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 know 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, inconcistency 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 $\widehat{q}$ 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 are 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 $\forall x\, \neg(P_1(x) \wedge \cdots \wedge P_n(x))$ and a *monadic GAV constraint (MGAV)* be of the form $\forall x\, (P_1(x) \wedge \cdots \wedge P_n(x) \rightarrow Q(x))$. 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 $\subseteq$ 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 MMSNP$_2$ [32]), which means to replace monadicity as stipulated in MMSNP for certain relations 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 $\mathbf{S} = (S_1, \ldots, S_k)$ of relation symbols with associated non-zero arity. A *fact* over $\mathbf{S}$ is an expression of the form $S(a_1, \ldots, a_n)$ where $S \in \mathbf{S}$ 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 $\mathbf{S}$ is a finite set of facts over $\mathbf{S}$. The *active domain* $\mathsf{adom}(I)$ 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 $\mathbf{a} = a_1 \cdots a_n$.

A *conjunctive query (CQ)* takes the form $q = \exists \mathbf{x}\, \varphi(\mathbf{x})$ where φ 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 $\forall \mathbf{x}\, \neg\varphi(\mathbf{x})$, where φ is a conjunction of relational atoms. A *global as view constraint (GAV)* takes the form $\forall \mathbf{x}\, \varphi(\mathbf{x}) \rightarrow S(\mathbf{x})$ where φ 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 $I \,\Delta\, J' \subsetneq I \,\Delta\, J$, 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 $\mathrm{CQA}(C, q)$, is defined by a set of constraints C and a query q. As input, an $\mathbf{S}$-instance I is given where $\mathbf{S}$ is the set of relation symbols used in C or q. The question is whether $I \models_C q$. We use $(\mathrm{DC}, \mathrm{UCQ})$ to denote the set of problems $\mathrm{CQA}(C, q)$ where C is a set of DCs and q a UCQ, and likewise for $(\mathrm{GAV}, \mathrm{UCQ})$ 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 $P_1 \preceq_p P_2$ if P_1 reduces to P_2 by a polynomial time reduction. We write $P_1 \preceq_{\mathsf{FO}} P_2$ if P_1 reduces to P_2 by an *FO-reduction*, defined as in [26]. However, most of our FO-reductions are of the following simple form. For problems P_1 and P_2 over schemas $\mathbf{S}_1$ and $\mathbf{S}_2$, a map T that assigns to each $\mathbf{S}_1$-instance I an $\mathbf{S}_2$-instance $T(I)$ is *FO-definable* if for every k-ary relation symbol R in $\mathbf{S}_2$, there is an FO-formula φ_R over $\mathbf{S}_1$ (equality and constants allowed) with k free variabes such that $R(\mathbf{a}) \in T(I)$ iff $I \models \varphi_R[\mathbf{a}]$, for all $\mathbf{a}$ in $\mathsf{adom}(I)$. Such a map gives rise to an FO-reduction of P_1 to P_2 if for all $\mathbf{S}_1$-instances I, we have $I \in P_1$ iff $T(I) \in P_2$. FO-reductions of this simple form differ from the general case in that (i) no arithmetic operations are admitted in the formulas φ_R and (ii) the domain of the $T(I)$ cannot be larger than the domain of I. With *FO-equivalence* and *PTime-equivalence* of two problems P_1 and P_2, denoted $P_1 \approx_{\mathsf{FO}} P_2$ and $P_1 \approx_p P_2$, we mean that there are reductions between P_1 and P_2 in both directions. For two classes of decision probems $\mathcal{C}_1$ and $\mathcal{C}_2$, we write $\mathcal{C}_1 \preceq_{\mathsf{FO}} \mathcal{C}_2$ if for every problem $p_1 \in \mathcal{C}_1$, there is a problem $p_2 \in \mathcal{C}_2$ such that $p_1 \preceq_{\mathsf{FO}} p_2$ and $p_2 \preceq_{\mathsf{FO}} p_1$. We write $\mathcal{C}_1 \approx_{\mathsf{FO}} \mathcal{C}_2$ and say that $\mathcal{C}_1$ and $\mathcal{C}_2$ are *FO-equivalent* if $\mathcal{C}_1 \preceq_{\mathsf{FO}} \mathcal{C}_2$ and $\mathcal{C}_2 \preceq_{\mathsf{FO}} \mathcal{C}_1$. The definition of $\mathcal{C}_1 \preceq_p \mathcal{C}_2$ and $\mathcal{C}_1 \approx_p \mathcal{C}_2$ is analogous, but based on polynomial time reductions.

Let A be an instance over schema $\mathbf{S}$. The *constraint satisfaction problem* $\mathrm{CSP}(A)$ is to decide, given an instance I over $\mathbf{S}$, 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 $\mathrm{CSP}(A)$. 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 $a \in \mathsf{adom}(A)$, there is a unary relation symbol $P_a \in \mathbf{S}$ 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 $\mathrm{CSP}(A) \approx_{\mathsf{FO}} \mathrm{CSP}(A')$ [31]. We use $\mathrm{coCSP}(A)$ to denote the complement problem of $\mathrm{CSP}(A)$ and coCSP to denote the set of all problems $\mathrm{coCSP}(A)$ 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 idenpotence' and 'pointedness'.

