On the Relationship between Consistent Query Answering and Constraint Satisfaction Problems

Carsten Lutz$^1$ and Frank Wolter$^2$

1 Fachbereich Informatik, Universität Bremen, Germany
clu@uni-bremen.de
2 Department of Computer Science, University of Liverpool, UK
wolter@liverpool.ac.uk

Abstract

Recently, Fontaine has pointed out a connection between consistent query answering (CQA) and constraint satisfaction problems (CSP) [22]. We investigate this connection more closely, identifying classes of CQA problems based on denial constraints and GAV constraints that correspond exactly to CSPs in the sense that a complexity classification of the CQA problems in each class is equivalent (up to FO-reductions) to classifying the complexity of all CSPs. We obtain these classes by admitting only monadic relations and only a single variable in denial constraints/GAVs and restricting queries to hypertree UCQs. We also observe that dropping the requirement of UCQs to be hypertrees corresponds to transitioning from CSP to its logical generalization MMSNP and identify a further relaxation that corresponds to transitioning from MMSNP to GMSNP (also known as MMSNP$_2$). Moreover, we use the CSP connection to carry over decidability of FO-rewritability and Datalog-rewritability to some of the identified classes of CQA problems.

1998 ACM Subject Classification H.2.4 [Systems]: Relational databases

Keywords and phrases Consistent Query Answering, Constraint Satisfaction, Data Complexity, Dichotomies, Rewritability

Digital Object Identifier 10.4230/LIPIcs.ICDT.2015.363

1 Introduction

In modern applications of database systems, it cannot always be guaranteed that the data is consistent with the relevant integrity constraints; for example, inconsistency occurs easily when the data is extracted from the web or integrated from multiple sources. A prominent approach to address this problem is consistent query answering (CQA) as introduced in [3] where one returns the certain answers over all minimal repairs of the inconsistent database, see also the surveys [7, 15, 41]. Since the data complexity of CQA can be coNP-complete or higher [2, 14, 16, 38], CQA is in general significantly harder than traditional query answering. This observation has resulted in a lot of research activity aiming to more precisely clarify the computational complexity of CQA, separating in particular the easy cases from the hard ones. Here, ‘easy’ might mean different things. The ideal result is that a CQA problem CQA$(C, q)$, defined by a set $C$ of integrity constraints and a query $q$, is rewritable into a first-order logic (FO) query $\hat{q}$ and thus answers can be computed by a classical RDBMS and in $AC_0$ data complexity [24, 39]. If FO-rewritability is not attainable, one might at least hope for Datalog-rewritability or PTime data complexity. Ultimate goals of this research programme would be to classify the exact complexity of every CQA problem and, closely related, to decide for a given CQA problem whether it is easy in some relevant sense, say whether it admits an FO-rewriting. Completely classifying the border between PTime and coNP...
prominently involves solving dichotomy questions: for some important classes of integrity constraints and queries, it has been conjectured that CQA is in PTime or coNP-hard for every problem in the class [1, 40, 41]. In fact, with today’s methods one cannot hope to completely classify the complexity of a class of CQA problems if that class does not have such a dichotomy.

Despite serious efforts, comprehensive results for dichotomies in CQA have so far been elusive. For several restricted cases which all require that the query to be answered is free of self-joins, dichotomies were obtained in [27, 40, 28]. An explanation of why general dichotomy results for CQA is difficult to obtain was recently given by Fontaine [22], who linked CQA with the area of constraint satisfaction problems (CSPs). CSPs constitute a subclass of NP that contains many relevant NP-complete problems such as 3SAT, 3COL, and integer programming over bounded domains, and it is widely believed that this class is computationally more well-behaved than NP itself. In particular, a long-standing conjecture due to Feder and Vardi states that CSPs enjoy a dichotomy between PTime and NP [20]. Massive research efforts in logic, complexity, and algebra have been directed towards proving this conjecture, and although steady progress has been made, the conjecture is still open. In contrast, strong results have been obtained on the FO-definability and Datalog-definability of CSPs, the counterpart of FO- and Datalog-rewritability in CQA: while both problems are undecidable for the entire class NP, they are decidable and NP-complete for CSPs [30, 5, 23].

Fontaine’s main result in [22] links the Feder-Vardi conjecture to dichotomies in CQA under GAV constraints by showing that for every CSP problem CSP(A) defined by some template A, there is a CQA problem CQA(C, q) with C a set of GAV constraints and q a union of conjunctive queries (UCQ) such that CQA(C, q) and the complement coCSP(A) of CSP(A) are PTime-equivalent, that is, they have the same complexity up to PTime-reductions. Consequently, establishing a dichotomy between PTime and coNP for CQA with GAV constraints and UCQs implies the Feder-Vardi conjecture. The aim of this paper is to study the CSP-CQA connection in more detail, focussing on denial constraints and GAV constraints which have both received significant attention in CQA [17, 16, 8, 2, 37, 38, 22]. In particular, we aim to identify classes of CQA problems that exactly correspond to the class of CSPs in the sense that classifying the complexity of problems from both classes is equivalent in a strong sense.

Our first main observation is that CSPs correspond to CQA problems whose (denial or GAV) constraints involve only monadic relation symbols and only a single variable and in which the query to be answered is a UCQ in which all CQs take the form of a hypertree (all queries in this paper are Boolean). More specifically, let a monadic disjointness constraint (MDiC) be of the form \( \forall x \rightarrow (P_1(x) \land \cdots \land P_n(x)) \) and a monadic GAV constraint (MGAV) be of the form \( \forall x (P_1(x) \land \cdots \land P_n(x) \rightarrow Q(x)) \). We use (MDiC, tUCQ) to denote the class of all CQA problems whose constraints are MDiCs and whose query is a UCQ in which every CQ is a hypertree in the sense that its incidence graph is a tree (without multi-edges), and likewise for (MGAV, tUCQ). We then show that (i) for every CSP(A) there is a problem CQA(C, q) in (MDiC, tUCQ) such that coCSP(A) and CQA(C, q) are FO-equivalent (that is, they have the same complexity up to FO-reductions), (ii) for every problem in (MDiC, tUCQ) there is one from (MGAV, tUCQ) that is FO-equivalent, and (iii) for every problem CQA(C, q) in (MGAV, tUCQ) there is a CSP(A) such that CQA(C, q) and coCSP(A) are FO-equivalent. This improves upon the main result of [22] in several ways. First, the CQA problems constructed in [22] involve constraints that contain non-monadic relations and the UCQs used there are not hypertrees since they include multi-edges; we thus identify much more restricted CQA classes whose complexity classification is already as hard
as classifying CSPs; second, we also translate from CQA to CSP and thus show that a PTime vs. coNP dichotomy for (MDiC, tUCQ) and (MGAV, tUCQ) is equivalent to the Feder-Vardi conjecture (instead of only implying it); and third, we replace PTime-equivalence with FO-equivalence which (a) means that not only a PTime vs. (co)NP dichotomy carries over, but any complexity classification that involves complexity classes closed under FO-reductions (such as LogSpace), and (b) enables us to transfer results from CSP to the classes of CQA problems that we have identified.

Regarding (b), we show that for (MDiC, tUCQ) and (MGAV, tUCQ), rewritability into FO and into Datalog are decidable, referring to the version of Datalog which allows negation in front of body atoms that use a monadic EDB relation. Our approach yields NExpTime upper bounds for these problems and we demonstrate that the complexity is indeed high (despite the fact that we deal with rather restricted CQA problems) by establishing a PSPACE lower bound for FO-rewritability, leaving the exact complexity open. We also transfer Bulatov’s result that CSPs whose templates have at most three elements enjoy a dichotomy between PTime and NP [13] to CQA, identifying a (rather restricted) corresponding fragment of (MDiC, tUCQ) that has a dichotomy between PTime and coNP.

