')$. Thus, $(J^*, \bar{t}) \models \exists \bar{y}(\phi(\bar{x}, \bar{y}) \wedge u \neq v)$, 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 $\exists \bar{y}_i(\phi_i(\bar{x}, \bar{y}_i) \wedge u_i \neq v_i)$, where $\phi_i(\bar{x}, \bar{y}_i)$ is a conjunction of relational atoms over $\mathbf{T}$, and $u_i, v_i \in \{\bar{x}, \bar{y}_i\}$, and for each $s < j \leq \ell$, $Q_j(\bar{x})$ is of the form $\exists \bar{y}_j \phi_j(\bar{x}, \bar{y}_j)$, where $\phi_j(\bar{x}, \bar{y}_j)$ is a conjunction of relational atoms over $\mathbf{T}$. Then:

(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 $Q(\bar{x})$:

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

Intuitively, the formula $A(z_1, z_2, \bar{x})$ defines the adjacency in the graph $G(Q, I, \bar{x})$, with respect to elements in $\textsc{Can}(I)$;

(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 μ and ν in $G(Q, I, \bar{x})$; and

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

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

where $(w_1, w_2) \in \mathrm{TrCl}.A(u, v, \bar{x})$ 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, $\mathrm{certain}_{\mathcal{M}}(Q, I) = Q'(\textsc{Can}(I))$. 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 μ and ν 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$^{\neq}$ 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* CERTAIN-ANSWERS$(\mathcal{M}, Q)$ *is* coNP*-complete.*

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

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

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

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

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

Furthermore, Boolean query Q is defined as:

$$\exists x \exists y \exists z (U(x,y,z) \wedge T(x,x) \wedge T(y,y) \wedge T(z,z)) \vee \\ \exists x \exists y \exists w \exists z (T(x,y) \wedge T(w,z) \wedge S(x,w) \wedge x \neq y \wedge w \neq z).$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

It is important to notice that although the problem of computing certain answers to UCQ$^{\neq}$ queries has been considered in the literature, none of the results of Theorem 5.2 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 coNP-complete even for the union of two queries, the first of which is a conjunctive query and the second of which is a conjunctive query with two inequalities. The difficulty in our case is that the second query is restricted to have constant joins and almost constant inequalities, while Fagin et al. considered a query that does not satisfy any of these conditions. Moreover, Mądry proved in (Madry, 2005) a similar result to (2) and (3), namely that the problem of computing certain answers is coNP-complete for conjunctive queries with two inequalities. The difficulty in our case is that we consider a query that has constant joins in (2) and a query that has almost constant inequalities in (3), while Mądry considered a query that does not satisfy any of these conditions. In fact, we provide in (2) and (3) two new proofs of the fact that the problem of computing certain answer to a conjunctive query with two inequalities is coNP-complete.

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

PROPOSITION 5.3. *There exist a* LAV *data exchange setting $\mathcal{M}$ and a Boolean conjunctive query Q with three inequalities, such that Q has both constant joins and almost constant inequalities under $\mathcal{M}$, but the problem* CERTAIN-ANSWERS$(\mathcal{M}, Q)$ *is* coNP*-complete.*

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

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

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$^{\mathbf{C}(\neq)}$ queries. The study continues by considering some restrictions of DATALOG$^{\mathbf{C}(\neq)}$ that lead to lower combined complexity, and which are expressed in the form of conjunctive queries with single inequalities. We conclude this study by examining unrestricted CQ$^{\neq}$ queries, which are not rewritable in DATALOG$^{\mathbf{C}(\neq)}$ (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$^{\mathbf{C}(\neq)}$ programs

We showed in Proposition 3.1 that the certain answers of a DATALOG$^{\mathbf{C}(\neq)}$ program can be computed by directly posing the query over the canonical universal solution. It can be shown that such an approach can compute the certain answers to a DATALOG$^{\mathbf{C}(\neq)}$ program in exponential time, although canonical universal solutions can be of exponential size if data exchange settings are not considered to be fixed. And not only that it can be proved that this is a tight bound.

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

Proof: The EXPTIME-hardness follows directly from Theorems 4.1 and 6.2: The problem of computing certain answers is already 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$^{\mathbf{C}(\neq)}$ program Π_Q such that $\text{certain}_{\mathcal{M}}(Q, I) = \text{certain}_{\mathcal{M}}(\Pi_Q, I)$, for every source instance I.

For membership in EXPTIME, in chapter 3 we showed that in order to compute the certain answers of a DATALOG$^{\mathbf{C}(\neq)}$ program Π for a source instance I, it suffices to evaluate Π over CAN(I). It is well known that every DATALOG program Π 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, 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$^{\mathbf{C}(\neq)}$ 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 LAV settings. However, for the case of DATALOG$^{\mathbf{C}(\neq)}$ programs, the combined complexity is inherently exponential, and thus reducing the size of canonical universal solutions does not help in improving the upper bound.

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

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

TABLE 6.1. Combined complexity of computing certain answers.

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 $\bar{t}$ belongs to the evaluation of a DATALOG program Π' over an instance I. This problem is well known to be EXPTIME-hard (see e.g. (Abiteboul et al., 1995), (Vardi, 1982)). It is clear that every DATALOG is also a DATALOG$^{\mathbf{C}(\neq)}$ program. Let $\mathcal{M}$ be a copying data exchange setting, that is, a (LAV) setting $(\mathbf{S}, \mathbf{T}, \Sigma_{st})$ such that $\mathbf{S}$ contains all the relation symbols in I, $\mathbf{T}$ is a copy of $\mathbf{S}$ and Σ_{st} consists of dependencies of the form $R(\bar{x}) \rightarrow R'(\bar{x})$ for each relation symbol R in $\mathbf{S}$, where R' is the copy in $\mathbf{T}$ of the relation R in $\mathbf{S}$. From the results in chapter 3 and the construction of CAN(I), it is straightforward to prove that $\bar{t} \in \Pi(I)$ if and only if $\bar{t} \in \text{certain}_{\mathcal{M}}(Q, I)$. □

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

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

Theorem 6.2. $\text{CERTAIN-ANSWERS}(\text{GLAV}, 1\text{-CQ}^{\neq})$ *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 $\mathcal{M}$, 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$^{\neq}$ 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 $\bar{t} \in \Pi(\emptyset)$. 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. $\square$

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. $\text{CERTAIN-ANSWERS}(\text{GLAV}, k\text{-CQ}^{\neq})$ *is* CONEXPTIME-*complete, for every* $k \geq 2$.

Proof: First, we prove the membership in CONEXPTIME. The certain answers to each k-UCQ$^{\neq}$ query for a source instance I, under a setting $\mathcal{M}$, can be computed in CONP time in the size of $\textsc{Can}(I)$ (Fagin, Kolaitis, Miller, & Popa, 2005). But as we mentioned before, the size of $\textsc{Can}(I)$ is at most $|I|^{|\Sigma_{st}|}$. It follows that the certain answers to each k-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 $\textsc{Certain-Answers}(\textsc{Glav}, 2\text{-CQ}^{\neq})$. 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 ψ is quantifier-free and does not contain function symbols. Along the proof we show the following: For every Boolean FO formula ϕ in the Bernays-Schönfinkel class, one can construct in polynomial time a GLAV data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, a conjunctive query Q with two inequalities, and an instance I of $\mathbf{S}$, such that ϕ is satisfiable iff $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$.

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 ϕ is constant-free and that ψ 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 ψ. For each relation R_i, $1 \leq i \leq n$, we let r_i denote the arity of R_i. Along the proof we heavily use the following property of ϕ: Either ϕ is unsatisfiable, or it has a model 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 ϕ a GLAV data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, a query Q that is a *union* of conjunctive queries with at most two inequalities per disjunct, and an instance I of $\mathbf{S}$, such that ϕ is satisfiable iff $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$. Although this is still not sufficient to prove the theorem, because Q belongs to 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 ϕ another GLAV data exchange setting $\mathcal{M}' = (\mathbf{S}', \mathbf{T}', \Sigma'_{st})$, a conjunctive query Q' with two inequalities, and an instance I' of $\mathbf{S}'$, such that ϕ is satisfiable iff $\text{certain}_{\mathcal{M}'}(Q', I') = \texttt{false}$.

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

We need some additional notation. A *subformula* of ψ is inductively defined as follows:

- If the atomic formula $R_i(\bar{z})$ appears in ψ, then $R_i(\bar{z})$ is a subformula of ψ;
- If ξ and γ are subformulas of ψ, then $(\neg\xi)$, $(\xi \vee \gamma)$, and $(\xi \wedge \gamma)$ are also subformulas of ψ.