which can conveniently be defined as (negation-free) monadic disjunctive Datalog [10]. A *monadic disjunctive Datalog (MDDLog) rule* ρ has the form $R_1(\mathbf{y}_1) \wedge \cdots \wedge R_n(\mathbf{y}_n) \rightarrow S_1(x_1) \vee \cdots \vee S_m(x_m)$ with $m \geq 0$, $n > 0$, and $S_1, \ldots, S_m$ monadic. Every variable that occurs in the *head* $S_1(x_1) \vee \cdots \vee S_m(x_m)$ of ρ is also required to occur in ρ's *body* $R_1(\mathbf{y}_1) \wedge \cdots \wedge R_n(\mathbf{y}_n)$. Empty rule heads are denoted $\bot$. An *MDDLog program* Π is a finite set of MDDLog rules with a selected nullary *goal relation* goal that does not occur in rule bodies and only in *goal rules* of the form $R_1(\mathbf{x}_1) \wedge \cdots \wedge R_n(\mathbf{x}_n) \rightarrow \mathsf{goal}()$. Relation symbols that occur in the head of at least one rule of Π are *intensional (IDB)*, and all remaining relation symbols in Π are *extensional (EDB)*. Every MDDLog program Π defines a decision problem: given an $\mathbf{S}$-instance I, where $\mathbf{S}$ is the set of EDB relations in Π, decide whether $I \models \Pi$, that is, whether in every extension of I that satisfies all non-goal rules in Π, the body of at least one goal rule applies. We use 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 $(\mathrm{DC}, \mathrm{UCQ})$ and $(\mathrm{MGAV}, \mathrm{UCQ})$ that are FO-equivalent to coCSP. They involve restrictions on the admitted constraints as well as on the admitted queries. A denial constaint 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 $\forall x\, \neg(\mathsf{person}(x) \wedge \mathsf{process}(x))$ asserts disjointness of the types person and $\mathsf{process}$ while the MGAV $\forall x\, (\mathsf{professor}(x) \rightarrow \mathsf{person}(x))$ ensures a subsumption between the types $\mathsf{professor}$ and 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 $R(\mathbf{x})$ such that $x \in \mathbf{x}$. 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. Our notion of hypertree CQ is rather restrictive, see e.g. [6, 25] for more general forms of hypertrees; our form of hypertrees, though, is known to be intimately connected to the expressive power of CSPs [34].

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

▶ **Lemma 1.** *$(MDiC, \mathcal{Q}) \preceq_{\mathsf{FO}} (MGAV, \mathcal{Q})$ for $\mathcal{Q} \in \{UCQ, tUCQ\}$. Moreover, given a problem $CQA(C, q)$ from $(MDiC, \mathcal{Q})$ one can construct a problem $CQA(C', q')$ in $(MGAV, \mathcal{Q})$ such that $CQA(C, q) \approx_{\mathsf{FO}} CQA(C', q')$ in polynomial time.*

Proof. Let $\mathrm{CQA}(C, q)$ be a problem from $(\mathrm{MDiC}, \mathcal{Q})$ over schema $\mathbf{S}$. 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 $q' = q \vee \exists x\, M(x)$. We show that $\mathrm{CQA}(C, q) \preceq_{\mathsf{FO}} \mathrm{CQA}(C', q')$ and vice versa. For the former, we observe that for all $\mathbf{S}$-instances I, we have $I \models_C q$ iff $I \models_{C'} q'$. For the latter, it is easy to show that, for all $\mathbf{S} \cup \{M\}$-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 $(\mathrm{MDiC}, \mathrm{tUCQ}) \approx_{\mathsf{FO}} (\mathrm{MGAV}, \mathrm{tUCQ}) \approx_{\mathsf{FO}} \mathrm{coCSP}$. By Lemma 1, it suffices to show that $\mathrm{coCSP} \preceq_{\mathsf{FO}} (\mathrm{MDiC}, \mathrm{tUCQ})$ and $(\mathrm{MGAV}, \mathrm{tUCQ}) \preceq_{\mathsf{FO}} \mathrm{coCSP}$. We start with the former, improving upon a reduction by Fontaine [22] which shows that $\mathrm{coCSP} \preceq_p (\mathrm{GAV}, \mathrm{UCQ})$. Consider $\mathrm{CSP}(A)$ over schema $\mathbf{S}$ where A admits precoloring. We construct a problem $\text{CQA}(C_A, q_A)$ from $(\text{MDiC}, \text{tUCQ})$ over schema $\mathbf{S}'$ which extends $\mathbf{S}$ with unary relation symbols Q_a, $a \in \mathsf{adom}(A)$ (these symbols should be distinguished from the monadic relation symbols P_a in $\mathbf{S}$, $a \in \mathsf{adom}(A)$, which exist since A admits precoloring). C_A contains one monadic disjointness constraint:

$$\forall x \neg \bigwedge_{a \in \mathsf{adom}(A)} Q_a(x).$$

For each $a \in \mathsf{adom}(A)$, we use $\mathsf{con}_a(x)$ to denote the conjunction $\bigwedge_{e \in \mathsf{adom}(A) \setminus \{a\}} Q_e(x)$. The tUCQ q_A contains the following tCQ for each $R \in \mathbf{S}$ of arity n and each $\mathbf{a} = a_1 \cdots a_n \in \mathsf{adom}(A)^n$ such that $R(\mathbf{a}) \notin A$:

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

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

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

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

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 CQA(C, q) from (MDiC,tUCQ) that satisfies CQA$(C, q) \approx_{\mathsf{FO}}$ coCSP(A) and can be constructed in polynomial time.*

