Complexity and Expressive Power of Ontology-Mediated Queries*

Carsten Lutz

Fachbereich Informatik
Universität Bremen, Germany
clu@uni-bremen.de

Abstract

Data sets that have been collected from multiple sources or extracted from the web or often highly incomplete and heterogeneous, which makes them hard to process and query. One way to address this challenge is to use ontologies, which provide a way to assign a semantics to the data, to enrich it with domain knowledge, and to provide an enriched and uniform vocabulary for querying. The combination of a traditional database query with an ontology is called an ontology-mediated query (OMQ). The aim of this talk is to survey fundamental properties of OMQs such as their complexity, expressive power, descriptive strength, and rewritability into traditional query languages such as SQL and Datalog. A central observation is that there is a close and fruitful connection between OMQs and constraint satisfaction problems (CSPs) as well as related fragments of monadic NP, which puts OMQs into a more general perspective and gives raise to a number of interesting results.

1998 ACM Subject Classification H.2.3 Database Management, Query Languages

Keywords and phrases Ontology-Mediated Queries, Description Logic, Constraint Satisfaction

Digital Object Identifier 10.4230/LIPIcs.STACS.2016.2

Category Invited Talk

1 Description Logic and Ontologies

Description logics (DLs) are a widely known family of knowledge representation formalisms that originated in the 1980s and are now popular as ontology languages; a slightly outdated though still useful overview is given in the DL handbook [6]. From a logic perspective, DLs are best viewed as decidable fragments of first-order logic (FO), some of which are variants of modal logic while others are more closely related to Datalog-like rule languages. DLs come with their own syntax, which involves using logical quantifier symbols in a non-standard way (actually in the same way in which diamonds and boxes are used in modal logic). As a basic example, we introduce the description logic $\mathcal{ALC}$ originating from [37].[1] Its syntax is defined in two steps. In the first step, one inductively defines a set of logical formulas called *concepts* and in the second step, concepts are put into relation with each other in a logical theory called a *TBox*. We fix two sets of symbols beforehand. A set of *concept names* which are denoted with A and B and correspond to monadic predicates in FO; and a set of *role names* which are denoted with r and correspond to dyadic predicates. Predicates of

* This work was partially supported by the ERC Consolidator Grant 647289 (CODA). The material surveyed in this text is mostly taken from [9], which is joint work with Meghyn Bienvenu, Balder ten Cate, and Frank Wolter.

[1] $\mathcal{ALC}$ stands for *Attributive concept Language with Complementation*, a largely historic name.

higher (or lower) arity are not included in most DLs. The following table lists the concept constructors of $\mathcal{ALC}$ along with their FO translation:[2]

$$\begin{array}{lll@{\qquad}lll@{\qquad}lll} A & \equiv & A(x) & C \sqcap D & \equiv & C(x) \wedge D(x) & \exists r.C & \equiv & \exists y\,(r(x,y) \wedge C(y)) \\ \neg C & \equiv & \neg C(x) & C \sqcup D & \equiv & C(x) \vee D(x) & \forall r.C & \equiv & \forall y\,(r(x,y) \rightarrow C(y)) \end{array}$$

where C and D range over (potentially compound) concepts. Note that every concept translates into an equivalent FO formula with one free variable. As in modal logic (and unlike in FO), monadic and dyadic predicates are used in a non-interchangeable way in the syntax. In a sense, DLs have a strong focus on monadic predicates and on formulas with one free variable while dyadic predicates only help to define such formulas. In fact, the connection to CSPs discussed in Section 4 rests on that setup in an essential way. The focus on monadic predicates came from application demands, where it is most important to describe *classes* of objects with similar properties. As an example, let us assume that we want to capture knowledge about the domain of traveling. We can use an $\mathcal{ALC}$ concept to describe the class of hotels in Orléans that got a high rating or are close to the STACS2016 venue:

$$\mathsf{Hotel} \sqcap \exists \mathsf{location}.\mathsf{Orleans} \sqcap (\exists \mathsf{ranking}.\mathsf{High} \sqcup \exists \mathsf{closeTo}.\mathsf{UniversityCampus}).$$