Notice that in particular, ψ is a subformula of itself. Let $S_1, \ldots, S_\ell$ be an enumeration of all the subformulas of ψ. 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 dom(I) consists of p elements $a_1, \ldots, a_p$. This is justified by the fact, mentioned above, that if ϕ is satisfiable then it has a model of size at most p. We then construct a set Σ_{st} of stds such that for each tuple $\bar{a}$ of size r_i of elements in dom(I), $i \in [1, n]$, $R'_i(\bar{a}, \bot)$ belongs to $\text{C}\small{\text{AN}}(I)$, where $\bot$ is a fresh null value. The target schema $\mathbf{T}$ will contain one relation F_j of arity $m+1$ for each subformula S_j of ψ ($j \in [1, \ell]$), such that for each tuple $\bar{b}$ of size m in dom(I), $F_j(\bar{b}, \bot)$ belongs to $\text{C}\small{\text{AN}}(I)$, 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 $\mathcal{A}_J$ over the vocabulary $\{R_1, \ldots, R_k\}$ as follows: $\bar{a}$ belongs to the interpretation of the symbol R_i in $\mathcal{A}_J$ iff $R'_i(\bar{a}, 1) \in J$. Moreover, for $j \in [1, \ell]$, $F_j(\bar{b}, 1) \in J$ iff $\mathcal{A}_J$ satisfies subformula S_j when we respectively assign each element from $\bar{b}$ to variables $y_1, ..., y_m$ 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 $\mathbf{S}$ consists of three unary relations B, O and U, a set of unary relations $\{V_1, \ldots, V_p\}$ (recall that p is the number of existentially quantified variables in ϕ), two ternary relations C and D, and one binary relation E.
- The target schema $\mathbf{T}$ consists of a relation R'_i of arity r_i+1, for each $i \in [1, n]$, a set $\{V'_1, \ldots, V'_p\}$ of unary relations, two other unary relations O' and U', two ternary relations C' and D', a binary relation E', and an extra set of relations $\{F_1, \ldots, F_\ell\}$ each with arity $m + 1$ (recall that ℓ is the number of subformulas of ψ, and that m is the number of universally quantified variables of ϕ).
- The instance I is as follows. The domain of I contains the elements $a_1, ..., a_p$, plus two different constants not used elsewhere in the instance, 1 and 0. The interpretation of each symbol in $\mathbf{S}$ 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 Σ_{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:

$$\begin{array}{rcl} V_1(x) & \rightarrow & V_1'(x) \\ & \vdots & \\ V_p(x) & \rightarrow & V_p'(x) \\ O(x) & \rightarrow & O'(x) \\ U(x) & \rightarrow & U'(x) \\ C(x,y,z) & \rightarrow & C'(x,y,z) \\ D(x,y,z) & \rightarrow & D'(x,y,z) \\ E(x,y) & \rightarrow & E'(x,y) \end{array}$$

– 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:

$$\begin{array}{rcl} B(x_1), ..., B(x_{r_1}) & \rightarrow & \exists z R_1'(x_1, ..., x_{r_1}, z) \\ & \vdots & \\ B(x_1), ..., B(x_{r_n}) & \rightarrow & \exists z R_r'(x_1, ..., x_{r_n}, z) \end{array}$$

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:

$$\begin{aligned} B(x_1),...,B(x_m) &\rightarrow \exists z F_1(x_1,...,x_m,z) \\ &\vdots \\ B(x_1),...,B(x_m) &\rightarrow \exists z F_\ell(x_1,...,x_m,z) \end{aligned}$$

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

It is clear at this point what 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 $\perp$ such that the tuple $R_i(\bar{a},\perp)$ 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 $\perp$ such that the tuple $F_j(\bar{b},\perp)$ 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 $\mathcal{F}_1(\bar{c},0)$. 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 $Q_\alpha \vee Q_\beta \vee Q_\gamma \vee Q_\delta$, where

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

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

Note that if the evaluation of Q_α 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_β is defined as $\bigcup_{j \in \Theta} Q_\beta^j$, 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_β^j is as follows:

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

  Assume that S_j holds (resp., does not hold) in $\mathcal{A}_J$ when we assign elements $a_{i_1}, ..., a_{i_m}$ to variables $y_1, ..., y_m$ and elements $a_1, \ldots, a_p$ to variables $x_1, \ldots, x_p$, where J is a solution that falsifies Q_β^j. If $F_j(a_{i_1}, \ldots, a_{i_m}, \bot)$ belongs to $\textsc{Can}(I)$ then $\bot$ must have been replaced by 1 (resp. 0) in J.

- Q_γ is defined as $\bigcup_{k \notin \Theta} Q_\gamma^k$, where each query Q_γ^k is defined as follows:

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

$$Q_\gamma^k \equiv \exists y_1 \ldots \exists y_m \exists n \exists v \exists w \exists z \, (F_k(y_1, ..., y_m, n) \;\wedge\; F_g(y_1, ..., y_m, v) \wedge F_h(y_1, ..., y_m, w) \\ \wedge D'(v, w, z) \wedge n \neq z).$$

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

$$Q_\gamma^k \equiv \exists y_1 \ldots \exists y_m \exists n \exists v \exists w \exists z \, (F_k(y_1, ..., y_m, n) \;\wedge\; F_g(y_1, ..., y_m, v) \wedge F_h(y_1, ..., y_m, w) \\ \wedge C'(v, w, z) \wedge n \neq z).$$

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

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

  The purpose of this query is similar to Q_β, but here we ensure the correct interpretation of subformulas of ψ that are Boolean combinations of other subformulas. Recall that the tuples in relations C', D' and E' only encode the truth tables of $\wedge$, $\vee$, and $\neg$, respectively. For example, if $S_k \equiv (S_g \wedge S_h)$, 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_δ is defined to be $\exists y_1 \ldots \exists y_m \exists v(F_1(y_1, ..., y_m, v) \wedge O'(v))$. 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 the tuples in the interpretation of F_1 in J contains a 0 in the last position.

Example 6.1. Let ϕ be the formula $\exists x_1 \exists x_2 \forall y_1(R_1(x_1, y_1) \vee (\neg R_1(x_2, y_1)))$. Recall that the source schema $\mathbf{S}$ consists of relations B, O, U, C, D, E as described above, plus extra relations V_1 and V_2. The target schema $\mathbf{T}$ consists of relations O', U', C', D', E', R_1', F_1, F_2, F_3 and F_4 (because $(R_1(x_1, y_1) \vee (\neg R_1(x_2, y_1)))$ has 4 subformulas). The enumeration of the subformulas 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 $S_1 \equiv (R_1(x_1, y_1) \vee (\neg R_1(x_2, y_1)))$.

The source-to-target dependencies are:

$$\begin{array}{rcl}
V_1(x) & \rightarrow & V_1'(x) \\
O(x) & \rightarrow & O'(x) \\
U(x) & \rightarrow & U'(x) \\
C(x, y, z) & \rightarrow & C'(x, y, z) \\
D(x, y, z) & \rightarrow & D'(x, y, z) \\
E(x, y) & \rightarrow & E'(x, y) \\
B(x_1), B(x_2) & \rightarrow & \exists z R_1'(x_1, x_2, z) \\
B(x_1) & \rightarrow & \exists z F_1(x_1, z)
\end{array}$$

$$
\begin{aligned}
B(x_1) &\rightarrow \exists z F_2(x_1, z) \\
B(x_1) &\rightarrow \exists z F_3(x_1, z) \\
B(x_1) &\rightarrow \exists z F_4(x_1, z)
\end{aligned}
$$

The instance I of **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_α, Q_β, Q_γ and Q_δ in this case are as follows:

- $Q_\alpha \equiv \exists x_1 \exists x_2 \exists n \exists v \exists w (R_1'(x_1, x_2, n) \wedge O'(v) \wedge U'(w) \wedge n \neq v \wedge n \neq w)$;

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

$$
\begin{aligned}
Q_\beta^3 &\equiv \exists y_1 \exists n \exists v \exists x_1 (F_3(y_1, n) \wedge R_1'(x_1, y_1, w) \wedge V_1'(x_1) \wedge n \neq w) \\
Q_\beta^4 &\equiv \exists y_1 \exists n \exists v \exists x_2 (F_4(y_1, n) \wedge R_1'(x_2, y_1, w) \wedge V_2'(x_2) \wedge n \neq w)
\end{aligned}
$$

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

$$
\begin{aligned}
Q_\gamma^1 &\equiv \exists y_1 \exists n \exists v \exists w \exists z (F_1(y_1, n) \wedge F_3(y_1, v) \wedge F_2(y_1, w) \wedge D'(v, w, z) \wedge n \neq z) \\
Q_\gamma^2 &\equiv \exists y_1 \exists n \exists v \exists z (F_2(y_1, n) \wedge F_4(y_1, v) \wedge E'(v, z) \wedge n \neq z)
\end{aligned}
$$

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

This finishes the example. $\square$

We now continue with the proof of the theorem. We prove next that ϕ is satisfiable if and only if $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$.