We now show that $(\text{MGAV}, \text{tUCQ}) \preceq_{\mathsf{FO}}$ coCSP. Let $\text{CQA}(C, q_0)$ be over schema $\mathbf{S}$ with C a set of MGAVs and q_0 a hypertree UCQ. We use $\mathbf{S}^{(1)}$ to denote the restriction of $\mathbf{S}$ to monadic relation symbols and $\mathbf{S}^{(>1)}$ for the restriction of $\mathbf{S}$ to non-monadic symbols. In the following, we define a CSP template A_{C,q_0} over schema $\mathbf{S}' = \mathbf{S}^{(>1)} \cup \{P_\Gamma \mid \Gamma \subseteq \mathbf{S}^{(1)}\}$. Note that there is an obvious natural translation of $\mathbf{S}$-instances into corresponding $\mathbf{S}'$-instances and vice versa; for example, an $\mathbf{S}'$-instance I is translated to an $\mathbf{S}$-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 $\Gamma \subseteq \mathbf{S}^{(1)}$, we use $\mathsf{rep}(\Gamma)$ to denote the set of all sets $\Gamma' \subseteq \mathbf{S}^{(1)}$ 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 q_0 (which are again hypertrees) and of all queries of the form $P(x)$, $P \in \mathbf{S}^{(1)}$. A *place* is a pair (q,x) with $q \in Q$ and $x \in \mathsf{var}(q)$. A *type* t is a set of places. The type $\mathsf{tp}_I(a)$ *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* q_0 if $(q,x) \notin t$ for any disjunct q of q_0 and $x \in \mathsf{var}(q)$. For $R \in \mathbf{S}$ of arity n, we say that a tuple $(t_1,\dots,t_n)$ of types is R*-coherent* if for any instance I and tuples of constants $(a_1,\dots,a_n)$ such that $\mathsf{tp}_I(a_i) = t_i$ for $1 \le i \le n$, after adding to I the fact $R(a_1,\dots,a_n)$, we still have $\mathsf{tp}_I(a_i) = t_i$ for $1 \le i \le n$. Now, the template A_{C,q_0} is defined as follows:

- the constants in A_{C,q_0} are the pairs $\langle t,\Gamma\rangle$ with t a realizable type that avoids q_0 and $\Gamma \subseteq \mathbf{S}^{(1)}$ such that $t|_{\mathbf{S}^{(1)}} \in \mathsf{rep}(\Gamma)$ where $t|_{\mathbf{S}^{(1)}}$ is the restriction of t to schema $\mathbf{S}^{(1)}$;
- A_{C,q_0} contains all facts of the form $P_\Gamma(\langle t,\Gamma\rangle)$;
- A_{C,q_0} contains all facts $R(\langle t_1,\Gamma_1\rangle,\dots,\langle t_n,\Gamma_n\rangle)$, R of arity n, if $(t_1,\dots,t_n)$ is R-coherent.

To understand the construction, consider the reduction from the complement of CQA(C,q_0) to CSP(A_{C,q_0}) and let I be an $\mathbf{S}$-instance that is an input to the former. We replace I with the corresponding $\mathbf{S}'$-instance $T^\uparrow(I)$ obtained from I by dropping all facts that involve a monadic relation and adding $P_{\Gamma_a}(a)$ for every element $a \in I$, where $\Gamma_a = \{P \mid P(a) \in I\}$. Then, a homomorphism h from $T^\uparrow(I)$ to A_{C,q_0} defines the repair of I that is obtained by repairing the monadic relations at each $a \in \mathsf{adom}(I)$ as suggested by $h(a) = \langle t,\Gamma\rangle$, namely to remove all $P(a)$ with $(P(x),x) \notin t$. Indeed, we show in the appendix that $I \not\models_C q_0$ iff $T^\uparrow(I) \to A_{C,q_0}$. It is again easy to see that $T^\uparrow(I)$ is FO-definable. For later use, we note explicitly that an FO-formula φ_{P_Γ} for defining P_Γ in $T^\uparrow(I)$ is given by

$$\varphi_{P_\Gamma}(x) = \bigwedge_{P\in\Gamma} P(x) \wedge \bigwedge_{P\in\mathbf{S}^{(1)}\setminus\Gamma} \neg P(x).$$

It is also interesting to note that the $\mathbf{S}$-instance I_A obtained from the template A_{C,q_0} by dropping all facts $P_\Gamma(\langle t,\Gamma\rangle)$ and adding $P(\langle t,\Gamma\rangle)$ for all $(P(x),x) \in t$ is a *universal minimal repair* in the following sense: (i) it is a minimal repair of the $\mathbf{S}$-instance that corresponds to A_{C,q_0} and (ii) any minimal repair of any $\mathbf{S}$-instance homomorphically maps to I_A.

▶ **Lemma 4.** *CQA*$(C,q_0) \preceq_{\mathsf{FO}}$ *coCSP*(A_{C,q_0}) *and coCSP*$(A_{C,q_0}) \preceq_{\mathsf{FO}}$ *CQA*(C,q_0).

Proof. The first reduction was described above. Correctness is proved in full detail in the appendix. For the second reduction, let an $\mathbf{S}'$-instance I be given. If there exists $a \in \mathsf{adom}(I)$ with $P_\Gamma(a), P_\Delta(a) \in I$ for some $\Gamma \neq \Delta$, then there is no homomorphism from I to A_{C,q_0} and "false" is returned. Otherwise define an $\mathbf{S}$-instance $T^\downarrow(I)$ by dropping all facts of the form $P_\Gamma(a)$ and adding $P(a)$ whenever $P_\Gamma(a) \in I$ with $P \in \Gamma$. We show in the appendix that $T^\downarrow(I) \to A_{C,q_0}$ iff $I \not\models_C q_0$. Clearly $T^\downarrow(I)$ is FO-definable. ◀

▶ **Theorem 5.** *For every problem CQA*(C,q) *from (MGAV,tUCQ), there is a CSP template* A *that satisfies coCSP*$(A) \approx_{\mathsf{FO}}$ *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)* $\approx_{\mathsf{FO}}$ *(MGAV, tUCQ)* $\approx_{\mathsf{FO}}$ *coCSP.*

It follows that there is a dichotomy between PTime and coNP for (MDiC, tUCQ) and (MGAV, tUCQ) if and only if the Feder-Vardi conjecture is true, and more generally that classifying the complexity of these classes of CQA problems is equivalent to classifying the complexity of CSPs (up to FO reductions).