We then investigate the effect of dropping the requirement that queries are hypertrees while maintaining all restrictions on integrity constraints, which gives rise to the classes of CQA problems (MDiC, UCQ) and (MGAV, UCQ). We show that this corresponds to transitioning from CSP to its logical generalization MMSNP, which was introduced by Feder and Vardi when studying the descriptive complexity of CSP [20] and has received considerable interest, see e.g. [33, 11]. More precisely, we establish results that exactly parallel (i) to (iii) above, replacing tUCQs with UCQs and CSP with MMSNP. Again, all reductions are FO-reductions. The known results that CSP ⊆ MMSNP and that for every MMSNP problem there is a CSP problem that is PTIME-equivalent [20, 29] then also yields that (MDiC, UCQ) has a dichotomy between PTime and coNP if and only if this is the case for (MDiC, tUCQ), and likewise for the corresponding CQA classes based on MGAVs. This does not imply, though, that a full complexity classification of these classes is equivalent since the relation between CSP and MMSNP in terms of FO-reductions is open. These results also shed some light on the decidability of FO- and Datalog-rewritability in (MDiC, UCQ) and (MGAV, UCQ), which is equivalent to the decidability of FO-definability of MMSNP problems, an open problem. Finally, we generalize (MDiC, UCQ) by giving up the restriction that integrity constraints are monadic and comprise only a single variable, instead requiring that every atom in the integrity constraint comprises the same variables in the same order. We then show that this corresponds to the transition from MMSNP to GMSNP [10] (also known as MMSNP² [32]), which means to replace monadicity as stipulated in MMSNP with a guardedness condition.

Some proof details are deferred to the appendix of the long version of this paper available at http://www.informatik.uni-bremen.de/tdki/research/papers.html.

2 Preliminaries

A schema is a finite collection $S = (S_1, \ldots, S_k)$ of relation symbols with associated non-zero arity. A fact over $S$ is an expression of the form $S(a_1, \ldots, a_n)$ where $S \in S$ is an $n$-ary relation symbol, and $a_1, \ldots, a_n$ are elements of some fixed, countably infinite set const of constants. An instance $I$ over $S$ is a finite set of facts over $S$. The active domain $\text{adom}(I)$ of $I$ is the set of all constants that occur in the facts of $I$. We will frequently use boldface notation for tuples, such as in $a = a_1 \cdots a_n$. 
A conjunctive query (CQ) takes the form \( q = \exists x \varphi(x) \) where \( \varphi \) is a conjunction of relational atoms, neither constants nor equality allowed. A union of conjunctive queries (UCQ) is a disjunction of CQs. Note that we consider only Boolean queries for simplicity, see the conclusion for some further remarks on this issue.

A denial constraint (DC) has the form \( \forall x \neg \varphi(x) \), where \( \varphi \) is a conjunction of relational atoms. A global as view constraint (GAV) takes the form \( \forall x \varphi(x) \rightarrow S(x) \) where \( \varphi \) is a conjunction of relational atoms. Let \( I \) be an instance and \( C \) a set of constraints. An instance \( J \) is a minimal repair of \( I \) w.r.t. \( C \) if \( J \) satisfies all constraints in \( C \) and there is no instance \( J' \) such that \( J' \) satisfies all constraints in \( C \) and \( I \Delta J' \subseteq I \Delta J \), where ‘\( \Delta \)’ denotes symmetric difference. We generally omit ‘w.r.t. \( C \)’ when \( C \) is clear from the context. For a query \( q \), we write \( I \models C q \) if every minimal repair \( J \) of \( I \) satisfies \( J \models q \).

A consistent query answering (CQA) problem, denoted \( \text{CQA}(C,q) \), is defined by a set of constraints \( C \) and a query \( q \). As input, an \( S \)-instance \( I \) is given where \( S \) is the set of relation symbols used in \( C \) or \( q \). The question is whether \( I \models C q \). We use (DC, UCQ) to denote the set of problems \( \text{CQA}(C,q) \) where \( C \) is a set of DCs and \( q \) a UCQ, and likewise for (GAV, UCQ) and other combinations of constraint language and query language.

In this paper, all considered decision problems take instances over some fixed schema as inputs. For two such decision problems \( P_1, P_2 \), we write \( P_1 \preceq_{\text{FO}} P_2 \) if \( P_1 \) reduces to \( P_2 \) by a polynomial time reduction. We write \( P_1 \preceq_{\text{P}} P_2 \) if \( P_1 \) reduces to \( P_2 \) by an \( \text{FO-reduction} \), defined as in [26]. However, most of our \( \text{FO-reductions} \) are of the following simple form.

For problems \( P_1 \) and \( P_2 \) over schemas \( S_1 \) and \( S_2 \), a map \( T \) that assigns to each \( S_1 \)-instance \( I \) an \( S_2 \)-instance \( T(I) \) is \( \text{FO-definable} \) if for every \( k \)-ary relation symbol \( R \) in \( S_2 \), there is an \( \text{FO-formula} \ \varphi_R \) over \( S_1 \) (equality and constants allowed) with \( k \) free variables such that \( R(a) \in T(I) \) if and only if \( \varphi_R[a] \), for all \( a \) in \( \text{adom}(I) \). Such a map gives rise to an \( \text{FO-reduction} \) of \( P_1 \) to \( P_2 \) if for all \( S_1 \)-instances \( I \), we have \( I \in P_1 \) iff \( T(I) \in P_2 \). \( \text{FO-reductions} \) of this simple form differ from the general case in that (i) no arithmetic operations are admitted in the formulas \( \varphi_R \) and (ii) the domain of the \( T(I) \) cannot be larger than the domain of \( I \). With \( \text{FO-equivalence} \) and \( \text{PTime-equivalence} \) of two problems \( P_1 \) and \( P_2 \), denoted \( P_1 \approx_{\text{FO}} P_2 \) and \( P_1 \approx_{\text{P}} P_2 \), we mean that there are reductions between \( P_1 \) and \( P_2 \) in both directions. For two classes of decision problems \( C_1 \) and \( C_2 \), we write \( C_1 \preceq_{\text{FO}} C_2 \) if for every problem \( p_1 \in C_1 \), there is a problem \( p_2 \in C_2 \) such that \( p_1 \preceq_{\text{FO}} p_2 \) and \( p_2 \preceq_{\text{FO}} p_1 \). We write \( C_1 \approx_{\text{FO}} C_2 \) and say that \( C_1 \) and \( C_2 \) are \( \text{FO-equivalent} \) if \( C_1 \preceq_{\text{FO}} C_2 \) and \( C_2 \preceq_{\text{FO}} C_1 \). The definition of \( C_1 \preceq_{\text{P}} C_2 \) and \( C_1 \approx_{\text{P}} C_2 \) is analogous, but based on polynomial time reductions.

Let \( A \) be an instance over schema \( S \). The constraint satisfaction problem \( \text{CSP}(A) \) is to decide, given an instance \( I \) over \( S \), whether there is a homomorphism from \( I \) to \( A \), which we denote with \( I \rightarrow A \). In this context, \( A \) is called the template of \( \text{CSP}(A) \). We will generally and w.l.o.g. assume that the template is a core, that is, every automorphism is an isomorphism. It is often useful to further assume that the template \( A \) admits precoloring,\(^1\) that is, for each \( a \in \text{adom}(A) \), there is a unary relation symbol \( P_a \in S \) such that \( P_a(b) \in A \) iff \( b = a \) [18]. It is known that for every template \( A \) (which is a core), there is a template \( A' \) that admits precoloring such that \( \text{CSP}(A) \approx_{\text{FO}} \text{CSP}(A') \) [31]. We use \( \text{coCSP}(A) \) to denote the complement problem of \( \text{CSP}(A) \) and \( \text{coCSP} \) to denote the set of all problems \( \text{coCSP}(A) \) whose template \( A \) admits precoloring.

The logic MMSNP was introduced by Feder and Vardi as a descriptive complexity counterpart of CSPs [20]. Since we will mostly be concerned with the complement of MMSNP, we refrain from giving the original definition and directly introduce its complement.