($\Leftarrow$) Assume first that ϕ is satisfiable. Then, as we mentioned above, it is satisfiable by a structure of cardinality at most p. Let $\mathcal{A}$ be such structure, and assume that the elements of $\mathcal{A}$ are $\{b_1, \ldots, b_p\}$. Further, assume without loss of generality that $\mathcal{A}$ satisfies $\forall y_1, \ldots, \forall y_m \psi$ when we assign to each free variable x_i in ψ 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 $\perp$ such that the tuple $R'_i(a_{i_1}, \ldots, a_{r_i}, \perp)$ belongs to $\text{CAN}(I)$, and the interpretation of R_i in $\mathcal{A}$ contains the tuple $(b_{i_1}, \ldots, b_{r_i})$, then $h(v) = 1$;
- if v is the null value $\perp$ such that the tuple $F'_j(a_{i_1}, \ldots, a_{i_m}, \perp)$ belongs to $\text{CAN}(I)$, and such that the formula S_j holds in $\mathcal{A}$ 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 $Q(J^*) = \texttt{false}$, and, thus, that $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$.

Assume for the sake of contradiction that $Q(J^*) = \texttt{true}$. Clearly, $Q_\alpha(J^*) = \texttt{false}$. Furthermore, $Q_\delta(J^*) = \texttt{false}$: Since $\mathcal{A}$ 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 $\mathcal{A}$ 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 $Q_\beta(J^*) = \texttt{true}$, and, in particular, that $Q_\beta^j(J^*) = \texttt{true}$, for $j \in A$. Further, assume that Q_β^j is of the form

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

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 $f : \{y_1, \ldots, y_m, n, v, \bar{x}\} \to \mathrm{dom}(J^*)$, 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 $f(y_1), \ldots, f(y_p)$ 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 $Q_\gamma(J^*) = \texttt{true}$, and, in particular, that $Q_\gamma^k(J^*) = \texttt{true}$, for some $k \notin \Theta$. Further, assume without loss of generality that S_k is of the form $S_g \vee S_h$ (the other two cases are completely symmetrical). Also assume Q_γ^k is of the form

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

Then there exists a function $f : \{y_1, \ldots, y_n, v, w, z\} \to \mathrm{dom}(J^*)$, 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 $S_k \equiv (S_g \vee S_h)$, 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 $b_1, \dots, b_p$ to $x_1, \dots, x_p$ and elements $f(y_1), \dots, f(y_p)$ to $y_1, \dots, y_m$. This is a contradiction.

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

Construct from J' a structure $\mathcal{A}$ as follows: The domain of $\mathcal{A}$ is $\{a_1, \dots, a_p\}$. 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 ϕ. In order to do so we start by proving the following: We prove that for every $j \in [1, \ell]$, if the tuple $F_j(a_{i_1}, \dots, a_{i_m}, 1)$ belongs to J', then $\mathcal{A}$ satisfies the formula S_j when we assign $a_{i_1}, \dots, a_{i_m}$ to $y_1, \dots, y_m$ and $a_1, \dots, a_p$ to $x_1, \dots, x_p$. We prove this by induction on the structure of the subformulas of ψ.

- For the base case, assume that $S_j = R_i(\bar{x}, \bar{y})$, where $\bar{x}$ is a tuple of variables in $\{x_1, \dots, x_p\}$ and $\bar{y}$ is a tuple of variables in $\{y_1, \dots, y_m\}$. Further, assume that $F_j(b_{i_1}, \dots, b_{i_m}, 1)$ belongs to J'.
  Let $g : \{x_1, \dots, x_p, y_1, \dots, y_m\} \to \text{dom}(J')$ be a function, such that $g(x_i) = a_i$, for each $i \in [1, p]$, and $g(y_j) = a_{i_j}$, for each $j \in [1, m]$. We know that $\textsc{Can}(I)$ 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_β is of the form

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

  and we know that $Q^j_\beta(J') = \texttt{false}$, 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 $a_{i_1}, \dots, a_{i_m}$ to $y_1, \dots, y_m$ and $a_1, \dots, a_p$ to $x_1, \dots, x_p$.
- For the inductive case, assume that $S_j \equiv (S_{j_1} \wedge S_{j_2})$ (the cases where $S_j \equiv (S_{j_1} \vee S_{j_2})$ and $S_j \equiv (\neg S_{j_1})$ are completely symmetrical). Further, assume that $F_j(a_{i_1}, \dots, a_{i_m}, 1)$ belongs to J'. We know there are tuples $F_{j_1}(a_{i_1}, \dots, a_{i_m}, \bot_{j_1})$ and $F_{j_2}(a_{i_1}, \dots, a_{i_m}, \bot_{j_2})$ in $\textsc{Can}(I)$, where $\bot_{j_1}$ and $\bot_{j_2}$ are null values. Therefore, J' contains the tuples $F_g(a_{i_1}, \dots, a_{i_m}, h(\bot_{j_1}))$ and $F_h(a_{i_1}, \dots, a_{i_m}, h(\bot_{j_2}))$.

We know that Q^k_γ is of the form

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

Further, $Q^k_\gamma(J') = \texttt{false}$. Then it must be the case that $h(\bot_{j_1}) = h(\bot_{j_2}) = 1$. 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 $Q_\delta(J') = \texttt{false}$, 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 ϕ. 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$^{\mathbf{C}(\neq)}$ programs, this restriction does not help in reducing the complexity of computing certain answers. However, the evaluation of CQ$^{\neq}$ queries is not inherently exponential and, thus, we are able to considerably reduce the complexity by considering LAV settings, as we show in the following proposition.

PROPOSITION 6.2. CERTAIN-ANSWERS(LAV, 1-CQ$^{\neq}$) *is* NP*-complete, and for every* $k \geq 2$, CERTAIN-ANSWERS(LAV, k-CQ$^{\neq}$) *is* Π^p_2*-complete* .

Proof: We only show that $\textsc{Certain-Answers}(\textsc{Lav}, k\text{-CQ}^{\neq})$ is Π_2^p-complete. That the problem $\textsc{Certain-Answers}(\textsc{Lav}, 1\text{-CQ}^{\neq})$ is 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 Π_2^p. The certain answers to each k-UCQ$^{\neq}$ queries for a source instance I, under a setting $\mathcal{M}$, can be computed in coNP time in the size of $\textsc{Can}(I)$ (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 $\textsc{Lav}$ settings, it is clear that $\textsc{Can}(I)$ 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 NP oracle in order to check that the evaluation of Q over J is empty, thus obtaining a Π_2^p bound.

The Π_2^p hardness is established from a reduction of $\forall \bar{x}\ \exists \bar{z}$ 3-SAT$(\bar{x}, \bar{z})$, which is the following problem: Given a boolean formula ϕ 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 ϕ' is satisfied with the overall assignment? This problem is widely known to be Π_2^p-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 ψ_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 ϕ, we will show a $\textsc{Lav}$ data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, a query Q and an instance I such that $\mathrm{certain}_{\mathcal{M}}(Q, I) =$ `true` if and only if ϕ is satisfiable.

- the $\textsc{Lav}$ setting ($\mathbf{S}$,$\mathbf{T}$,Σ_{st}) is as follows:
  The Source schema $\mathbf{S}$ 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 Σ_{st} of source-to-target dependencies is:

$$\hat{O}(x) \rightarrow O(x) \tag{6.1}$$

$$\hat{I}(x) \rightarrow I(x) \tag{6.2}$$

$$\begin{aligned} F(x,y) \rightarrow\ & \exists z, w\big(T(x,z), R(y,x,z), TV(x,z), R(x,w,w), \\ & C(w,w,w,w,w,w,w,w)\big) \end{aligned} \tag{6.3}$$

$$S(x,y) \rightarrow T(x,y) \tag{6.4}$$

$$E(x,y,z) \rightarrow TV(x,z), R(y,x,z) \tag{6.5}$$

$$\hat{C}(a,b,c,d,e,f,g,h) \rightarrow C(a,b,c,d,e,f,g,h) \tag{6.6}$$

- 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 $\psi_k = (s_{1k} \vee s_{2k} \vee s_{3k})$ in ϕ ($1 \leq k \leq \ell$). The variables u, v, w represent the values of satisfying assignments for ψ_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 ϕ. For example, if $\phi \equiv (x_1 \vee \neg x_2 \vee z_1)$, 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 $\mathbf{S}$, because $(0, 0, 1)$ is not a satisfying assignment for ϕ

- Finally, the query Q is as follows:

$$\begin{aligned} Q \equiv\ & \exists e \exists f \exists g \exists u \exists x_1, ..., x_n \exists v_1, ..., v_n \exists z_1, ..., z_m \exists w_1, ..., w_m \exists y_1, ..., y_{\ell+1} \\ & \Big(\bigwedge_{1 \le i \le n} \big(TV(x_i, v_i) \wedge R(g, x_i, v_i)\big) \wedge \bigwedge_{1 \le j \le m} \big(TV(z_j, w_j) \wedge R(g, z_j, w_j)\big) \wedge \\ & \bigwedge_{1 \le k \le p} \big(C(s_{1k}, s_{2k}, s_{3k}, v_{1k}, v_{2k}, v_{3k}, y_k, y_{k+1})\big) \wedge I(e) \wedge O(f) \wedge T(g, u) \wedge u \neq e, u \neq f \Big) \end{aligned}$$

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 $i \in \{1, ..., n\}$
- 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 ϕ 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 ϕ 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 $i \in \{1, ..., n\}$.