4 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 $(\text{MDiC}, \text{tUCQ})$ and $(\text{MGAV}, \text{tUCQ})$. A problem $\text{CQA}(C, q)$ over schema $\mathbf{S}$ is *FO-rewritable* if there is a Boolean FO-query $\widehat{q}_{C,q}$ such that for all $\mathbf{S}$-instances I, we have $I \models_C q$ iff $I \models \widehat{q}_{C,q}$. Thus, FO-rewritability ensures that $\text{CQA}(C, q)$ 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 $\text{CQA}(C, q)$ where $C = \{\forall x\, \neg(A_1(x) \wedge A_2(x))\}$ and $q = \exists x\, A_1(x)$, which can be rewritten into $A_1(x) \wedge \neg A_2(x) \rightarrow \mathsf{goal}()$, but not into any Datalog program without negated monadic EDB atoms. Note that in constrast to consistent query answering problems, a class $\text{coCSP}(A)$ 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 $\text{CQA}(C, q)$ and $\text{coCSP}(A_{C,q})$ used in the proof of Theorem 5 are FO-reductions, it follows immediately that $\text{CQA}(C, q)$ is FO-rewritable if and only if $\text{coCSP}(A_{C,q})$ is FO-definable [26]. Given that $A_{C,q}$ can be constructed in single exponential time, we obtain a NExpTime upper bound for deciding FO-rewritability of CQA problems in $(\text{MDiC}, \text{tUCQ})$ and $(\text{MGAV}, \text{tUCQ})$.

▶ **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 $\text{CQA}(C, q)$ such that if M accepts an input x, then any FO-rewriting of $\text{CQA}(C, q)$ 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 $\forall x\, \neg(B(x) \wedge B'(x))$. ◀

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 $T^{\uparrow}(I)$ and $T^{\downarrow}(I)$ in the proof of Lemma 4.

▶ **Lemma 9.** *Let $CQA(C, q_0)$ and A_{C,q_0} be as in Lemma 4. Then $CQA(C, q_0)$ is Datalog-rewritable iff $coCSP(A_{C,q_0})$ is Datalog-definable.*

Proof. Let Π be a Datalog program that defines $\text{coCSP}(A_{C,q_0})$. Replace in Π all body atoms $P_\Gamma(x)$ with $\bigwedge_{P\in\Gamma} P(x) \wedge \bigwedge_{P\in\mathbf{S}^{(1)}\setminus\Gamma} \neg P(x)$ and denote the resulting program by Π'. It can be verified using the proof of Lemma 4 that $I \models_C q_0$ iff $I \models \Pi'$ for all $\mathbf{S}$-instances I. Thus Π' is a Datalog-rewriting of $\text{CQA}(C, q_0)$.

Conversely, assume that Π is a Datalog-rewriting of $\text{CQA}(C, q_0)$. Replace every rule ρ in Π by a set of rules as follows. For every variable x in ρ, replace the set S of conjuncts in the body of ρ that are of the form $P(x)$ and $\neg P(x)$, $P \in \mathbf{S}^{(1)}$, by any atom $P_\Gamma(x)$ such that $S \subseteq \{P(x) \mid P \in \Gamma\} \cup \{\neg P(x) \mid P \in \mathbf{S}^{(1)} \setminus \Gamma\}$. Moreover, add $P_\Gamma(x) \wedge P_\Lambda(x) \rightarrow \mathsf{goal}()$ as a new rule for all $\Gamma, \Lambda \subseteq \mathbf{S}^{(1)}$ with $\Gamma \neq \Lambda$. Denote the resulting program by Π'. It can be verified using the proof of Lemma 4 that Π' 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* NExpTime *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-definability of $\text{coCSP}(A_{C,q})$. The latter implies (non-recursive) Datalog-rewritability of $\text{CQA}(C, q)$. ◀

5 A Dichotomy Result

For restricted classes of CSPs, a dichotomy between PTime and 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 PTime and coNP.

We consider the class of problems $\text{CQA}(C, q_0)$ such that C consists of a single MDiC of the form $\forall x\, \neg(A_1(x) \wedge A_2(x))$ 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 r2CQA of CQA problems is not trivial since a straightforward analysis of the proof of Theorem 3 shows that $2\text{coCSP} \preceq_{\mathsf{FO}} \text{r2CQA}$, where $i\text{coCSP}$ is the class of complements of all problems that can be defined by a CSP template with at most i elements. Consequently, establishing a PTime / coNP dichotomy for r2CQA implies Schaefer's theorem; we actually find it remarkable that a class of CQA problems as simple as 2rCQA turns out to be that complex. In the following, we show that $\text{r2CQA} \preceq_{\mathsf{FO}} 3\text{coCSP}$ and thus obtain a dichotomy between PTime and coNP for r2CQA by Bulatov's result.

Let CQA(C, q_0) be a restricted binary CQA problem over schema $\mathbf{S}$. We define a CSP template A_{C,q_0} over schema $\mathbf{S}' = (\mathbf{S} \setminus \{A_1, A_2\}) \cup \{P_\Gamma \mid \Gamma \subseteq \{A_1, A_2\}\}$ as follows:

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

The general idea is the same as in the proof of Theorem 5, that is, a homomorphism h from an $\mathbf{S}'$-instance J to A_{C,q_0} defines a repair of the corresponding $\mathbf{S}$-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 $A_{h(a)}$ and remove $A_{3-h(a)}$. The element 0 in A_{C,q_0} is needed as a homomorphism target for constants $a \in \mathsf{adom}(I)$ such that neither $A_1(a)$ nor $A_2(a)$ are in I.