---

\(^1\) This property is also known as ‘full idempotence’ and ‘pointedness’. 
which can conveniently be defined as (negation-free) monadic disjunctive Datalog [10]. A monadic disjunctive Datalog (MDDLog) rule \( \rho \) has the form \( R_1(y_1) \land \cdots \land R_n(y_n) \rightarrow S_1(x_1) \lor \cdots \lor S_m(x_m) \) with \( m \geq 0 \), \( n > 0 \), and \( S_1, \ldots, S_m \) monadic. Every variable that occurs in the head \( S_1(x_1) \lor \cdots \lor S_m(x_m) \) of \( \rho \) is also required to occur in \( \rho \)'s body \( R_1(y_1) \land \cdots \land R_n(y_n) \).

Empty rule heads are denoted \( \bot \). An MDDLog program \( P \) is a finite set of MDDLog rules with a selected nullary goal relation \( \text{goal} \) that does not occur in rule bodies and only in goal rules of the form \( R_1(x_1) \land \cdots \land R_n(x_n) \rightarrow \text{goal}() \). Relation symbols that occur in the head of at least one rule of \( P \) are intensional (IDB), and all remaining relation symbols in \( P \) are extensional (EDB). Every MDDLog program \( P \) defines a decision problem: given an \( S \)-instance \( I \), where \( S \) is the set of EDB relations in \( I \), decide whether \( I \models P \), that is, whether in every extension of \( I \) that satisfies all non-goal rules in \( P \), the body of at least one goal rule applies. We use \( \text{coMMSNP} \) to denote the class of all these decision problems and MMSNP to denote the class of their complements.

3 Relating CQA and CSP

We identify fragments of (DC, UCQ) and (MGAV, UCQ) that are FO-equivalent to \( \text{coCSP} \). They involve restrictions on the admitted constraints as well as on the admitted queries. A denial constraint is called a monadic disjointness constraint (MDiC) if all relation symbols that occur in it are monadic and it contains only a single variable; a GAV is called monadic or an MGAV if it satisfies the same conditions. MDiCs and MGAVs are clearly rather restricted classes of constraints. Note, however, that they are useful for speaking about type systems. For example, the MDiC \( \forall x \neg (\text{person}(x) \land \text{process}(x)) \) asserts disjointness of the types \text{person} and \text{process} while the MGAV \( \forall x (\text{professor}(x) \rightarrow \text{person}(x)) \) ensures a subsumption between the types \text{professor} and \text{person}.

While we restrict constraints to MDiCs and MGAVs, UCQs are required to contain only CQs that have the shape of a hypertree. The incidence graph of a CQ \( q \) is the bipartite undirected graph whose nodes are the variables and the atoms in \( q \) and whose edges connect a variable \( x \) with each atom \( R(x) \) such that \( x \in x \). A CQ is a hypertree conjunctive query (tCQ) if its incidence graph is a tree (without multi-edges). A hypertree UCQ (tUCQ) is a UCQ in which every CQ is a tCQ.

Before proceeding, we observe the following connection between MDiCs and MGAVs.

Lemma 1. ([MDiC, Q] \( \preceq_{\text{FO}} \) (MGAV, Q) for Q \( \in \) \{UCQ, tUCQ\}). Moreover, given a problem CQA(\( C, q \)) from (MDiC, Q) one can construct a problem CQA(\( C', q' \)) in (MGAV, Q) such that CQA(\( C, q \)) \( \equiv_{\text{FO}} \) CQA(\( C', q' \)) in polynomial time.

Proof. Let CQA(\( C, q \)) be a problem from (MDiC, Q) over schema S. Define \( C' = \{ \forall x \varphi(x) \rightarrow M(x) \mid \forall x \neg \varphi(x) \in C \} \) where \( M \) is a fresh monadic relation symbol and \( q' = q \lor \exists x M(x) \).

We show that CQA(\( C, q \)) \( \preceq_{\text{FO}} \) CQA(\( C', q' \)) and vice versa. For the former, we observe that for all \( S \)-instances \( I \), we have \( I \models C \ q \) iff \( I \models C' \ q' \). For the latter, it is easy to show that, for all \( S \cup \{ M \} \)-instances \( I \), we have \( I \models C' \ q' \) iff \( I \models \exists x M(x) \) or \( I \models C \ q \). Clearly, these reductions can be implemented as FO-reductions.

Our aim is to show that (MDiC, tUCQ) \( \preceq_{\text{FO}} \) (MGAV, tUCQ) \( \preceq_{\text{FO}} \) coCSP. By Lemma 1, it suffices to show that coCSP \( \preceq_{\text{FO}} \) (MDiC, tUCQ) and (MGAV, tUCQ) \( \preceq_{\text{FO}} \) coCSP. We start with the former, improving upon a reduction by Fontaine [22] which shows that coCSP \( \preceq_{\text{p}} \) (GAV, UCQ). Consider CSP(\( A \)) over schema S where A admits precoloring. We
On the Relationship between CQA and CSP

construct a problem $\text{CQA}(C_A, q_A)$ from ($\text{MDiC}, \text{tUCQ}$) over schema $S'$ which extends $S$ with unary relation symbols $Q_a$, $a \in \text{adom}(A)$ (these symbols should be distinguished from the monadic relation symbols $P_a$ in $S$, $a \in \text{adom}(A)$, which exist since $A$ admits precoloring). $C_A$ contains one monadic disjointness constraint:

$$\forall x \rightarrow \bigwedge_{a \in \text{adom}(A)} Q_a(x).$$

For each $a \in \text{adom}(A)$, we use $\text{con}_a(x)$ to denote the conjunction $\bigwedge_{o \in \text{adom}(A) \setminus \{a\}} Q_o(x)$. The tUCQ $q_A$ contains the following tCQ for each $R$ of arity $n$ and each $a = a_1 \cdots a_n \in \text{adom}(A)^n$ such that $R(a) \not\in A$:

$$\exists x_1 \cdots \exists x_n (\text{con}_{a_1}(x_1) \land \cdots \land \text{con}_{a_n}(x_n) \land R(x_1, \ldots, x_n)).$$

To understand the construction of $\text{CQA}(C_A, q_A)$, consider the reduction from CSP($A$) to the complement of $\text{CQA}(C_A, q_A)$). Given an $S$-instance $I$ that is an input to CSP($A$), we construct an $S'$-instance $T^i(I)$ by adding $Q_a(b)$ for all $b \in \text{adom}(I)$ and all $a \in \text{adom}(A)$. Then each minimal repair $J$ of $T^i(I)$ must satisfy, for each $b \in \text{adom}(I)$, $J \models \text{con}_b[b]$ for a unique $a \in \text{adom}(A)$. In this way, $J$ represents a function $h$ that assigns to each $b \in \text{adom}(I)$ the unique $a \in \text{adom}(A)$ such that $Q_a(b) \not\in \mathcal{J}$. If $J \not\models q_A$, then $h$ must clearly be a homomorphism. Indeed, we show in the appendix that $I \rightarrow A$ iff $T^i(I) \not\models q_A$.

Lemma 2. $\text{coCSP}(A) \preceq_{\text{FO}} \text{CQA}(C_A, q_A)$ and $\text{CQA}(C_A, q_A) \preceq_{\text{FO}} \text{coCSP}(A)$.

Proof. The first reduction was already described above. Clearly, $T^i$ is FO-definable and thus the reduction is an FO-reduction. For the converse reduction, let $I$ be an $S'$-instance. Denote by $X$ the set of $b \in \text{adom}(I)$ such that there are at least two distinct $a_1, a_2 \in \text{adom}(A)$ with neither $Q_{a_1}(b) \in \mathcal{I}$ nor $Q_{a_2}(b) \in \mathcal{I}$. Then $T^i(I)$ is obtained from $I$ by dropping all facts that involve a constant from $X$ or a relation symbol in $S' \setminus S$, and adding all facts $P_a(b)$ such that $Q_e(b) \in \mathcal{I}$ iff $e \neq a$ for all $e \in \text{adom}(A)$. Note that it is crucial here that $A$ admits precoloring as we use the relation symbols $P_a$. Clearly $T^i(I)$ is FO-definable. We show in the appendix that $I \not\models q_A$ iff $T^i(I) \rightarrow A$.

It is easy to see that, given $A$, we can construct $\text{CQA}(C_A, q_A)$ in polynomial time. We have thus established the following.