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

Suppose $\phi \equiv \forall x_1, x_2\ \exists z_1\ (x_1 \vee \neg x_2 \vee \neg z_1)$. We then create the following LAV setting $(\mathbf{S}, \mathbf{T}, \Sigma_{st})$:

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

The canonical solution contains four null values: $\#_1$, $\perp_1$ and $\#_2$, $\perp_2$, 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:

- $I = \{(1)\}, O = \{(0)\}$
- $T = \{(\hat{x}_1, \perp_1), (\hat{x}_2, \perp_2), (a, b)\}$
- $TV = \{(\hat{x}_1, \perp_1), (\hat{x}_2, \perp_2), (\hat{z}_1, 1), (\hat{z}_1, 0), (\#_1, \#_1), (\#_2, \#_2)\}$
- $R = \{(a, \hat{x}_1, \perp_1), (a, \hat{x}_2, \perp_{x_2}), (a, \hat{z}_1, 1), (a, \hat{z}_1, 0), (\hat{x}_1, \#_1, \#_1), (\hat{x}_2, \#_2, \#_2)\}$
- $C = \{(\hat{x}_1, \hat{x}_2, \hat{z}_1, 0, 0, 0, p_1, p_2), (\hat{x}_1, \hat{x}_2, \hat{z}_1, 0, 0, 1, p_1, p_2),$ $(\hat{x}_1, \hat{x}_2, \hat{z}_1, 0, 1, 0, p_1, p_2), (\hat{x}_1, \hat{x}_2, \hat{z}_1, 1, 0, 0, p_1, p_2), (\hat{x}_1, \hat{x}_2, \hat{z}_1, 1, 0, 1, p_1, p_2),$ $(\hat{x}_1, \hat{x}_2, \hat{z}_1, 1, 1, 0, p_1, p_2), (\hat{x}_1, \hat{x}_2, \hat{z}_1, 1, 1, 1, p_1, p_2),$ $(\#_1, \#_1, \#_1, \#_1, \#_1, \#_1, \#_1, \#_1), (\#_2, \#_2, \#_2, \#_2, \#_2, \#_2, \#_2, \#_2)\}$

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

$$\begin{aligned} Q' \;\equiv\; & I(e) \wedge O(f) \wedge T(x, u) \wedge TV(x_1, v_1) \wedge TV(x_2, v_2) \wedge R(x, x_1, v_1) \wedge R(x, x_2, v_2) \wedge \\ & \wedge TV(z_1, w_1), R(x, z_1, w_1) \wedge C(x_1, x_2, z_1, v_1, v_2, w_1, y_1, y_2) \wedge u \neq e \wedge u \neq f \end{aligned}$$

Intuitively, the relation TV contains the truth value of every variable in ϕ, 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

q. 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 $J^\star$ has all of the $\perp$s replaced by 0 or 1. In order for the evaluation of Q over A_J to be true, there must exists 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 ϕ. Due to Σ_{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 $J^\star$ will be true. Naturally, if we can do this for all the solutions J that represent a valuation of the universally quantified set, then ϕ will be $\forall\exists$-satisfiable.

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

($\Leftarrow$) If ϕ is not $\forall\exists$-satisfiable, that is, there is valuation σ_x for the universally quantified variables such that for every valuation σ_z for the existentially quantified variables ϕ 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 ϕ and $\sigma_x(x_i)$ = 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 ϕ and $\sigma_x(x_i)$ = 0
- $h(y)$ = y otherwise.

Let $J^\star$ be the solution obtained from the canonical solution J by replacing each element y with $h(y)$. We show that $certain_{\mathcal{M}}(Q, I) = \texttt{false}$.

Assume for the sake of contradiction that there is a tuple in the evaluation of Q over $J^\star$. The witness of the relation T in Q cannot be a pair of the form $(\hat{x}_i, \perp_i)$, because then $h(\perp_i) = \perp_i$, 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 $(\hat{s}_{11}, \hat{s}_{21}, \hat{s}_{31}, v_{11}, v_{21}, v_{31}, p_1, p_2)$, .., $(\hat{s}_{1\ell}, \hat{s}_{2\ell}, \hat{s}_{3\ell}, v_{1\ell}, v_{2\ell}, v_{3\ell}, p_\ell, p_{\ell+1})$ plus a series of tuples of the form $TV(\hat{x}_i, v_i)$, $TV(\hat{z}_j, v_j)$ for $i \in \{1, ..., n\}$ and $j \in \{1, ..., m\}$ witnessing all of the relations C and TV in the query. Choose σ_z such that for $j \in \{1, ..., m\}$, $\sigma_z(z_j) = 1$ 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 σ_x and σ_z will satisfy all of the clauses in ϕ, which is a contradiction, since ϕ is not satisfiable.

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

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

Assume for the sake of contradiction that ϕ is satisfiable: for every valuation of the universally quantified variables there is a valuation for the existentially quantified variables that satisfy ϕ. Then, in particular, for σ_x there must exist a valuation σ_z such that σ_x, σ_z satisfy ϕ. There are tuples $(\hat{s}_{11}, \hat{s}_{21}, \hat{s}_{31}, v_{11}, v_{21}, v_{31}, p_1, p_2)$, .., $(\hat{s}_{1\ell}, \hat{s}_{2\ell}, \hat{s}_{3\ell}, v_{1\ell}, v_{2\ell}, v_{3\ell}, p_\ell, p_{\ell+1})$ in $J^\star$, and, since ϕ 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 $(\hat{x}_i, h(\perp_i))$ as witness to the relations TV for the query Q, then the evaluation of Q over $J^\star$ will be nonempty, and thus $certain_{\mathcal{M}}(Q, I) = \texttt{true}$. This is again a contradiction, so we prove that ϕ can not be satisfiable. $\square$

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$^{\neq}$ is the class of unions of k-CQ$^{\neq}$ queries, then

PROPOSITION 6.3.

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

(2) CERTAIN-ANSWERS$(\text{LAV}, 1\text{-UCQ}^{\neq})$ *is* NP-*complete, and, for every* $k \geq 2$*, the problem* CERTAIN-ANSWERS$(\text{LAV}, k\text{-UCQ}^{\neq})$ *is* Π_2^p*-complete.*

7. CONCLUSION AND FUTURE RESEARCH

7.1. General Remarks

In this work, we proposed the language $\text{DATALOG}^{\mathbf{C}(\neq)}$ that extends DATALOG with a restricted form of negation, and studied some of its fundamental properties. In particular, we showed that the certain answers to a $\text{DATALOG}^{\mathbf{C}(\neq)}$ program can be computed in polynomial time (in terms of data complexity), and we used this property to find tractable fragments of the class of unions of conjunctive queries with inequalities. We also studied the combined complexity of computing certain answers to $\text{DATALOG}^{\mathbf{C}(\neq)}$ 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 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 $\text{DATALOG}^{\mathbf{C}(\neq)}$ 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 $\text{DATALOG}^{\mathbf{C}(\neq)}$ 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 $\text{DATALOG}^{\mathbf{C}(\neq)}$ program Π_q. It would also be important to establish an upper bound to the expressive power of $\text{DATALOG}^{\mathbf{C}(\neq)}$ 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 $\text{DATALOG}^{\mathbf{C}(\neq)}$ program.

Concerning the use of $\text{DATALOG}^{\mathbf{C}(\neq)}$ 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

Abiteboul, S., & Duschka, O. (1999). Complexity of Answering Queries Using Materialized Views . *Gemo Report 383*.

Abiteboul, S., Hull, R., & Vianu, V. (1995). *Foundations of databases*. Addison-Wesley.

Afrati, F., & Kolaitis, P. G. (2008). Answering aggregate queries in data exchange. In *Pods '08: Proceedings of the twenty-seventh acm sigmod-sigact-sigart symposium on principles of database systems* (pp. 129–138). New York, NY, USA: ACM.

Arenas, M., Barceló, P., Fagin, R., & Libkin, L. (2004). Locally consistent transformations and query answering in data exchange. In *Pods '04: Proceedings of the twenty-third acm sigmod-sigact-sigart symposium on principles of database systems* (pp. 229–240). New York, NY, USA: ACM.

Atserias, A., Dawar, A., & Kolaitis, P. G. (2006). On preservation under homomorphisms and unions of conjunctive queries. *J. ACM*, *53*(2), 208–237.

Beeri, C., & Vardi, M. Y. (1984). A proof procedure for data dependencies. *J. ACM*, *31*(4), 718–741.

Börger, E., Grädel, E., & Gurevich., Y. (2001). *The classical desicion problem*. Springer.

Chandra, A. K., & Merlin, P. M. (1977). Optimal implementation of conjunctive queries in relational data bases. In *Stoc '77: Proceedings of the ninth annual acm symposium on theory of computing* (pp. 77–90). New York, NY, USA: ACM.

Fagin, R., Kolaitis, P., Miller, R., & Popa, L. (2005). Data exchange: Semantics and query answering. *Theoretical Computer Science*, *336*(1), 89–124.

Fagin, R., Kolaitis, P. G., & Popa, L. (2005). Data exchange: getting to the core. *ACM Trans. Database Syst.*, *30*(1), 174–210.