▶ **Lemma 12.** *CQA$(C, q_0) \preceq_{\mathsf{FO}}$ coCSP(A_{C,q_0}) and coCSP$(A_{C,q_0}) \preceq_{\mathsf{FO}}$ 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, tUCQ) and (MGAV, tUCQ) 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, tUCQ) if and only if there is such a dichotomy for (MDiC, UCQ) if 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 Π be an MDDLog program over schema $\mathbf{S}$ and assume that $\mathbf{Q}$ is the set of IDB relations in Π. A $\mathbf{Q}$-*type* is a subset $t \subseteq \mathbf{Q}$. We say that Π *admits precoloring* if the EDB schema $\mathbf{S}$ includes a monadic relation symbol S_t for each $\mathbf{Q}$-type t and Π includes rules (i) $S_t(x) \to Q(x)$ for all $\mathbf{Q}$-types t and all $Q \in t$ and (ii) $S_t(x) \wedge Q(x) \to \bot$ for all $\mathbf{Q}$-types t and all $Q \in \mathbf{Q} \setminus t$; the S_t relations are not allowed to be used in any other rule. We use coMMSNP$^{\mathsf{pre}}$ to denote the class of all problems defined by a MDDLog program that admits precoloring and MMSNP$^{\mathsf{pre}}$ 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]). *MMSNP $\preceq_{\mathsf{FO}}$ MMSNP$^{\mathsf{pre}}$.*

We start with establishing a counterpart of Theorem 3. Let Π be an MDDLog program over schema $\mathbf{S}$ that admits precoloring with IDB relations $\mathbf{Q}$, as above. We use tp to denote the set of all $\mathbf{Q}$-types. Construct a problem CQA(C_Π, q_Π) in (MDiC, UCQ) over schema $\mathbf{S}'$ which extends $\mathbf{S}$ with unary relation symbols Q_t, $t \in \mathsf{tp}$ (to be distinguished from the EDB relations S_t required because Π admits precoloring). C_Π contains one monadic disjointness constraint:

$$\forall x \neg \bigwedge_{t \in \mathsf{tp}} Q_t(x).$$

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

1. Consider each non-goal rule

$$\rho = \bigwedge_{1 \leq i \leq n} R_i(\mathbf{x}_i) \wedge \bigwedge_{1 \leq i \leq m} S_i(y_i) \rightarrow \bigvee_{1 \leq i \leq \ell} S'_i(z_i)$$

where all R_i are from $\mathbf{S}$ and all S_i and S'_i are from $\mathbf{Q}$. Let $x_1, \ldots, x_k$ be the variables in ρ. Then include in q_Π the following CQ, for all sequences of $\mathbf{Q}$-types $t_1, \ldots, t_k$ such that (i) if $S(x_i)$ occurs the body of ρ with $S \in \mathbf{Q}$, then $S \in t_i$ and (ii) if $S(x_i)$ occurs in the head of ρ, then $S \notin t_i$:

$$\bigwedge_{1 \leq i \leq n} R_i(\mathbf{x}_i) \wedge \bigwedge_{1 \leq i \leq k} \mathsf{con}_{t_i}(x_i) \wedge \bigwedge_{1 \leq i \leq \ell} \mathsf{con}_{t'_i}(z_i);$$

2. Consider each goal rule

$$\rho = \bigwedge_{1 \leq i \leq n} R_i(\mathbf{x}_i) \wedge \bigwedge_{1 \leq i \leq m} S_i(y_i) \rightarrow \mathsf{goal}()$$

where all R_i are from $\mathbf{S}$ and all S_i are from $\mathbf{Q}$. Let $x_1, \ldots, x_k$ be the variables in ρ. Then include in q_Π the following CQ, for all sequences of $\mathbf{Q}$-types $t_1, \ldots, t_k$ such that if $S(x_i)$ occurs in the body of ρ with $S \in \mathbf{Q}$, then $S \in t_i$:

$$\bigwedge_{1 \leq i \leq n} R_i(\mathbf{x}_i) \wedge \bigwedge_{1 \leq i \leq k} \mathsf{con}_{t_i}(x_i).$$

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

▶ **Theorem 15.** *For every MDDLog program Π that admits precoloring, there is a problem CQA(C, q) from (MDiC, UCQ) that satisfies CQA$(C, q) \approx_{\mathsf{FO}} \Pi$ and can be constructed in polynomial time.*

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

are $\mathbf{Q} = \mathbf{S}^{(1)} \cup \{Q_\Gamma \mid \Gamma \subseteq \mathbf{S}^{(1)}\}$ where the Q_Γ are monadic. The rules of $\Pi_{C,q}$ are as follows:

$$\begin{array}{rcll} P_\Gamma(x) & \to & \bigvee_{\Lambda \in \mathsf{rep}(\Gamma)} Q_\Lambda(x) & \text{for each } \Gamma \subseteq \mathbf{S}^{(1)} \\ Q_\Gamma(x) & \to & P(x) & \text{for each } \Gamma \subseteq \mathbf{S}^{(1)},\ P \in \Gamma \\ P_\Gamma(x) \wedge P_\Lambda(x) & \to & \mathsf{goal}() & \text{for all distinct } \Gamma, \Lambda \subseteq \mathbf{S}^{(1)} \\ q' & \to & \mathsf{goal}() & \text{for each disjunct } q' \text{ of } q. \end{array}$$

▶ **Lemma 16.** *$CQA(C,q) \preceq_{\mathsf{FO}} \Pi_{C,q}$ and $\Pi_{C,q} \preceq_{\mathsf{FO}} CQA(C,q)$.*

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 (MGAV, UCQ), there is an FO-equivalent MDDLog program Π that can be constructed in single exponential time.*

We thus obtain the following equivalences.