Theorem 3. For every CSP template $A$ that admits precoloring, there is a problem $\text{CQA}(C, q)$ from ($\text{MDiC}, \text{tUCQ}$) that satisfies $\text{CQA}(C, q) \approx_{\text{FO}} \text{coCSP}(A)$ and can be constructed in polynomial time.

We now show that ($\text{MGAV}, \text{tUCQ}$) $\preceq_{\text{FO}} \text{coCSP}$. Let $\text{CQA}(C, q_0)$ be over schema $S$ with $C$ a set of MGAVs and $q_0$ a hyper-tree UCQ. We use $S^{(1)}$ to denote the restriction of $S$ to monadic relation symbols and $S^{(>1)}$ for the restriction of $S$ to non-monadic symbols. In the following, we define a CSP template $A_{C\cdot q_0}$ over schema $S' = S^{(>1)} \cup \{P_\Gamma \mid \Gamma \subseteq S^{(1)}\}$. Note that there is an obvious natural translation of $S$-instances into corresponding $S'$-instances and vice versa; for example, an $S'$-instance $I$ is translated to an $S$-instance $J$ by replacing every fact $P_\Gamma(a)$ with $P(a)$ for all $P \in \Gamma$. The change of schema is used to deal with the complication that the ‘yes’-instances of each CSP are closed under homomorphic pre-images while the ‘no’-instances of CQA problems from ($\text{MGAV}, \text{tUCQ}$) are not (unless the schema is modified in the described way). For any set $\Gamma \subseteq S^{(1)}$, we use $\text{rep}(\Gamma)$ to denote the set of all sets $\Gamma' \subseteq S^{(1)}$ such that $\{P(a) \mid P \in \Gamma'\}$ is a minimal repair of the instance $\{P(a) \mid P \in \Gamma\}$.
Let Q be the set of all connected subqueries of CQs in qo (which are again hypertrees) and of all queries of the form P(x), P ∈ S\lceil 1\rceil. A place is a pair (q, x) with q ∈ Q and x ∈ var(q). A type t is a set of places. The type tpₐ(a) realized by constant a in instance I is the set of all places (q, x) such that there is a homomorphism h from q to I that takes x to a. We say that a type t is realizable if there is an instance I such that I satisfies all constraints in C and t is realized by some constant in I; we say that t avoids q0 if (q, x) ∉ t for any disjunct q of q0 and x ∈ var(q). For R ∈ S of arity n, we say that a tuple (t₁, . . . , tₙ) of types is R-coherent if for any instance I and tuples of constants (a₁, . . . , aₙ) such that tpₜ(aᵢ) = ti for 1 ≤ i ≤ n, after adding to I the fact R(a₁, . . . , aₙ), we still have tpₜ(aᵢ) = ti for 1 ≤ i ≤ n. Now, the template AC,q₀ is defined as follows:

= the constants in AC,q₀ are the pairs ⟨t, Γ⟩ with t a realizable type that avoids q₀ and Γ ⊆ S\lceil 1\rceil such that t\lceil Γ\rceil is the restriction of t to schema S\lceil 1\rceil;
= AC,q₀ contains all facts of the form Pₜ((t, Γ));
= AC,q₀ contains all facts R((t₁, Γ₁), . . . , (tₙ, Γₙ)), R of arity n, if (t₁, . . . , tₙ) is R-coherent.

To understand the construction, consider the reduction from the complement of CQA complexity of CSPs (up to FO reductions).

Lemma 4. CQA(C, q₀) ≤_FO coCSP(AC,q₀) and coCSP(AC,q₀) ≤_FO CQA(C, q₀).

Proof. The first reduction was described above. Correctness is proved in full detail in the appendix. For the second reduction, let an S’-instance I be given. If there exists a ∈ adom(I) with Pₜ((t, Γ)) for all (P(x), x) ∈ t is a universal minimal repair in the following sense: (i) it is a minimal repair of the S-instance that corresponds to AC,q₀ and (ii) any minimal repair of any S-instance homomorphically maps to Iₜ.

Theorem 5. For every problem CQA(C, q) from (MGAV, tUCQ), there is a CSP template A that satisfies coCSP(A) ≈_FO CQA(C, q) and can be constructed in single exponential time.

Summarizing Theorems 3 and 5, we thus obtain the following FO-equivalences.

Corollary 6. (MDIC, tUCQ) ≈_FO (MGAV, tUCQ) ≈_FO coCSP.
FO- and Datalog-Rewritability

We exploit the reductions from the previous section and recent results concerning the FO- and Datalog-definability of CSPs to show that FO-rewritability and Datalog-rewritability are decidable in (MDic, UCQ) and (MGAV, tUCQ). A problem CQA(C, q) over schema S is FO-rewritable if there is a Boolean FO-query \( \hat{q}_{C,q} \) such that for all S-instances I, we have \( I \models q \) iff \( I \models \hat{q}_{C,q} \). Thus, FO-rewritability ensures that CQA(C, q) can be implemented using a conventional RDBMS. Datalog-rewritability is defined accordingly, where we refer to the version of Datalog that admits negated monadic EDB atoms in rule bodies. Without such atoms, Datalog-rewritability would be an extremely elusive property, as illustrated by the trivial problem CQA(C, q) where \( C = \{ \forall x (A_1(x) \land A_2(x)) \} \) and \( q = \exists x A_1(x) \), which can be rewritten into \( A_1(x) \land \neg A_2(x) \to \text{goal()} \), but not into any Datalog program without negated monadic EDB atoms. Note that in contrast to consistent query answering problems, a class coCSP(A) is Datalog-definable with negated EDB atoms in rule bodies if and only if it is definable by a negation-free Datalog program [21]. We rely on the following results from [30, 5, 23].

**Theorem 7.** Given a CSP template A, it is NP-complete to decide whether coCSP(A) is FO-definable. The same is true for Datalog-definability.

Since the reductions between CQA(C, q) and coCSP(AC,q) used in the proof of Theorem 5 are FO-reductions, it follows immediately that CQA(C, q) is FO-rewritable if and only if coCSP(AC,q) is FO-definable [26]. Given that AC,q can be constructed in single exponential time, we obtain a NExpTime upper bound for deciding FO-rewritability of CQA problems in (MDic, tUCQ) and (MGAV, tUCQ).

**Theorem 8.** Given (C, q) such that CQA(C, q) is from (MDic, tUCQ) or (MGAV, tUCQ), it is decidable in NExpTime whether CQA(C, q) is FO-rewritable. Both problems are PSpace-hard.

**Proof (Sketch).** The PSpace lower bounds are proved in the appendix by a non-trivial reduction of the word problem of polynomially space-bounded Turing machines. Similar reductions have been used to establish PSpace-hardness of boundedness in linear monadic datalog [19] and of certain FO-rewritability problems in ontology-based data access [9]. The general idea is to start with a DTM M that solves a PSpace-complete problem and to first modify it so that M terminates when started in any configuration (which must not even be reachable from an initial configuration). Then, one crafts a problem CQA(C, q) such that if M accepts an input x, then any FO-rewriting of CQA(C, q) would have to query for the existence of unboundedly long paths of facts that represent an accepting computation of M on x, repeated over and over again. Clearly, this contradicts the locality of FO-queries. For technical reasons, we actually cannot ensure that the first computation on the mentioned paths starts with the initial configuration for x. However, M terminates from any configuration, and the second computation on the paths (as well as all subsequent ones) are guaranteed to start with the initial configuration for x. Details are given in the appendix. We note that C consists only of a single constraint, which is of the form \( \forall x \neg(B(x) \land B'(x)) \).

The exact complexity remains open, but we speculate that the problems in Theorem 8 are at least ExpTime-hard. To obtain complexity bounds for Datalog-rewritability, we inspect the FO-formulas required to define \( T^1(I) \) and \( T^4(I) \) in the proof of Lemma 4.