Fagin, R., Kolaitis, P. G., Popa, L., & Tan, W.-C. (2005). Composing schema mappings: Second-order dependencies to the rescue. *ACM Trans. Database Syst.*, *30*(4), 994–1055.

Gottlob, G., & Papadimitriou, C. (2003). On the complexity of single-rule datalog queries. *Inf. Comput.*, *183*(1), 104–122.

Greenlaw, R., Hoover, H., & Ruzzo, W. (1995). *Limits to parallel computation: P-completeness theory*. Oxford University Press.

Haas, L. M., Hernández, M. A., Ho, H., Popa, L., & Roth, M. (2005). Clio grows up: from research prototype to industrial tool. In *Sigmod '05: Proceedings of the 2005 acm sigmod international conference on management of data* (pp. 805–810). New York, NY, USA: ACM.

Imielinski, T., & Lipski, W. (1983). Incomplete information and dependencies in relational databases. *SIGMOD Rec.*, *13*(4), 178–184.

Kolaitis, P. G. (2005). Schema mappings, data exchange, and metadata management. In *Pods '05: Proceedings of the twenty-fourth acm sigmod-sigact-sigart symposium on principles of database systems* (pp. 61–75). New York, NY, USA: ACM.

Kolaitis, P. G., Panttaja, J., & Tan, W.-C. (2006). The complexity of data exchange. In *Pods '06: Proceedings of the twenty-fifth acm sigmod-sigact-sigart symposium on principles of database systems* (pp. 30–39). New York, NY, USA: ACM.

Lenzerini, M. (2002). Data integration: a theoretical perspective. In *Pods '02: Proceedings of the twenty-first acm sigmod-sigact-sigart symposium on principles of database systems* (pp. 233–246). New York, NY, USA: ACM.

Libkin, L. (2004). *Elements of finite model theory*. Springer.

Libkin, L. (2006). Data exchange and incomplete information. In *Pods '06: Proceedings of the twenty-fifth acm sigmod-sigact-sigart symposium on principles of database systems* (pp. 60–69). New York, NY, USA: ACM.

Madry, A. (2005). Data exchange: On the complexity of answering queries with inequalities. In (Vol. 94, pp. 253–257).

Shu, N. C., Housel, B. C., Taylor, R. W., Ghosh, S. P., & Lum, V. Y. (1977). Express: a data extraction, processing, and restructuring system. *ACM Trans. Database Syst.*, *2*(2), 134–174.

Vardi, M. Y. (1982). The complexity of relational query languages (extended abstract). In *Stoc '82: Proceedings of the fourteenth annual acm symposium on theory of computing* (pp. 137–146). New York, NY, USA: ACM.

APPENDIX A. ADDITIONAL PROOFS

Proof of Proposition 5.3

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

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

Furthermore, Boolean query Q is defined as:

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

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

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

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

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

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

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

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

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

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

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

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

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

($\Leftarrow$) Assume that $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$. Then there exists a solution J' such that $Q(J') = \texttt{false}$. Let $h : J \rightarrow J'$ be an homomorphism from J into J', and let κ

be the following truth assignment for the propositional variables mentioned in ϕ: $\kappa(q) = 1$ iff $h(\bot_q) = q$. We show next that $\kappa(C_j) = 1$, for each $1 \leq j \leq m$, and, thus, that ϕ is satisfiable.

Consider an arbitrary $j \in [1, m]$, and assume that $C_j = (\alpha \vee \beta \vee \gamma)$. Then, since $S(\alpha, \beta, \gamma)$, $T(\alpha, h(\bot_\alpha))$, $T(\beta, h(\bot_\beta))$, 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 CERTAIN-ANSWERS$(\mathcal{M}, Q)$ is CONP-complete.

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

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

The Boolean query Q is as follows:

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

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

Membership in CONP follows from (Fagin, Kolaitis, Miller, & Popa, 2005). The CONP-hardness is established from a reduction from POSITIVE-NOT-ALL-EQUAL-3SAT, which is the following decision problem: Given a propositional formula ϕ in 3CNF consisting entirely of positive clauses $(p \vee q \vee r)$, is there a valuation to the propositional variables of ϕ such that for every clause of ϕ at least one variable is assigned value 1 and at least one variable is assigned value 0? This problem is known to be NP-hard (see e.g. the proof of Theorem 5.11 in (Fagin, Kolaitis, Miller, & Popa, 2005)). More precisely, for every 3CNF propositional formula ϕ consisting entirely of positive clauses, we construct in polynomial time an instance I_ϕ of S such that ϕ is NOT-ALL-EQUAL-satisfiable iff $\text{certain}_{\mathcal{M}}(Q, I_\phi) = \texttt{false}$.

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

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

Clearly, I_ϕ can be constructed in polynomial from ϕ.

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

- The interpretation of the relation P in J is just a copy of the interpretation of the relation M in I_ϕ;

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

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

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

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

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

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

(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 ϕ. Further, $h(y_1) = f(\bot_q)$,

$h(y_2) = 1$, and $h(y_3) = f(\#_q)$. It follows that $f(\perp_q) \neq 1$ 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 ϕ. Then there is a clause $(p \vee q \vee r)$ in ϕ such that $h(x_1) = p$ and $h(x_3) = r$. Further, $h(y_1) = f(\perp_p)$, $h(y_2) = f(\#_q)$, and $h(y_3) = f(\perp_r)$, and $f(\perp_p) \neq f(\#_q)$ and $f(\#_q) \neq f(\perp_r)$. It follows from the definition of f that $f(\perp_p) = f(\perp_q) = f(\perp_r)$, and, thus, that $\kappa(p) = \kappa(q) = \kappa(r)$. This is a contradiction because κ is NOT-ALL-EQUAL.

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

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

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

Consider now an arbitrary $j \in [1, m]$, and assume that $C_j = (p \vee q \vee r)$. Consider the function $f : \{x_1, y_1, x_2, y_2, x_3, y_3\} \to \text{dom}(J')$, such that $f(x_1) = p$, $f(y_1) = h(\perp_p)$, $f(x_2) = q$, $f(y_2) = h(\#_q)$, $f(x_3) = r$, and $f(y_3) = h(\perp_r)$. Then

$P(f(x_1), f(x_2), f(x_3))$, $R(f(x_1), f(y_1))$, $S(f(x_2), f(y_2))$, as well as $R(f(x_3), f(y_3))$ belong to J'. Therefore, since $Q(J') = \texttt{false}$, it must be the case that $h(\bot_p) = h(\#_q)$ or $h(\#_q) = h(\bot_r)$. From the previous remark, either $\kappa(p) = 1 - \kappa(q)$ or $\kappa(q) = 1 - \kappa(r)$. In any case, $\kappa(C_j) = 1$, and it is not the case that $\kappa(p) = \kappa(q) = \kappa(r) = 1$.

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

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

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

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

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

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

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

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

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

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

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

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

- The interpretation of S in J contains the pairs $(q, \neg q)$, $(\neg q, q)$, and $(\#_q, \#_q)$, for each propositional variable q mentioned in ϕ;
- the interpretation of T in J contains a copy of the interpretation of N in I_ϕ and the pairs $(q, \perp_q)$, $(\neg q, \perp_q)$, for each propositional variable q mentioned in ϕ;

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

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

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

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

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

Assume, for the sake of contradiction, that $Q(J^*) = \texttt{true}$. Then there is a function $h : \{x, x', y, y', z, z', x_1, y_1, x_2, y_2\} \rightarrow$ dom(J^*), such that $T(h(x_1), h(y_1))$, $T(h(x_2), h(y_2)), U(h(x), h(y), h(z)), S(h(x), h(x')), S(h(y), h(y')), S(h(z), h(z'))$, $V(h(x_1), h(x_2), h(x'), h(x'))$, $V(h(x_1), h(x_2), h(y'), h(y'))$, and $V(h(x_1), h(x_2), h(z'), h(z'))$ belong to J^*, and, furthermore, $h(x_1) \neq h(y_1)$ and $h(x_2) \neq h(y_2)$. Since $V(h(x_1), h(x_2), h(x'), h(x'))$ belongs to J^*, there are only two cases to consider with respect to the values $h(x_1)$ and $h(x_2)$:

(i) The first case is that $h(x_1) = q$ and $h(x_2) = \neg q$, for some propositional variable q mentioned in ϕ. But then $h(y_1) = h(y_2) = f(\bot_q)$, because $T(h(x_1), h(y_1)$ and $T(h(x_2), h(y_2))$ belong to J^* . It follows that $f(\bot_q) \neq q$ and $f(\bot_q) \neq \neg q$, which is in contradiction with the definition of the function f.