▶ **Corollary 18.**
1. *(MDiC, UCQ) $\approx_{\mathsf{FO}}$ (MGAV, UCQ) $\approx_{\mathsf{FO}}$ coMMSNP;*
2. *(MDiC, UCQ) $\approx_p$ (MDiC, tUCQ) and (MGAV, UCQ) $\approx_p$ (MGAV, 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 (MDiC, UCQ) and (MGAV, UCQ) is decidable iff 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' accross the atoms in the constraint. More precisely, a *disjointness constraint (DiC)* has the form $\forall \mathbf{x}\, \neg(R_1(\mathbf{x}) \wedge \cdots \wedge R_n(\mathbf{x}))$ where all relations R_i have the same arity and all atoms $R_i(\mathbf{x})$ use the same variables from $\mathbf{x}$ in the same order (multiple occurrences are allowed). We show that the connection between (MDiC, UCQ) and coMMSNP established in Section 6 can be lifted to (DiC, UCQ) and a generalization of coMMSNP called 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 coMMSNP in terms of MDDLog. A *frontier-guarded disjunctive Datalog (GDDLog) rule* takes the form $R_1(\mathbf{y}_1) \wedge \cdots \wedge R_k(\mathbf{y}_k) \to S_1(\mathbf{z}_1) \vee \cdots \vee S_m(\mathbf{z}_m)$ where the IDB predicates need not be monadic and for each head atom $S_i(\mathbf{z}_i)$, there must be a body atom $R_j(\mathbf{y}_j)$ with $\mathbf{z}_i \subseteq \mathbf{y}_j$. A GDDLog program is then defined in the obvious way (with nullary goal predicate) and we use coGMSNP to denote the class of decision problems that are defined by a GDDLog program. GMSNP, which is also known as MMSNP_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 coGMSNP$^{\mathsf{pre}} \preceq_{\mathsf{FO}}$ (DiC, UCQ) $\preceq_{\mathsf{FO}}$ coGMSNP where GMSNP$^{\mathsf{pre}}$ is a version of GMSNP that admits precoloring defined as follows in analogy with MMSNP$^{\mathsf{pre}}$. Let Π be a GDDLog program over schema $\mathbf{S}$, assume that $\mathbf{Q}$ is the set of IDB relations in Π, and let m be the maximal arity of relations in $\mathbf{Q}$. Fix variables $x_1, \dots, x_m$. For $i \leq m$, an *i-type* is a set t of relational atoms using relation symbols from $\mathbf{Q}$ and variables from $\mathbf{x}_i := x_1 \cdots x_i$. We say that Π *admits precoloring* if the EDB schema $\mathbf{S}$ includes a relation symbol S_t of arity i for each i-type t, $i \leq m$, and Π includes rules (i) $S_t(\mathbf{x}_i) \rightarrow R(x_{i_1}, \dots, x_{i_\ell})$ for all i-types t and all $R(x_{i_1}, \dots, x_{i_\ell}) \in t$ and (ii) $S_t(\mathbf{x}_i) \wedge R(x_{i_1}, \dots, x_{i_\ell}) \rightarrow \bot$ for all i-types t and all $R(x_{i_1}, \dots, x_{i_\ell}) \notin t$ with $R \in \mathbf{Q}$ and $x_{i_1}, \dots, x_{i_\ell} \in \mathbf{x}_i$; the S_t relations are not allowed to be used in any other rule. We leave it open whether coGMSNP $\preceq_p$ coGMSNP$^{\mathsf{pre}}$, but consider it likely that this is the case given the corresponding result for MMSNP mentioned in Section 6.

We start with proving coGMSNP$^{\mathsf{pre}} \preceq_{\mathsf{FO}}$ (DiC, UCQ), first observing that it suffices to consider GDDLog programs of a certain form, which we introduce next. Let $\mathbf{Q}$ be a schema that consists of relations which all have the same arity m. A $\mathbf{Q}$-*type* is a set of relational atoms $R(\mathbf{x})$ with $R \in \mathbf{Q}$ and where $\mathbf{x}$ is a permutation of $x_1 \cdots x_m$. We say that a GDDlog program Π with IDB relations $\mathbf{Q}$ 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 Π includes all rules of the form $R(\dots, x, \dots, x, \dots) \rightarrow \bot$ for each EDB and IDB relation R;
2. the EDB schema $\mathbf{S}$ includes a relation symbol S_t of arity m for each $\mathbf{Q}$-type t and Π includes rules (i) $S_t(x_1, \dots, x_m) \rightarrow R(x_{i_1}, \dots, x_{i_m})$ for all $\mathbf{Q}$-types t and all $R(x_{i_1}, \dots, x_{i_m}) \in t$ and (ii) $S_t(x_1, \dots, x_m) \wedge R(x_{i_1}, \dots, x_{i_m}) \rightarrow \bot$ for all $\mathbf{Q}$-types t and atoms $R(x_{i_1}, \dots, x_{i_m}) \notin t$, where $R \in \mathbf{Q}$ and $x_{i_1} \dots x_{i_m}$ is a permutation of $x_1 \dots x_m$; the S_t relations are not allowed to be used in any other rule.[2]

Working with normalized programs will simplify the subsequent constructions.

▶ **Lemma 19.** *For every GDDLog program Π that admits precoloring, there is a normalized GDDLog-Program Π' with $\Pi \approx_{\mathsf{FO}} \Pi'$.*

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

[2] Note that this is different from requiring Π to admit precoloring because admitting precoloring is about i-types whereas for normalized programs we use $\mathbf{Q}$-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 $\mathbf{a} \in \mathsf{adom}(I)^m$ is $\mathbf{S}$-*guarded* if I contains some fact $R(\mathbf{a})$ with $R \in \mathbf{S}$.

$\mathbf{S}$-instance and an $\mathbf{S} \cup \mathbf{Q}$-instance: the former is the reduct of J to schema $\mathbf{S}$ and the latter is obtained by starting with that reduct and then adding the fact $R(a_{i_1}, \ldots, a_{i_m})$ whenever $\mathbf{a} = a_1 \cdots a_m \in \mathsf{adom}(J)^m$ is assigned $\mathbf{Q}$-type t and $R(x_{i_1}, \ldots, x_{i_m}) \in t$. Intuitively, the latter instance is supposed to represent an extension of the former instance obtained by applying the rules in Π.