**Lemma 9.** Let CQA(C, q0) and AC,q be as in Lemma 4. Then CQA(C, q0) is Datalog-rewritable iff coCSP(AC,q) is Datalog-definable.
Proof. Let $\Pi$ be a Datalog program that defines $\text{coCSP}(A_{C,q_0})$. Replace in $\Pi$ all body atoms $P_I(x)$ with $\bigwedge_{P \in \Gamma} P(x) \land \bigwedge_{P \in \Gamma} \neg P(x)$ and denote the resulting program by $\Pi'$. It can be verified using the proof of Lemma 4 that $I \models_{C,q_0} I'$ for all $\mathbf{S}$-instances $I$. Thus $\Pi'$ is a Datalog-rewriting of $\text{CQA}(C,q_0)$.

Conversely, assume that $\Pi$ is a Datalog-rewriting of $\text{CQA}(C,q_0)$. Replace every rule $\rho$ in $\Pi$ by a set of rules as follows. For every variable $x$ in $\rho$, replace the set $S$ of conjuncts in the body of $\rho$ that are of the form $P(x)$ and $\neg P(x)$, $P \in \mathbf{S}^{(1)}$, by any atom $P_I(x)$ such that $\rho \subseteq \{ P(x) \mid P \in \Gamma \} \cup \{ \neg P(x) \mid P \in \mathbf{S}^{(1)} \setminus \Gamma \}$. Moreover, add $P_I(x) \land P_S(x) \rightarrow \text{goal}()$ as a new rule for all $\Gamma, \Lambda \subseteq \mathbf{S}^{(1)}$ with $\Gamma \neq \Lambda$. Denote the resulting program by $\Pi'$. It can be verified using the proof of Lemma 4 that $\Pi'$ defines $\text{coCSP}(A_{C,q_0})$.

The following is a consequence of Lemma 4, Lemma 9, and Theorem 7.

**Theorem 10.** Given $(C,q)$ such that $\text{CQA}(C,q)$ is from $(\text{MDiC}, \text{tUCQ})$ or $(\text{MGAV}, \text{tUCQ})$, it is decidable in $\text{NEXP}$ whether $\text{CQA}(C,q)$ is Datalog-rewritable.

We do not have any non-trivial lower bound. Finally, we observe that, thanks to allowing negation in Datalog-rewritings as described above, FO-rewritability implies Datalog-rewritability.

**Theorem 11.** In $(\text{MDiC}, \text{tUCQ})$ and $(\text{MGAV}, \text{tUCQ})$, FO-rewritability implies (non-recursive) Datalog-rewritability.

Proof. It was observed by Atserias (and follows from Rossman’s homomorphism preservation theorem) that for any CSP template $A$, $\text{coCSP}(A)$ being FO-definable implies that $\text{coCSP}(A)$ is UCQ-definable [4, 35]. Thus FO-rewritability of $\text{CQA}(C,q)$ implies FO-definability of $\text{coCSP}(A_{C,q})$. The latter implies UCQ-definability of $\text{coCSP}(A_{C,q})$ which implies (non-recursive) Datalog-rewritability of $\text{CQA}(C,q)$. The latter implies (non-recursive) Datalog-rewritability of $\text{CQA}(C,q)$.

## 5 A Dichotomy Result

For restricted classes of CSPs, a dichotomy between $\text{PTime}$ and $\text{NP}$ has been established. The most notable results include Schaefer’s famous dichotomy theorem for templates with at most two elements [36] and its generalization to three element templates obtained much later by Bulatov [13]. It is natural to ask whether these dichotomies transfer to restricted classes of CQA problems. In this section, we identify a subclass of $(\text{MDiC}, \text{tUCQ})$ for which this is the case, thus obtaining a dichotomy between $\text{PTime}$ and $\text{coNP}$.

We consider the class of problems $\text{CQA}(C,q_0)$ such that $C$ consists of a single MDiC of the form $\forall x \exists y (A_1(x) \land A_2(y))$ and $q_0$ is a tUCQ such that, in every tCQ in $q_0$, there is at most one atom with a relation symbol distinct from $A_1, A_2$. Let us call a problem of this form a restricted binary CQA problem where ‘binary’ refers to the number of atoms allowed in the MDiC. Note that the resulting class $\text{r2CQA}$ of CQA problems is not trivial since a straightforward analysis of the proof of Theorem 3 shows that $\text{2coCSP} \preceq_{\text{FO}} \text{r2CQA}$, where $\text{icoCSP}$ is the class of complements of all problems that can be defined by a CSP template with at most $i$ elements. Consequently, establishing a $\text{PTime} / \text{coNP}$ dichotomy for $\text{r2CQA}$ implies Schaefer’s theorem; we actually find it remarkable that a class of CQA problems as simple as $2\text{rCQA}$ turns out to be that complex. In the following, we show that $\text{r2CQA} \preceq_{\text{FO}} \text{3coCSP}$ and thus obtain a dichotomy between $\text{PTime}$ and $\text{coNP}$ for $\text{r2CQA}$ by Bulatov’s result.
Let CQA($C, q_0$) be a restricted binary CQA problem over schema $S$. We define a CSP template $A_{C,q_0}$ over schema $S'$ as $S \setminus \{A_1, A_2\}$ as follows:

1. the constants in $A_{C,q_0}$ are 0, 1, 2;
2. $A_{C,q_0}$ contains the facts $P_0(0)$, $P_{(A_1)}(1)$, $P_{(A_2)}(2)$, $P_{(A_1,A_2)}(1)$, and $P_{(A_1,A_2)}(2)$;
3. $A_{C,q_0}$ contains the facts $R(i_1, \ldots, i_k), R \in S \setminus \{A_1, A_2\}$ and $i_j \in \{1, 2\}$, when $q_0$ does not evaluate to true on the $S$-instance $(R(i_1, \ldots, i_k)) \cup \{A_{i_j}(i_j) \mid 1 \leq j \leq k\}$.

The general idea is the same as in the proof of Theorem 5, that is, a homomorphism $h$ from an $S'$-instance $J$ to $A_{C,q_0}$ defines a repair of the corresponding $S$-instance $I$. In fact, for any situation $A_1(a), A_2(a) \in I$ we must have $h(a) \in \{1, 2\}$ and in the repair of $I$ described by $h$ we then keep $A_2(a)$ and remove $A_1-h(a)$. The element 0 in $A_{C,q_0}$ is needed as a homomorphism target for constants $a \in \text{adom}(I)$ such that neither $A_1(a)$ nor $A_2(a)$ are in $I$.

**Lemma 12.** $\text{CQA}(C, q_0) \preceq_{\text{FO}} \text{coCSP}(A_{C,q_0})$ and $\text{coCSP}(A_{C,q_0}) \preceq_{\text{FO}} \text{CQA}(C, q_0)$.

The FO-reductions used for proving Lemma 12 are identical to those in the proof of Lemma 4, but of course the correctness proofs differ. Details are given in the appendix.

**Theorem 13.** Every binary restricted CQA problem is in PTime or coNP-complete.

### 6 Relating CQA and MMSNP

The classes of CQA problems identified so far all require queries to be hypertree UCQs. We now show that the transition from hypertree UCQs to unrestricted UCQs corresponds to the transition from CSP to MMSNP: while classifying the complexity of (MDiC, UCQ) and (MGAV, UCQ) is equivalent to classifying the complexity of coCSP (up to FO-reductions), classifying (MDiC, UCQ) and (MGAV, UCQ) is equivalent in the same sense to classifying coMMSNP. Since it is known that MMSNP $\approx_p$ CSP, the results in this section also imply that there is a dichotomy between PTime and coNP for (MDiC, UCQ) if and only if there is a dichotomy for (MDiC, UCQ) and only if the Feder-Vardi conjecture holds (and likewise for the corresponding CQA languages based on MGAVs). They also yield some insight into the problem of deciding FO-rewritability in (MDiC, UCQ) and (MGAV, UCQ): these problems are decidable if and only if FO-definability in MMSNP is decidable, which is an open problem.