(ii) The second case is that $h(x_1) = \mathtt{a}$ and $h(x_2) = \mathtt{c}$. But then $h(x') = q$ or $h(x') = \neg q$, for some propositional variable q mentioned in ϕ. Since $S(h(x), h(x'))$ belongs to J^*, it must be the case that for some clause $(\alpha \vee \beta \vee \gamma)$ in ϕ, $h(x) = \alpha$, $h(y) = \beta$ and $h(z) = \gamma$. Furthermore, $h(x') = \neg\alpha$. Since $\kappa(\phi) = 1$, it must be the case that $\kappa(\alpha) = 1$ or $\kappa(\beta) = 1$ or $\kappa(\gamma) = 1$. Assume that $\kappa(\alpha) = 1$. Since $V(h(x_1), h(x_2), h(x'), h(x')) = V(a, c, \neg\alpha, \neg\alpha)$ belongs to J^*, it follows that $f(\bot_\alpha) = \neg\alpha$. But then $\kappa(\alpha) = 0$, which contradicts our previous assumption. The cases $\kappa(\beta) = 1$ or $\kappa(\gamma) = 1$ are identical.

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

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

We prove second that for each propositional variable q mentioned in ϕ, $h(\bot_q) = q$ or $h(\bot_q) = \neg q$. Assume otherwise. Then the function $f : \{x, x', y, y', z, z', x_1, y_1, x_2, y_2\} \to \mathrm{dom}(J')$ 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 $Q(J') = \texttt{true}$, 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 $\text{CQ}^{\neq}$ 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 $\bar{t} \in \Pi(\emptyset)$. 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).

As in (Kolaitis et al., 2006), let Π be a program containing some facts with only constants and a single rule of the form $A(\bar{x}) \leftarrow Q_1(\bar{x}_1), \ldots, Q_n(\bar{x}_n)$, where each symbol Q_i ($1 \leq i \leq n$) either represents an extensional database predicate or the only intensional predicate A. Furthermore, we assume that $\bar{t} = (c_1, \ldots, c_k)$ and we say that $\bar{t}$ belongs to the evaluation of Π over the empty instance if and only if $\bar{t} \in A^{\mathcal{T}_\Pi^\infty(\emptyset)}$.

The idea of the reduction in (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$^{\neq}$ 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$^{\neq}$ query with one inequality.

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

- The source schema $\mathbf{S}$ consists of four unary relations T, V, F, S plus all the extensional database predicates R_1, ..., 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 Π, denoted by l_i, the arity of the relation A is k and the arity of the relation W is $k+1$.
- The target schema $\mathbf{T}$ consists of relations R'_1, ..., 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,..., 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 Π and $\bar{t}$.

  Intuitively, the constants 1 and 2 will allow us to use the same query for both simulating the sirup rule and checking 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 Σ_{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:

  $$\begin{aligned} F(z), T(y), R_1(x_1, ..., x_{l_1}) &\rightarrow R'_1(x_1, ..., x_{l_1}, y, z, z) \\ &\vdots \\ F(z), T(y), R_m(x_1, ..., x_{l_m}) &\rightarrow R'_m(x_1, ..., x_{l_m}, y, z, z) \end{aligned}$$

  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:

  $$\begin{aligned} S(z), T(y), V(x_1), ..., V(x_{l_1}) &\rightarrow R'_1(x_1, ..., x_{l_1}, y, z, z) \\ &\vdots \\ S(z), T(y), V(x_1), ..., V(x_{l_m}) &\rightarrow R'_m(x_1, ..., x_{l_m}, y, z, z) \end{aligned}$$

  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 Π 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:

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

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

  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:

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

$$S(z), T(y), V(x_1), ..., V(x_k) \quad \rightarrow \quad A'(x_1, ..., x_k, y, z, y) \tag{A.4}$$

- To define query Q, recall that we are considering a sirup rule of the form $A(\bar{x}) \leftarrow Q_1(\bar{x}_1), ..., Q_n(\bar{x}_n)$, 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:

$$Q \; = \; \exists u \exists v \exists w \exists w_1 \cdots \exists w_n \exists x_1 \cdots \exists x_k \exists y \exists z \, \Big(A'(x_1, \ldots, x_k, z, u, w) \wedge$$
$$T'(y) \, \wedge A'(x_1, \ldots, x_k, y, w, v) \wedge z \neq y \wedge \bigwedge_{1 \leq i \leq n} Q'_i(\bar{x}_i, y, u, w_i) \Big).$$

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 $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{true}$, 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 dom(I), mapping (A.2) includes in A' a tuple of the form $(\bar{b}, \perp, 1, 2)$, where $\perp$ is a fresh null value. We shall generically describe the null values added by (A.2) as $\perp$. 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 dom(I), mapping (A.4) includes in A' the tuple $(\bar{b}, c, 2, c)$.

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

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

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

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

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

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

- $i = 0$: 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 $j \in \{1, \dots, n\}$, if $Q'_j = R'_p$, for some $p \in \{1, \dots, m\}$, then $R_p(\bar{b}_j)$ is a fact in Π, and if $Q'_j = A'$, then $A(\bar{b}_j)$ is a fact in Π. Therefore, by definition of $\bar{t}_0$, we conclude that $A(\bar{a}_0)$ can be deduced from the facts of Π and, thus, $A(\bar{a}_0) \in \mathcal{T}_\Pi^1(\emptyset)$.

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

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

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

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

We build a sequence of solutions J_0, ..., J_i, ... and a corresponding sequence of sets of tuples T_0, ..., T_i, ... as follows.

- Let J_0 be the canonical universal solution for I under $\mathcal{M}$, and $T_0 = \hat{Q}(J_0)$, that is, the evaluation of $\hat{Q}$ over J_0.

- For every $i \geq 0$, let J_{i+1} be obtained from J_i by replacing every null value $\perp$ in a tuple of T_i by the constant c, if $\perp$ witnesses the inequality of $\hat{Q}$. Moreover, let $T_{i+1} = \hat{Q}(J_{i+1})$.

Given that J_0 has finite number of null values, we have that the sequence $J_0, \ldots, J_i, \ldots$ is finite, and we let J_m be its last element.

Next we show that from the assumption $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$, one can deduce that $Q(J_m) = \texttt{false}$. Let ψ be the following dependency:

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

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

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

- $i = 0$: Assume that $A(\bar{a})$ holds in $\mathcal{T}_\Pi^0(\emptyset)$. Then $A(\bar{a})$ is a fact in Π or it can be deduced from the facts of Π by using the only rule in this program. Thus, by the definition of Π, it is easy to see that in both cases $A'(\bar{a}, c, 1, 2)$ holds in J_1, since every fact in Π of the form $A(\bar{b})$ appears in J_0 as $A'(\bar{b}, c, 1, 2)$, and every fact in Π 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 $\mathcal{T}_\Pi^i(\emptyset)$. 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 $A'(\bar{a}, \perp, 1, 2)$ holds in J_i, where $\perp$ is a null value, and that $A(\bar{a})$ does not hold in $\mathcal{T}_\Pi^{i-1}(\emptyset)$ (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 $\mathcal{T}_\Pi^{i-1}(\emptyset)$ by using the only rule in Π. Thus, there exists an instantiation $A(\bar{a}) \leftarrow Q_1(\bar{a}_1), \ldots, Q_n(\bar{a}_n)$ of the rule of Π such that $Q_1(\bar{a}_1)$, ..., $Q_n(\bar{a}_n)$ belong to $\mathcal{T}_\Pi^{i-1}(\emptyset)$. Thus, by induction hypothesis and the definition of the sequence J_0, ..., J_m, we conclude that for every $p \in \{1, \ldots, n\}$, if $Q'_p = R_q$ for some $q \in \{1, \ldots, m\}$, then $R_q(\bar{a}_p, c, 1, 1)$ holds in J_{i-1}, and if $Q'_p = A'$, then $A'(\bar{a}_p, c, 1, 2)$ holds in J_{i-1}. Therefore, given that both $A'(\bar{a}, \perp, 1, 2)$ and $A'(\bar{a}, c, 2, c)$ hold in J_i, we conclude that one of the tuples of T_i has $\perp$ as a witness for the inequality of $\hat{Q}$. This implies that $A'(\bar{a}, c, 1, 2)$ holds in J_{i+1} since $\perp$ is replaced by c to obtain J_{i+1} from J_i.

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

Proof of Proposition 6.3

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

We will need some additional notation. Let again ℓ 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 Θ, that is, the number of atomic subformulas of ϕ. We will assume that Θ 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 Θ.