We now define $\text{CQA}(C_\Pi, q_\Pi)$. The set C_Π contains one disjointness constraint, namely

$$\forall x_1 \cdots \forall x_m \, \neg\big(\bigwedge_{t \text{ a } \mathbf{Q}\text{-type}} Q_t(x_1, \ldots, x_m) \big).$$

For a $\mathbf{Q}$-type t, we use $C_t(\mathbf{x})$ to denote the conjunction $\bigwedge_{t' \text{ a } \mathbf{Q}\text{-type}, t' \neq t} Q_{t'}(\mathbf{x})$. A CQ q over schema $\mathbf{S}'$ is *forbidden* if for each proper $\mathbf{S}'$-instance J such that the $\mathbf{S} \cup \mathbf{Q}$-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 UCQ q_Π consists of the following CQs:

1. all forbidden CQs q over schema $\mathbf{S}'$ 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 $R(\mathbf{x})$ there is a $\mathbf{Q}$-type t and a permutation $\mathbf{y}$ of $\mathbf{x}$ such that $C_t(\mathbf{y})$ is a subconjunction of q;
2. all CQs $C_t(\mathbf{x}) \wedge C_{t'}(\mathbf{y})$ such that $\mathbf{y}$ is a permutation of $\mathbf{x}$ and $t' \neq t[\mathbf{y}/\mathbf{x}]$.

▶ **Lemma 20.** $\Pi \preceq_{\mathsf{FO}}$ *$CQA(C_\Pi, q_\Pi)$ and $CQA(C_\Pi, q_\Pi) \preceq_{\mathsf{FO}} \Pi$.*

Proof. For the first reduction, assume that an $\mathbf{S}$-instance I is given. Define an $\mathbf{S}'$-instance $T^\uparrow(I)$ as the extension of I with all facts $Q_t(\mathbf{a})$ such that t is a $\mathbf{Q}$-type and $\mathbf{a} \in \mathsf{adom}(I)^m$. We show in the appendix that $I \models \Pi$ iff $T^\uparrow(I) \models_{C_\Pi} q_\Pi$. Clearly, $T^\uparrow$ is FO-definable.

For the second reduction, let I be an $\mathbf{S}'$-instance. Let $T^\downarrow(I)$ be the $\mathbf{S}$-instance obtained from the reduct of I to schema $\mathbf{S}$ by the following sequence of operations:
1. drop each fact $R(\mathbf{a})$ such that for every permutation $\mathbf{b}$ of $\mathbf{a}$, there are distinct $\mathbf{Q}$-types t and t' such that neither $Q_t(\mathbf{b})$ nor $Q_{t'}(\mathbf{b})$ are in I;
2. add $S_t(\mathbf{a})$ for each $\mathbf{S}$-guarded tuple $\mathbf{a} \in \mathsf{adom}(I)^m$ that is assigned $\mathbf{Q}$-type t in I (the assignment need not be unique).

We show in the appendix that $I \models_{C_\Pi} q_\Pi$ iff $T^\downarrow(I) \models \Pi$. Again $T^\downarrow$ is clearly FO-definable. ◀

▶ **Theorem 21.** *For every GDDLog program Π that admits precoloring, there is an FO-equivalent problem $CQA(C, q)$ from (DiC, UCQ).*

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

We define a GDDLog program $\Pi_{C,q}$ over schema $\mathbf{S}' = \{R_t \mid t \in \mathsf{tp}_i, i \leq m\}$. The quantified relations in $\Pi_{C,q}$ are $\mathbf{S} \cup \{Q_t \mid t \in \mathsf{tp}_i, i \leq m\}$ where the arities are defined in the obvious way. The disjointness constraints in C assign to each i-type t and each sequence of variables $\mathbf{x} \in \{x_1, \ldots, x_m\}^i$ (repetitions allowed) a set of possible minimal repairs (which are also i-types), denoted with $\mathsf{rep}_{\mathbf{x}}(t)$. Formally, $\mathsf{rep}_{\mathbf{x}}(t)$ are the minimal repairs of $\{R(\mathbf{x}) \mid R(x_1, \ldots, x_i) \in t\}$ viewed as an instance. Note that different sequences $\mathbf{x}$ may give rise to different repair sets for the same i-type t since the constraints in C might use variables multiple times in the same atom. The rules of $\Pi_{C,q}$ are as follows:

1. for each i-type t, $i \le m$, and each $\mathbf{x} \in \{x_1, \dots, x_m\}^i$: $R_t(\mathbf{x}) \to \bigvee_{t' \in \mathsf{rep}_{\mathbf{x}}(t)} Q_{t'}(\mathbf{x})$
2. for each i-type t, $i \le m$, and each $R(x_1, \dots, x_i) \in t$: $Q_t(x_1, \dots, x_i) \to R(x_1, \dots, x_i)$
3. for all distinct i-types t, t', $i \le m$: $R_t(x_1, \dots, x_i) \wedge R_{t'}(x_1, \dots, x_i) \to \mathsf{goal}()$
4. for each CQ q' in q: $q' \to \mathsf{goal}()$

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

Proof. For the first reduction, let an $\mathbf{S}$-instance I be given. Define an $\mathbf{S}'$-instance $T^{\uparrow}(I)$ that consists of all facts $R_{\mathsf{tp}_I(\mathbf{a})}(\mathbf{a})$ with $\mathbf{a} \in \mathsf{adom}(I)^i$, $i \le m$. Clearly, $T^{\uparrow}$ is FO-definable. We show in the appendix that $I \models_C q$ iff $T^{\uparrow}(I) \models \Pi_{C,q}$.

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

▶ **Theorem 23.** *For every CQA problem in (DiC, UCQ), there is an FO-equivalent GDDLog program Π.*

8 Conclusion

We find it intriguing that very simple integrity constraints such as MDiCs and MGAVs combined with structurally simple queries 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 GMSNP). 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 $\forall x \forall y \neg(A(x) \wedge B(y))$ 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

1 Foto N. Afrati and Phokion G. Kolaitis. Repair checking in inconsistent databases: algorithms and complexity. In *Proc. of ICDT*, volume 361 of *ACM International Conference Proceeding Series*, pages 31–41. ACM, 2009.

2 Marcelo Arenas and Leopoldo E. Bertossi. On the decidability of consistent query answering. In *Proc. of AMW*, volume 619 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.

3 Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In *Proc. of PODS*, pages 68–79. ACM Press, 1999.