Recall that coMMSNP is the class of problems definable by an MDDLog program. Thus, let $\Pi$ be an MDDLog program over schema $S$ and assume that $Q$ is the set of IDB relations in $\Pi$. A $Q$-type is a subset $t \subseteq Q$. We say that $\Pi$ admits precoloring if the EDB schema $S$ includes a monadic relation symbol $S_t$ for each $Q$-type $t$ and $\Pi$ includes rules (i) $S_t(x) \rightarrow Q(x)$ for all $Q$-types $t$ and all $Q \in t$ and (ii) $S_t(x) \land Q(x) \rightarrow \bot$ for all $Q$-types $t$ and all $Q \in Q \setminus t$; the $S_t$ relations are not allowed to be used in any other rule. We use coMMSNP$^{\text{pre}}$ to denote the class of all problems defined by an MDDLog program that admits precoloring and MMSNP$^{\text{pre}}$ to denote the class of their complements. Recall that we can w.l.o.g. assume CSPs to admit precoloring. The following result says that the same is true for (co)MMSNP.

**Theorem 14** (Bodirsky and Madelaine [12]). $\text{MMSNP} \preceq_{\text{FO}} \text{MMSNP}^{\text{pre}}$.

We start with establishing a counterpart of Theorem 3. Let $\Pi$ be an MDDLog program over schema $S$ that admits precoloring with IDB relations $Q$, as above. We use $\text{tp}$ to denote the set of all $Q$-types. Construct a problem $\text{CQA}(C_{\Pi}, q_{\Pi})$ in (MDiC, UCQ) over schema $S'$ which extends $S$ with unary relation symbols $Q_t$, $t \in \text{tp}$ (to be distinguished from the EDB
Consider each non-goal rule
\[ \forall x \rightarrow \bigwedge_{t \in \text{tp}} Q_t(x). \]

For each Q-type \( t \), we use \( \text{con}_i(x) \) to denote the conjunction \( \bigwedge_{t' \in \text{tp} \setminus \{t\}} Q_{t'}(x) \). We now construct the UCQ \( q_{\Pi} \), which contains the following two kinds of CQs.

1. Consider each non-goal rule
\[ \rho = \bigwedge_{1 \leq i \leq n} R_i(x_i) \land \bigwedge_{1 \leq i \leq m} S_i(y_i) \rightarrow \bigvee_{1 \leq i \leq \ell} S'_i(z_i) \]

where all \( R_i \) are from \( S \) and all \( S_i \) and \( S'_i \) are from \( Q \). Let \( x_1, \ldots, x_k \) be the variables in \( \rho \). Then include in \( q_{\Pi} \) the following CQ, for all sequences of Q-types \( t_1, \ldots, t_k \) such that (i) if \( S(x_i) \) occurs the body of \( \rho \) with \( S \in Q \), then \( S \in t_i \) and (ii) if \( S(x_i) \) occurs in the head of \( \rho \), then \( S \notin t_i \):
\[ \bigwedge_{1 \leq i \leq n} R_i(x_i) \land \bigwedge_{1 \leq i \leq k} \text{con}_{t_i}(x_i) \land \bigwedge_{1 \leq i \leq \ell} \text{con}_{t_i}(z_i); \]

2. Consider each goal rule
\[ \rho = \bigwedge_{1 \leq i \leq n} R_i(x_i) \land \bigwedge_{1 \leq i \leq m} S_i(y_i) \rightarrow \text{goal()} \]

where all \( R_i \) are from \( S \) and all \( S_i \) are from \( Q \). Let \( x_1, \ldots, x_k \) be the variables in \( \rho \). Then include in \( q_{\Pi} \) the following CQ, for all sequences of Q-types \( t_1, \ldots, t_k \) such that if \( S(x_i) \) occurs in the body of \( \rho \) with \( S \in Q \), then \( S \in t_i \):
\[ \bigwedge_{1 \leq i \leq n} R_i(x_i) \land \bigwedge_{1 \leq i \leq k} \text{con}_{t_i}(x_i). \]

The above construction parallels the one used in the proof of Theorem 3, with Q-types playing the role of elements of the CSP template. In the reduction from \( \Pi \) to CQA(\( \Pi, q_{\Pi} \)), we are given an \( S \)-instance \( I \) and construct an \( S' \)-instance \( T^I(I) \) by adding \( Q_{t_i}(a) \) for all \( a \in \text{adom}(I) \) and all Q-types \( t_i \). Then each minimal repair \( J \) of \( T^I(I) \) must satisfy, for each \( a \in \text{adom}(I) \), \( J \models \text{con}_{t_i}[a] \) for a unique Q-type \( t_i \) and thus assigns to each \( a \in \text{adom}(I) \) a Q-type \( t_a \). In this way, it describes a unique \( S \cup Q \)-instance \( I' \) that is obtained from \( I \) by adding \( P(a) \) whenever \( P \in t_a \). If \( J \neq q_{\Pi} \), then \( I' \) satisfies all non-goal rules of \( \Pi \), but no goal rule applies. Indeed, we show in the appendix that \( I \models \Pi \) iff \( T^I(I) \models q_{\Pi} \). We also give an FO-reduction from CQA(\( \Pi, q_{\Pi} \)) to \( \Pi \), which is again similar to what is done in the proof of Theorem 3.

**Theorem 15.** For every MDDLog program \( \Pi \) that admits precoloring, there is a problem CQA(\( C, q \)) from (MDC, UCQ) that satisfies CQA(\( C, q \)) and can be constructed in polynomial time.

We now show that (MGAV, UCQ) \( \preceq_{\text{FO}} \text{coMMSNP} \). Let CQA(\( C, q \)) be from (MGAV, UCQ) with schema \( S \). We define an MDDLog program \( \Pi_{C,Q} \) over EDB schema \( S' = S^{(1)} \cup \{ P_{t_i} \mid \Gamma \subseteq S^{(1)} \} \) where the \( P_{t_i} \) are fresh monadic relation symbols. Recall from Section 3 that for each \( \Gamma \subseteq S^{(1)} \), \( \text{rep}(\Gamma) \) denotes the corresponding set of repairs. The IDB relations of \( \Pi_{C,q} \)
are \( Q = S^{(1)} \cup \{ Q_{\Gamma} \mid \Gamma \subseteq S^{(1)} \} \) where the \( Q_{\Gamma} \) are monadic. The rules of \( \Pi_{C,q} \) are as follows:

\[
P_{\Gamma}(x) \rightarrow \bigvee_{\Lambda \in \text{rep}(\Gamma)} Q_{\Lambda}(x) \quad \text{for each } \Gamma \subseteq S^{(1)}
\]

\[
Q_{\Gamma}(x) \rightarrow P(x) \quad \text{for each } \Gamma \subseteq S^{(1)}, \ P \in \Gamma
\]

\[
P_{\Gamma}(x) \land P_{\Lambda}(x) \rightarrow \text{goal}() \quad \text{for all distinct } \Gamma, \Lambda \subseteq S^{(1)}
\]

\[
q' \rightarrow \text{goal}() \quad \text{for each disjunct } q' \text{ of } q.
\]

Lemma 16. \( CQA(C, q) \preceq_{\text{FO}} \Pi_{C,q} \) and \( \Pi_{C,q} \succeq_{\text{FO}} CQA(C, q) \).

The reductions used for establishing this lemma are exactly the FO-reductions from the proof of Lemma 4. In the appendix, we prove that these reductions are correct also in this case.

Theorem 17. For every CQA problem in \( (\text{MGAV}, \text{UCQ}) \), there is an FO-equivalent MDDLog program \( \Pi \) that can be constructed in single exponential time.

We thus obtain the following equivalences.

Corollary 18.

1. \( (\text{MDiC}, \text{UCQ}) \approx_{\text{FO}} (\text{MGAV}, \text{UCQ}) \approx_{\text{FO}} \text{coMMSNP} \);
2. \( (\text{MDiC}, \text{UCQ}) \approx_{p} (\text{MDiC}, \text{tUCQ}) \) and \( (\text{MGAV}, \text{UCQ}) \approx_{p} (\text{MGAV}, \text{tUCQ}) \).

Since the FO-reductions in the proof of Lemma 16 are identical to those in the proof of Lemma 4, all arguments used in Section 4 for relating FO- and Datalog-rewritability of CQA problems with hypertree UCQs to the FO- and Datalog-definability of CSPs can also be used to relate, in exactly the same way, CQA problems with unrestricted UCQs to MMSNP. Consequently, we obtain that FO-rewritability in \( (\text{MDiC}, \text{UCQ}) \) and \( (\text{MGAV}, \text{UCQ}) \) is decidable if FO-definability in MMSNP is decidable and the former is at most exponentially harder than the latter.