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

- The source schema $\mathbf{S}'$ consists of six unary relations D, E, B, C, O and U, a set $\{Q_1, ..., Q_n\}$ of unary relations (one for each relation R_i), another set of unary relations $\{V_1, ..., V_p\}$ (recall that p is the number of existentially quantified variables in ϕ), a binary relation G, a relation Z of arity 4, and a relation A of arity equal to $|\Theta| + 5$.
- The target schema $\mathbf{T}'$ 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 ϕ), and a set of binary relations $\{V_1', ..., V_p'\}$.
- The instance I' is as follows. The domain of I' contains the elements $a_1, \dots, a_p$, and elements $s_0, \dots, s_\ell, s_f$, $c_1, \dots, c_n$ 1, d, and 0. The interpretation of each symbol in $\mathbf{S}$ in I' is as follows:
  - $B^{I'} = \{a_1, \dots, a_p\}$.
  - $O^{I'} = \{0\}$, $U^{I'} = \{1\}$ and $D^{I'} = \{d\}$.
  - $E^{I'} = \{s_0, s_f\}$ and $C^{I'} = \{s_1, \dots, s_\ell\}$.
  - $Q_k^{I'} = \{c_k\}$ for every $k \in [1, n]$.
  - $V_i^{I'} = \{a_i\}$ for every $i \in [1, p]$.
  - $G^{I'} = \{(s_0, 0), (s_f, 0)\} \cup \{(s_i, 1), (s_i, 0) \mid i \in [1, \ell]\}$.
  - $A^{I'}$ is as follows:
    * It contains a tuple with only ds, 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: $A(s_f, d, d, d, d, d, s_1, 0)$.
    * For each relation $i \in [1, n]$, $A^{I'}$ will contain a tuple in which every position contains the element d, except for the first position that contains the element s_0, the second position that contains the element c_i, and the last position that contains the element 1.
    * For each subformula S_j of ψ of the form $R_i(\bar{z})$, $A^{I'}$ 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: $A(s_2, d, c_2, d, d, d, d, 0)$ and $A(s_2, d, c_2, d, d, d, d, 1)$

  * Then for each subformula S_j, $j \notin \Theta$, such that $S_j \equiv (S_g \wedge S_h)$ or $S_j \equiv (S_g \vee S_h)$, $A^{I'}$ contains two tuples, both with only d's except for 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 $S_1 \equiv (S_2 \wedge S_3)$ 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, $j \notin \Theta$, such that $S_j \equiv (\neg S_g)$, $A^{I'}$ 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 $A(s_1, d, d, d, d, s_2, 0)$ and $A(s_1, d, d, d, d, s_2, 1)$.

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

  This finishes the definition of I'.

• The set Σ'_{st} of source-to-target dependencies is as follows:

– We create a copy of G, A and Z into G', A' and Z', respectively:

$$G(x,y) \rightarrow G'(x,y) \tag{A.5}$$

$$A(\bar{x}) \rightarrow A'(\bar{x}) \tag{A.6}$$

$$Z(x,y,z,w) \rightarrow Z'(x,y,z,w) \tag{A.7}$$

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

$$\begin{aligned} V_1(x), C(y) &\rightarrow V_1'(y,x) \\ &\vdots \\ V_p(x), C(y) &\rightarrow V_p'(y,x) \end{aligned} \tag{A.8}$$

– 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':

$$\begin{aligned} B(x), E(z) &\rightarrow V_1'(x,z) \\ &\vdots \\ B(x), E(z) &\rightarrow V_p'(x,z) \end{aligned} \tag{A.9}$$

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

$$\begin{aligned} Q_1(y), E(z), D(w), O(v), & \\ B(x_1), ..., B(x_{r_1}), D(x_{r_1+1}), ..., D(x_s) &\rightarrow \exists n(R'(y, x_1, ..., x_s, n), \\ & \quad R'(w, x_1, ..., x_1, n), Z'(z, v, v, n)) \\ &\vdots \\ Q_n(y), E(z), D(w), O(v), & \\ B(x_1), ..., B(x_{r_n}), D(x_{r_1+1}), ..., D(x_s) &\rightarrow \exists n(R'(y, x_1, ..., x_s, n), \\ & \quad R'(w, x_1, ..., x_1, n), Z'(z, v, v, n)) \end{aligned} \tag{A.10}$$

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 $\textsc{Can}(I')$: $(s_i, a_{j_1}, \ldots, a_{j_{r_i}}, d, \ldots, d, \perp)$ and $(d, a_{j_1}, \ldots, a_{j_1}, \perp)$, where $\perp$ is a fresh null value. In such case, we also add the tuples $(s_0, 0, 0, \perp)$ and $(s_f, 0, 0, \perp)$ to the interpretation of Z' in $\textsc{Can}(I')$.

– The following stds are also in Σ_{st}:

$$C(y), B(x_1), ..., B(x_m) \quad \rightarrow \quad \exists n F(y, x_1, ..., x_m, n) \tag{A.11}$$

The idea is that the interpretation of F in $\textsc{Can}(I')$ 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 $(s_j, a_{i_1}, \ldots, a_{i_m}, \perp)$, where $\perp$ is a fresh null value.

– The following are also in Σ_{st}:

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

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

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 $\textsc{Can}(I')$.

– Finally, we also add the following stds to Σ_{st}:

$$D(y), O(z), B(x_1), ..., B(x_m) \quad \rightarrow \quad F(y, x_1, ..., x_m, z) \tag{A.14}$$

$$D(y), U(z), B(x_1), ..., B(x_m) \quad \rightarrow \quad F(y, x_1, ..., x_m, z) \tag{A.15}$$

$$E(y), O(z), B(x_1), ..., B(x_m) \quad \rightarrow \quad F(y, x_1, ..., x_m, z) \tag{A.16}$$

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 $\textsc{Can}(I')$. 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 $\textsc{Can}(I')$.

This finishes the definition of Σ_{st}.

Informally, we code in $\textsc{Can}(I')$ the interpretation of each relation R_i' in $\textsc{Can}(I)$ using the symbol R': $R_i'(\bar{a}, \bot)$ belongs to $\textsc{Can}(I)$ iff $R'(c_i, \bar{a}, d, \ldots, d, \bot)$ belongs to $\textsc{Can}(I')$. The interpretation of each F_j in $\textsc{Can}(I)$ is coded in the same way using F': $F_j(\bar{b}, \bot)$ belongs to $\textsc{Can}(I)$ iff $F(s_j, \bar{b}, \bot)$ belongs to $\textsc{Can}(I')$. 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_δ 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 $\kappa : \Theta \to \{1, \ldots, n\}$ such that $\kappa(j) = i$ iff $S_J = R_i(\bar{x}_i, \bar{y}_i)$. The query Q is as follows:

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

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 $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$ iff $\text{certain}_{\mathcal{M}'}(Q', I') = \texttt{false}$. We only prove the direction from right to left, the other direction being analogous.

($\Leftarrow$) Let $\hat{J}'$ be the a solution for which $Q'(\hat{J}') = false$. Build the following function $h : \textsc{Can}(I) \to \textsc{Can}(I)$:

- $h(n) = 1$ if n is a null value in a tuple of the form $(b_1, \ldots, b_{r_i}, n)$ in R'_i, $i \in [1, n]$ 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' in $\hat{J}'$
- $h(n) = 0$ if n is a null value in a tuple of the form $(b_1, \ldots, b_{r_i}, n)$ in R'_i, $i \in [1, n]$ 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' in $\hat{J}'$
- $h(n) = 1$ if n is a null value in a tuple of the form $(b_1, \ldots, b_m, n)$ in F'_j, $j \in [1, \ell]$ and there is a tuple of the form $(s_j, b_1, \ldots, b_m, 1)$ in F' in $\hat{J}'$
- $h(n) = 0$ if n is a null value in a tuple of the form $(b_1, \ldots, b_m, n)$ in F'_j, $j \in [1, \ell]$ and there is a tuple of the form $(s_j, b_1, ..., b_m, 0)$ in F' in $\hat{J}'$
- $h(n) = n$ otherwise.

We prove that $Q(h(\text{CAN}(I))) = \texttt{false}$. Assume for the sake of contradiction that $Q_\alpha(h(\text{CAN}(I))) = \texttt{true}$. From the definition of Q_α, 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 $h(\text{CAN}(I))$. Let g be an homomorfism such that $g(\text{CAN}(I')) = \hat{J}'$. We let $\perp$ represent a null value such that $R'(c_i, b_1, \ldots, b_{r_i}, ..., g(\perp))$ belongs to $\hat{J}'$. 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 $Q(\hat{J}') = \texttt{true}$, we build a function $f : \{x_1, \ldots, x_p, y_1, \ldots, y_m, t_0, t_1, \ldots, t_{|\Theta|}, z_1, \ldots, z_{r_{\max}}, q, r, k, k', u, v, w, w', h_1, \ldots, h_{|\Theta|}\} \rightarrow dom(\hat{J}')$, and then show that the tuples $A'(f(q), f(t_0), f(t_1), \ldots, f(t_{|\Theta|}), f(k), f(k'), f(u))$, $R'(f(t_0), f(z_1), \ldots, f(z_{r_{\max}}), f(v))$, $F'(f(q), f(y_1), \ldots, f(y_m), f(n))$, $F'(f(q), f(y_1), \ldots, f(y_m), f(n))$,
$F'(f(k'), f(y_1), \ldots, f(y_m), f(w'))$, $Z'(f(q), f(w), f(w'), f(v))$, $G(f(q), f(w)')$;
$R'(f(t_{\tau(j)}), f(\bar{x}_{\kappa(j)}), f(\bar{y}_{\kappa(j)}), f(\bar{h}_{\tau(j)}, f(v))$ for $j \in \Theta$; and $V'_i(f(q), f(x_i))$ for $i \in [1, p]$ belong to $\hat{J}'$.