4 Albert Atserias. On digraph coloring problems and treewidth duality. In *Proc. of LICS*, pages 106–115, 2005.

5 Libor Barto and Marcin Kozik. Constraint satisfaction problems of bounded width. In *Proc. of FOCS*, pages 595–603, 2009.

6 Catriel Beeri, Ronald Fagin, David Maier, and Mihalis Yannakakis. On the desirability of acyclic database schemes. *J. ACM*, 30(3):479–513, 1983.

7 Leopoldo E. Bertossi. Consistent query answering in databases. *SIGMOD Record*, 35(2):68–76, 2006.

8 Leopoldo E. Bertossi, Loreto Bravo, Enrico Franconi, and Andrei Lopatenko. The complexity and approximation of fixing numerical attributes in databases under integrity constraints. *Inf. Syst.*, 33(4-5):407–434, 2008.

9 Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. First-order rewritability of atomic queries in Horn description logics. In *Proc. of IJCAI*. IJCAI/AAAI, 2013.

10 Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: a study through disjunctive datalog, CSP, and MMSNP. In *Proc. of PODS*, pages 213–224. ACM, 2013.

11 Manuel Bodirsky, Hubie Chen, and Tomás Feder. On the complexity of MMSNP. *SIAM J. Discrete Math.*, 26(1):404–414, 2012.

12 Manuel Bodirsky and Florent Madeleine. Feder and Vardi's logic revisited. In preparation.

13 Andrei A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. ACM*, 53(1):66–120, 2006.

14 Andrea Calì, Domenico Lembo, and Riccardo Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. of PODS*, pages 260–271. ACM, 2003.

15 Jan Chomicki. Consistent query answering: Five easy pieces. In *Proc. of ICDT*, volume 4353 of *LNCS*, pages 1–17. Springer, 2007.

16 Jan Chomicki and Jerzy Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.*, 197(1-2):90–121, 2005.

17 Jan Chomicki, Jerzy Marcinkowski, and Slawomir Staworko. Computing consistent query answers using conflict hypergraphs. In *Proc. of CIKM*, pages 417–426. ACM, 2004.

18 David Cohen and Peter Jeavons. *The complexity of constraint languages*, chapter 8. Elsevier, 2006.

19 Stavros S. Cosmadakis, Haim Gaifman, Paris C. Kanellakis, and Moshe Y. Vardi. Decidable optimization problems for database logic programs (preliminary report). In *Proc. of STOC*, pages 477–490. ACM, 1988.

20 Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998.

21 Tomás Feder and Moshe Y. Vardi. Homomorphism closed vs. existential positive. In *18th IEEE Symposium on Logic in Computer Science (LICS 2003), 22-25 June 2003, Ottawa, Canada, Proceedings*, pages 311–320, 2003.

22 Gaëlle Fontaine. Why is it hard to obtain a dichotomy for consistent query answering? In *Proc. of LICS*, pages 550–559. IEEE Computer Society, 2013.

23 Ralph Freese, Marcin Kozik, Andrei Krokhin, Miklós Maróti, Ralph KcKenzie, and Ross Willard. On Maltsev conditions associated with omitting certain types of local structures. In preparation. Manuscript available from `http://www.math.hawaii.edu/~ralph/Classes/619/OmittingTypesMaltsev.pdf`.

24 Ariel Fuxman and Renée J. Miller. First-order query rewriting for inconsistent databases. In *Proc. of ICDT*, volume 3363 of *LNCS*, pages 337–351. Springer, 2005.

25 Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions and tractable queries. *J. Comput. Syst. Sci.*, 64(3):579–627, 2002.

26 Neil Immerman. *Descriptive complexity.* Springer, 1999.

27 Phokion G. Kolaitis and Enela Pema. A dichotomy in the complexity of consistent query answering for queries with two atoms. *Inf. Process. Lett.*, 112(3):77–85, 2012.

28 Paraschos Koutris and Dan Suciu. A dichotomy on the complexity of consistent query answering for atoms with simple keys. In *Proc. of ICDT*, pages 165–176. OpenProceedings.org, 2014.

29 Gábor Kun. Constraints, MMSNP and expander relational structures. *Combinatorica*, 33(3):335–347, 2013.

30 Benoit Larose, Cynthia Loten, and Claude Tardif. A characterisation of first-order constraint satisfaction problems. *Logical Methods in Computer Science*, 3(4), 2007.

31 Benoit Larose and Pascal Tesson. Universal algebra and hardness results for constraint satisfaction problems. *Theor. Comput. Sci.*, 410(18):1629–1647, 2009.

32 Florent R. Madelaine. Universal structures and the logic of forbidden patterns. *Logical Methods in Computer Science*, 5(2), 2009.

33 Florent R. Madelaine and Iain A. Stewart. Constraint satisfaction, logic and forbidden patterns. *SIAM J. Comput.*, 37(1):132–163, 2007.

34 Jaroslav Nesetril. Many facets of dualities. In *Bonn Workshop of Combinatorial Optimization*, pages 285–302, 2008.

35 Benjamin Rossman. Homomorphism preservation theorems. *J. ACM*, 55(3), 2008.

36 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proc. of STOC*, pages 216–226. ACM, 1978.

37 Slawomir Staworko and Jan Chomicki. Consistent query answers in the presence of universal constraints. *Inf. Syst.*, 35(1):1–22, 2010.

38 Balder ten Cate, Gaëlle Fontaine, and Phokion G. Kolaitis. On the data complexity of consistent query answering. In *Proc. if ICDT*, pages 22–33. ACM, 2012.

39 Jef Wijsen. On the first-order expressibility of computing certain answers to conjunctive queries over uncertain databases. In *Proc. of PODS*, pages 179–190. ACM, 2010.

40 Jef Wijsen. Charting the tractability frontier of certain conjunctive query answering. In *Proc. of PODS*, pages 189–200. ACM, 2013.

41 Jef Wijsen. A survey of the data complexity of consistent query answering under key constraints. In *Proc. of FoIKS*, volume 8367 of *LNCS*, pages 62–78. Springer, 2014.