7 Relating CQA and GMSNP

Monadic disjointness constraints are surely a rather restricted class of denial constraints. In this section, we analyze a slightly more powerful class obtained by dropping the monadicity of MDiCs while still retaining some ‘uniformity’ across the atoms in the constraint. More precisely, a disjointness constraint (DiC) has the form \( \forall x \neg (R_{1}(x) \land \cdots \land R_{n}(x)) \) where all relations \( R_{i} \) have the same arity and all atoms \( R_{i}(x) \) use the same variables from \( x \) in the same order (multiple occurrences are allowed). We show that the connection between \( (\text{MDiC}, \text{UCQ}) \) and \( \text{coMMSNP} \) established in Section 6 can be lifted to \( (\text{DiC}, \text{UCQ}) \) and a generalization of \( \text{coMMSNP} \) called \( \text{coGMSNP} \) that corresponds to replacing MDDLog with frontier-guarded disjunctive Datalog [10]. It is straightforward to define a corresponding generalization of GAV constraints and to establish analogous results for them, but for simplicity we refrain from doing so.

Recall that we defined \( \text{coMMSNP} \) in terms of MDDLog. A frontier-guarded disjunctive Datalog (GDDLog) rule takes the form \( R_{1}(y_{1}) \land \cdots \land R_{k}(y_{k}) \rightarrow S_{1}(z_{1}) \lor \cdots \lor S_{m}(z_{m}) \) where the IDB predicates need not be monadic and for each head atom \( S_{i}(z_{i}) \), there must be a body atom \( R_{j}(y_{j}) \) with \( z_{i} \subseteq y_{j} \). A GDDLog program is then defined in the obvious way (with nullary goal predicate) and we use \( \text{coGMSNP} \) to denote the class of decision problems that are defined by a GDDLog program. GMSNP, which is also known as MMSNP\textsubscript{2} [32],
is considered an interesting candidate for a generalization of MMSNP that is still PTime-equivalent to CSP. It is, however, an open problem whether this is really the case. We nevertheless believe that relating (DiC, UCQ) and coGMSNP provides interesting additional insight into the computational complexity of CQA. Note that coGMSNP is also closely related to ontology-based data access with the guarded fragment of FO [10].

What we actually show in this section is coGMSNPpre \( \preceq_{\text{FO}} \) (DiC, UCQ) \( \preceq_{\text{FO}} \) coGMSNP where GMSNPpre is a version of GMSNP that admits precoloring defined as follows in analogy with MMSNPpre. Let \( \Pi \) be a GDDLog program over schema \( S \), assume that \( Q \) is the set of IDB relations in \( \Pi \), and let \( m \) be the maximal arity of relations in \( Q \). Fix variables \( x_1, \ldots, x_m \). For \( i \leq m \), an \( i\)-type is a set of relational atoms using relation symbols from \( Q \) and variables from \( x_i := x_1 \cdots x_i \). We say that \( \Pi \) admits precoloring if the EDB schema \( S \) includes a relation symbol \( S_i \) of arity \( i \) for each \( i\)-type \( t \), \( i \leq m \), and \( \Pi \) includes rules (i) \( S_i(x_1) \rightarrow R(x_1, \ldots, x_i) \) for all \( i\)-types \( t \) and all \( R(x_1, \ldots, x_i) \in t \) and (ii) \( S_i(x_1) \land R(x_1, \ldots, x_i) \rightarrow \bot \) for all \( i\)-types \( t \) and all \( R(x_1, \ldots, x_i) \notin t \) with \( R \in Q \) and \( x_1, \ldots, x_i \in x_i \); the \( S_i \) relations are not allowed to be used in any other rule. We leave it open whether coGMSNP \( \preceq_{\text{FO}} \) coGMSNPpre, but consider it likely that this is the case given the corresponding result for MMSNP mentioned in Section 6.

We start with proving coGMSNPpre \( \preceq_{\text{FO}} \) (DiC, UCQ), first observing that it suffices to consider GDDLog programs of a certain form, which we introduce next. Let \( Q \) be a schema that consists of relations which all have the same arity \( m \). A \( Q\)-type is a set of relational \( Q(x) \) with \( R \in Q \) and where \( x \) is a permutation of \( x_1 \cdots x_m \). We say that a GDDLog program \( \Pi \) with IDB relations \( Q \) is normalized if it satisfies the following conditions:

1. all EDB and IDB relations (except the goal relation) have the same arity \( m \), each variable may occur at most once in each head and body atom, and \( \Pi \) includes all rules of the form \( R(\ldots, x, \ldots, x, \ldots) \rightarrow \bot \) for each EDB and IDB relation \( R \);
2. the EDB schema \( S \) includes a relation symbol \( S_i \) of arity \( m \) for each \( Q\)-type \( t \) and \( \Pi \) includes rules (i) \( S_i(x_1, \ldots, x_m) \rightarrow R(x_{i_1}, \ldots, x_{i_m}) \) for all \( Q\)-types \( t \) and all \( R(x_{i_1}, \ldots, x_{i_m}) \in t \) and (ii) \( S_i(x_1, \ldots, x_m) \land R(x_{i_1}, \ldots, x_{i_m}) \rightarrow \bot \) for all \( Q\)-types \( t \) and atoms \( R(x_{i_1}, \ldots, x_{i_m}) \notin t \), where \( R \in Q \) and \( x_{i_1} \cdots x_{i_m} \) is a permutation of \( x_1 \cdots x_m \); the \( S_i \) relations are not allowed to be used in any other rule.\(^2\)

Working with normalized programs will simplify the subsequent constructions.

\textbf{Lemma 19.} For every GDDLog program \( \Pi \) that admits precoloring, there is a normalized GDDLog-Program \( \Pi' \) with \( \Pi \equiv_{\text{FO}} \Pi' \).

Let \( \Pi \) be a normalized GDDLog program over schema \( S \), let \( Q \) be the set of IDB relations in \( \Pi \) and \( m \) the unique arity of relations in \( \Pi \). We define a CQA setup CQA(\( \Pi', qn \)) from (DiC, UCQ) over schema \( S' \), that is, \( S \) extended with one relation \( Q_t \) of arity \( m \) for each \( Q\)-type \( t \). For an \( S'\)-instance \( J \), and a tuple \( a \in \text{adom}(J)^m \), we say that \( a \) is assigned \( Q\)-type \( t \) in \( J \) if there is a permutation \( b \) of \( a \) and a \( Q\)-type \( t' \) such that (i) \( Q_t'(b) \in J \) if \( t' \neq \hat{t} \) for all \( Q\)-types \( t' \) and (ii) \( \hat{t} \) is obtained from \( t \) by permuting the variables \( x_1 \cdots x_m \) in the same way in which the constants in \( a \) are permuted in \( b \), that is, if \( a = a_1 \cdots a_m \) and \( b = a_{i_1} \cdots a_{i_m} \), then \( \hat{t} = t[x_{i_1} \cdots x_{i_m}/x_1 \cdots x_m] \). We say that \( J \) is proper if every \( S\)-guarded tuple\(^3\) \( a \in \text{adom}(J)^m \) is assigned a unique \( Q\)-type. A proper \( S'\)-instance \( J \) represents an

\( ^2 \) Note that this is different from requiring \( \Pi \) to admit precoloring because admitting precoloring is about \( i\)-types whereas for normalized programs we use \( Q\)-types. In fact, a program in normal form cannot admit precoloring in the original sense because the conditions on the rules required for normality and admitting precoloring are incompatible.

\( ^3 \) A tuple \( a \in \text{adom}(J)^m \) is \( S \)-guarded if \( I \) contains some fact \( R(a) \) with \( R \in S \).
S-instance and an S ∪ Q-instance: the former is the reduct of J to schema S and the latter is obtained by starting with that reduct and then adding the fact R(α₁, . . . , αₘ) whenever α = α₁ · · · αₘ ∈ adom(J)ᵐ is assigned Q-type t and R(x₁, . . . , xₘ) ∈ t. Intuitively, the latter instance is supposed to represent an extension of the former instance obtained by applying the rules in Π.