We define f as follows:

- $f(\mu) = b_j$ if $\mu = z_j$ for $j \in [1, r_{\max}]$
- $f(\mu) = b_1$ if $\mu = x_j$ for $j \in [1, p]$
- $f(\mu) = b_1$ if $\mu = y_j$ 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 $j \in [1, |\Theta|]$
- $f(t_0) = c_i$, $f(q) = s_0$, $f(u) = 1$, $f(n) = 0$
- $f(w) = f(w') = 0$
- $f(v) = g(\perp)$

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

- $A(s_0, c_i, d, \ldots, d, 1)$, obtained with (A.6).
- $R'(d, b_1, \ldots, b_1, g(\perp))$, obtained with (A.10).
- $V_1(s_0, b_1), \ldots, V_p(s_o, b_1)$, 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).
- $Z(s_0, 0, 0, g(\perp))$, obtained with (A.10).
- $G(s_0, 0)$, obtained with (A.5).

Next, assume for the sake of contradiction that $Q_\beta(h(\textsc{Can}(I))) = \texttt{true}$.

From the definition of Q_β, 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 $h(\textsc{Can}(I))$ and the tuple $(\bar{a}_x, \bar{a}_y, w)$ belongs to the interpretation of R'_i in $h(\textsc{Can}(I))$. Since $Q_\alpha(h(\textsc{Can}(I))) = \texttt{false}$, 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 $h(\textsc{Can}(I))$, there must be a tuple of the form $R'(c_i, \bar{a}_x, \bar{a}_y, ..., 1)$ in $\hat{J}'$, because otherwise it would contradict the definition of h. Let g be an homomorfism such that $g(\textsc{Can}(I')) = \hat{J}'$. In addition, it is easy to see that there is a null value $\perp$ and a tuple $F'(s_j, a_{i_1}, \ldots, a_{i_m}, \perp)$ in $\textsc{Can}(I')$. Again, by the definition of h, we obtain that $g(\perp) \neq 1$.

In order to show that $Q(\hat{J}') = \texttt{true}$, we build again a function $f : \{x_1, \ldots, x_p, y_1, \ldots, y_m, t_0, t_1, \ldots, t_{|\Theta|}, z_1, \ldots, z_{r_{\max}}, q, r, k, k', u, v, w, w', h_1, \ldots, h_{|\Theta 1|}\} \rightarrow dom(\hat{J}')$ Such that the tuples $A'(f(q), f(t_0), f(t_1), \ldots, f(t_{|\Theta|}), f(k), f(k'), f(u))$, $R'(f(t_0), f(z_1), \ldots, f(z_{r_{\max}}), f(v))$, $F'(f(q), f(y_1), \ldots, f(y_m), f(n))$, $F'(f(q), f(y_1), \ldots, f(y_m), f(n))$, $F'(f(k'), f(y_1), \ldots, f(y_m), f(w'))$, $Z'(f(q), f(w), f(w'), f(v))$, $G(f(q), f(w)')$; $R'(f(t_{\tau(j)}), f(\bar{x}_{\kappa(j)}), f(\bar{y}_{\kappa(j)}), f(\bar{h}_{\tau(j)}, f(v))$ for $j \in \Theta$; and $V_i'(f(q), f(x_i))$ for $i \in [1, p]$ belong to $\hat{J}'$.

We define f as follows:

- $f(\mu) = a_{i_k}$ 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 $k \in [1, r_{\max}]$
- $f(\mu) = a_1$ 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 μk, $\mu = k'$ or $\mu = t_k$ for $k \in [1, |\Theta|]$ such that $k \neq \tau(j)$
- $f(t_\tau(j)) = c_i$
- $f(t_0) = d$, $f(q) = s_j$, $f(u) = 1$, $f(v) = 1$
- $f(w) = f(w') = 0$
- $f(n) = g(\perp)$

Note that $f(v) \neq f(n)$ and $f(v) \neq f(u)$. In addition, from the definition of f we obtain that $F'(f(q), f(y_1), \ldots, f(y_m), f(n)) = F'(s_j, a_{i_1}, \ldots, a_{i_m}, g(\perp))$. It is also the case that $R'(f(t_{\tau(j)}), f(x_{\kappa(j)_1}), \ldots, f(x_{\kappa(j)_1}), f(y_{\kappa(j)_1}), \ldots, f(y_{\kappa(j)_1}), f(h_{\tau(j)_1}), \ldots, f(h_{\tau(j)_1}, f(v))$ $= R'(s_j, \bar{a}_x, \bar{a}_y, d, \ldots, d, 1)$.

It follows that $Q'(\hat{J}') = \texttt{true}$ since the following tuples also belong to $g(\textsc{Can}(I')) = \hat{J}'$:

- $A(s_j, d, d, \ldots, d, t_{\tau(j)}, d, \ldots, d, 1)$, 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)

- $V_1(s_j, b_1), \ldots, V_p(s_j, b_p)$, obtained with (A.8).
- $F(s_j, a_{i_1}, \ldots, a_{i_m}, g(\perp_1))$, obtained with (A.11).
- $F(d, a_{i_1}, \ldots, a_{i_m}, 0)$, obtained with (A.14).
- $z(s_j, 0, 0, 1)$, obtained with (A.7).
- $G(s_j, 0)$, obtained with (A.5).

Next, assume that $Q_\gamma(h(\textsc{Can}(I))) = \texttt{true}$. Let k be the index such that the evaluation of Q_γ^k is true in $h(\textsc{Can}(I))$. 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_γ^k is true in $h(\textsc{Can}(I))$, 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 $h(\textsc{Can}(I))$, and $v_1 \neq v_3$. The only tuples that belong to the interpretation of E' over $\textsc{Can}(I)$ 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_α and Q_β: Notice that the only tuples in the interpretation of F'_k and F'_g in $\textsc{Can}(I)$ that contain $(a_{k_1}, \ldots, a_{k_m})$ are $F'_k(a_{k_1}, \ldots, a_{k_m}, \perp_1)$ and $F'_g(a_{k_1}, \ldots, a_{k_m}, \perp_2)$, where $\perp_1$ and $\perp_2$ represent null values. It must be the case then that $h(\perp_2) = 1$, and $h(\perp_1) \neq 0$.

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 $\textsc{Can}(I')$ contains a tuple of the form $F'(s_k, a_{k_1}, \ldots, a_{k_m}, \perp_3)$. Let g be a homomorfism such that $g(\textsc{Can}(I')) = \hat{J}'$. From the definition of h, we obtain that $g(\perp_3) \neq 0$.

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

The function f is defined as follows:

- $f(\mu) = a_{k_i}$ if $\mu = y_i$ for $i \in [1, m]$
- $f(\mu) = a_i$ if $\mu = x_i$ for $i \in [1, p]$
- $f(\mu) = a_1$ if $\mu = z_i$ for $i \in [1, r_{\max}]$
- $f(\mu) = a_1$ if $\mu \in \bar{h}_i$ for $i \in [1, |\Theta|]$

- $f(\mu) = d$ 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 $Q'(\hat{J}') = \texttt{true}$ since the following tuples also belong to $g(\textsc{Can}(I')) = \hat{J}'$:

- $A(s_j, d, d, s_k, 0)$, obtained with (A.6).
- $R'(d, a_1, \ldots, a_1, 0)$, obtained with (A.12).
- $R'(d, f(y_{\kappa(j')_1}), \ldots, f(y_{\kappa(j')_1}), f(h_{\tau(j')_1}), \ldots, f(h_{\tau(j')_1}, 1))$ for every $j' \in [1, |\Theta|]$, obtained with (A.10)
- $V_1(s_j, b_1), \ldots, V_p(s_j, b_p)$, obtained with (A.8).
- $F(s_k, a_{i_1}, \ldots, a_{i_m}, g(\bot_3))$, obtained with (A.11).
- $F(s_g, a_{i_1}, \ldots, a_{i_m}, 1)$, from the definition of h.
- $F(d, a_{i_1}, \ldots, a_{i_m}, 0)$, 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 $Q_\delta(h(\textsc{Can}(I))) = \texttt{true}$. By the definition of Q_δ, there must be a tuple $(a_{1_1}, \ldots, a_{1_m})$ such that $F'_1(a_{1_1}, \ldots, a_{1_m}, 0)$ belongs to $h(\textsc{Can}(I))$. Then, by the definition of h, there is a tuple $F'(s_1, a_{1_1}, \ldots, a_{1_m}, 0)$ in $\hat{J}'$.

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

- $f(\mu) = a_{1_k}$ 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 $k \in [1, r_{\max}]$
- $f(\mu) = a_1$ if $\mu \in \bar{h}_k$ for $k \in [1, |\Theta|]$
- $f(\mu) = d$ if μk 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 $Q'(\hat{J}') = \texttt{true}$ since the following tuples also belong to $g(\textsc{Can}(I')) = \hat{J}'$:

- $A(s_f, d, d, s_1, 0)$, 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 $Q_\alpha(h(\textsc{Can}(I))) = \texttt{false}$, $Q_\beta(h(\textsc{Can}(I))) = \texttt{false}$, $Q_\gamma(h(\textsc{Can}(I))) = \texttt{false}$ and $Q_\delta(h(\textsc{Can}(I))) = \texttt{false}$, we obtain that the evaluation of Q over $h(\textsc{Can}(I))$, and then $\text{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$. $\square$