We now define CQA(Π, qΠ). The set CΠ contains one disjointness constraint, namely

\[ ∀x_1 ⋮ ∀x_m \neg \left( \bigwedge_{t \text{ a Q-type}} Q_t(x_1, . . . , x_m) \right). \]

For a Q-type t, we use Ct(x) to denote the conjunction \[ \bigwedge_{a' \text{ a Q-type}, t' \neq t} Q_{t'}(x). \] A CQ q over schema S is forbidden if for each proper S'-instance J such that the S ∪ Q-instance I of J satisfies all non-goal rules in Π and no goal-rule of Π applies in I, we have J \[ \not\models q. \] The UC Q qΠ consists of the following CQs:

1. all forbidden CQs q over schema S' such that
   a. the number of variables in q is bounded by the maximum number of variables in a rule body in Π;
   b. for every atom R(x) there is a Q-type t and a permutation y of x such that Ct(y) is a subconjunction of q;
2. all CQs Ct(x) ∨ Ct(y) such that y is a permutation of x and t' \neq t[y/x].

\[ \text{Lemma 20. } \Pi \preceq_{\text{FO}} \text{CQA(Π, qΠ) and CQA(Π, qΠ) \preceq_{\text{FO}} Π}. \]

\[ \text{Proof. } \] For the first reduction, assume that an S-instance I is given. Define an S'-instance \[ T^i(I) \] as the extension of I with all facts \[ Q_t(a) \] such that t is a Q-type and \[ a \in \text{adom}(I)^m \]. We show in the appendix that \[ I \models Π \iff T^i(I) \models_c qΠ. \]

For the second reduction, let \[ I \] be an S-instance. Let \[ T^i(I) \] be the S-instance obtained from the reduct of I to schema S by the following sequence of operations:

1. drop each fact \[ R(a) \] such that for every permutation \[ b \] of \[ a \], there are distinct Q-types t and t' such that neither \[ Q_t(b) \] nor \[ Q_{t'}(b) \] are in I;
2. add \[ S_t(a) \] for each S-guarded tuple \[ a \in \text{adom}(I)^m \] that is assigned Q-type t in I (the assignment need not be unique).

We show in the appendix that \[ I \models_c qΠ \iff T^i(I) \models Π. \] Again \[ T^i \] is clearly FO-definable.

\[ \text{Theorem 21. } \text{For every GDDLog program } Π \text{ that admits precoloring, there is an FO-equivalent problem CQA(C, q) from (DiC, UCQ).} \]

We now turn towards showing that (DiC, UCQ) \[ \preceq_{\text{FO}} \text{coGMSNP}. \] Let CQA(C, q) over schema S be given with C a set of disjointness constraints. Let \[ m \] be the maximum arity of a relation in S. Fix variables \[ x_1, . . . , x_m. \] For \[ i \leq m, \] an i-type is a set of atoms \[ R(x_1, . . . , x_i) \] with \[ R \in S \] (all variables occur in fixed order and are distinct). We use \[ tp_i \] to denote the set of all i-types. For an S-instance I and \[ a \in \text{adom}(I)^i \], we use \[ tp_i(a) \] denote the i-type realized at \[ a \] in I, that is, the set of all atoms \[ R(x_1, . . . , x_i) \] such that \[ R(a) \in I. \]

We define a GDDLog program \[ Π_{C,q} \] over schema S' = \{ \[ R_t \mid t \in tp_i, i \leq m \}. \] The quantified relations in \[ Π_{C,q} \] are S ∪ \{ \[ Q_t \mid t \in tp_i, i \leq m \}. \] The disjointness constraints in C assign to each i-type t and each sequence of variables \[ x \in \{ x_1, . . . , x_m \}^i \] (repetitions allowed) a set of possible minimal repairs (which are also i-types), denoted with \[ rep_x(t). \] Formally, \[ rep_x(t) \] are the minimal repairs of \{ \[ R(x) \mid R(x_1, . . . , x_i) \in t \}. \]
1. for each $i$-type $t$, $i \leq m$, and each $x \in \{x_1, \ldots, x_m\}^{t}$: $R_t(x) \rightarrow \bigvee_{t' \in \text{rep}_i(t)} Q_{t'}(x)$
2. for each $i$-type $t$, $i \leq m$, and each $R(x_1, \ldots, x_i) \in t$: $Q_{t}(x_1, \ldots, x_i) \rightarrow R(x_1, \ldots, x_i)$
3. for all distinct $i$-types $t, t'$, $i \leq m$: $R_t(x_1, \ldots, x_i) \land R_{t'}(x_1, \ldots, x_i) \rightarrow \text{goal}()$
4. for each CQ $q'$ in $q$: $q' \rightarrow \text{goal}()$

Lemma 22. $\text{CQA}(C, q) \preceq_{\text{FO}} \Pi_{C, q}$ and $\Pi_{C, q} \preceq_{\text{FO}} \text{CQA}(C, q)$.

Proof. For the first reduction, let an $S$-instance $I$ be given. Define an $S'$-instance $T^+(I)$ that consists of all facts $R_{t_{\text{rep}}}(a)$ with $a \in \text{dom}(I)^i$, $i \leq m$. Clearly, $T^+$ is FO-definable. We show in the appendix that $I \models C q$ iff $T^+(I) \models \Pi_{C, q}$.

For the second reduction, let an $S'$-instance $I$ be given. If $R_t(a), R_{t'}(a) \in I$ for any $a \in \text{dom}(I)^i$, $i \leq m$, and $t \neq t'$, then answer ‘inconsistent’. Otherwise, replace each fact $R_t(a)$ with the set of facts $\{R(a) \mid R(x) \in t\}$, and call the result $T^+(I)$. Clearly, $T^+$ is FO-definable. We show in the appendix that $I \models \Pi_{C, q}$ iff $T^+(I) \models C q$.

Theorem 23. For every CQA problem in $(\text{DiC}, \text{UCQ})$, there is an FO-equivalent GDDLog program $\Pi$.

8 Conclusion

We find it intriguing that very simple integrity constraints such as MDiCs and MGAVs combined with structurally simple query languages such as tUCQs result in classes of CQA problems that are as difficult to analyze as CSPs, and that slight generalizations (such as DiCs) result in entering essentially unknown terrain (in the form of MMSNP). We believe that the CSP-CQA connection is not yet explored to the end. For example, the known non-dichotomy between PTime and NP for MMSNP extended with inequality [20] gives rise to the speculation that (MDiC, UCQ$^\neq$) might have no dichotomy between PTime and coNP either, where UCQ$^\neq$ denotes the class of UCQs that also admit inequality atoms. The proof of such a result will not be entirely simple, though, as it seems to require a version of Ladner’s theorem that relates to Ladner’s original theorem in a similar way in which the mortality problem of Turing machines relates to the halting problem. We leave this as future work.

Another interesting question is whether larger and more natural classes of CQA problems, such as those based on unrestricted denial constraints or on some form of functional dependency also have natural counterparts in the world of CSP and MMSNP. In fact, it is not even clear whether the correspondence between (MDiC, tUCQ) and coCSP can be extended from monadic disjointness constraints to monadic denial constraints in which relations are still required to be monadic, but where more than one variable can be used. For example, the constraint $\forall x \forall y : (A(x) \land B(y))$ is a monadic denial constraint, but not an MDiC. The main challenge is to deal with the problem that the ‘yes’-instances of each CSP are closed under homomorphic pre-images while the ‘no’-instances of CQA problems are not. Simply changing the monadic relations in the schema as in the proof of Theorem 5 is no longer sufficient because, in contrast to MDiCs, monadic denial constraints are not local to a single constant. We believe, however, that even if it should turn out that such differences prevent the CQA-CSP connection from gracefully extending beyond the classes of CQA problems considered here, it might still be possible to carry over techniques and intuitions from the CSP/MMSNP world, which has seen frantic development in the last decade.

Acknowledgements. We want to thank Manuel Bodirsky, Hubie Chen, and André Hernich for interesting and helpful discussions.
References