While concepts are used to describe classes such as the one above, TBoxes interrelate these classes and in this way formalize the knowledge of an application domain and give a semantics to the predicate symbols used. In $\mathcal{ALC}$, a TBox is simply a finite set of *concept inclusions* $C \sqsubseteq D$, where both C and D are $\mathcal{ALC}$ concepts. While concepts translate into FO formulas with one free variable, a concept inclusion $C \sqsubseteq D$ translates into the FO sentence $\forall x\,(C(x) \rightarrow D(x))$ and a TBox translates into the conjunction of (the FO translations) of the concept inclusions contained in it. For example, the following $\mathcal{ALC}$ TBox describes some basic knowledge about the traveling domain:

$$\begin{array}{rcl} \mathsf{Airlline} & \sqsubseteq & \mathsf{BudgetAirline} \sqcup \mathsf{RegularAirline} \\ \mathsf{BudgetAirline} & \sqsubseteq & \neg \mathsf{RegularAirline} \\ \mathsf{AirTicket} & \sqsubseteq & \exists \mathsf{issuedBy}.\mathsf{Airline} \\ \mathsf{AirTicket} \sqcap \exists \mathsf{issuedBy}.\mathsf{BudgetAirline} & \sqsubseteq & \neg \exists \mathsf{class}.\mathsf{Business} \end{array}$$

The above TBox is typical for a DL TBox in that it mainly concentrates on describing properties of classes important for the application domain such as $\mathsf{Airline}$ and $\mathsf{AirTicket}$. This is again the focus on monadic predicates mentioned above.

The name *TBox* stems from the area of knowledge representation and DLs where, historically, it stands for "terminological box". In more modern terms, TBoxes are often called *ontologies* [25]. Ontologies have applications in data access as described in more detail in Section 2 and are also used to produce standardized vocabularies. The latter is particularly important in genetics and biology where the terminology is very extensive and in medicine where there is a need to establish a standardized vocabuary for data exchange and accounting. Several hundred ontologies from the biomedical domain have been collected in the BioPortal maintained by the National Centre for Biomedical Ontology, see `http://www.bioontology.org/`. In these areas, ontologies often aim to be of general purpose and can contain up to several hundred thousand classes, as in the case of the SNOMED CT healthcare ontology which is developed and maintained by the International Health Terminology Standards Development Organisation [38]. In contrast, ontologies used

[2] The translation is actually into the two-variable guarded fragment of FO.

for data access are often custom tailored towards a particular application and tend to be smaller in size. Ontologies also play an important rôle in the semantic web, which has led to their standardization as the OWL family of web ontology languages by the World Wide Web committee (W3C). The first version of the OWL recommendation has been released in 2004, followed by the OWL 2 recommendation in 2012 [34]. OWL 2 is a collection of five languages, four of which are DLs. It comes with a variety of more "web friendly" syntaxes based, e.g., on XML and RDF. A large collection of tools for OWL ontologies is available including ontology editors, APIs, logical reasoners, and so on.

From a theoretical perspective, bisimulation is an important tool to understand the expressive power of the description logic $\mathcal{ALC}$. In fact, $\mathcal{ALC}$ concepts are a notational variant of formulas in the modal logic K: simply replace "$\sqcap$" with "$\wedge$", "$\sqcup$" with "$\vee$", "$\exists r.$" with $\Diamond_r$, and "$\forall r.$" with $\Box_r$, and read concept names as propositional letters. $\mathcal{ALC}$ concepts thus inherit the well-known relation between K and bisimulation, manifested e.g. in van Benthem's theorem [22]. TBoxes add a "global" flavour in the sense that the FO translation of concept inclusions universally quantifies over all elements of the universe. This gives raise to the following variation of van Benthem's theorem, which precisely characterizes the expressive power of $\mathcal{ALC}$ TBoxes within FO.

We refer to FO without function symbols and constants and with only monadic and dyadic predicates, which we identify with concept and role names, respectively. We say that a relational structure $\mathfrak{A}_1$ is *globally bisimilar* to a relational structure $\mathcal{A}_2$ if for every d_1 in the universe of $\mathfrak{A}_1$, there is a d_2 in the universe of $\mathfrak{A}_2$ such that there is a bisimulation between $\mathfrak{A}_1$ and $\mathfrak{A}_2$ that related d_1 with d_2, and vice versa. Further, we say that an FO sentence φ is *invariant under global bisimulation* if for all relational structures $\mathfrak{A}_1, \mathfrak{A}_2$ such that $\mathfrak{A}_1 \models \varphi$ and $\mathfrak{A}_1$ is globally bisimilar to $\mathfrak{A}_2$, we have $\mathfrak{A}_2 \models \varphi$.

▶ **Theorem 1** ([29]). *An FO sentence φ is equivalent to an $\mathcal{ALC}$ TBox iff φ is preserved under global bisimulation and under disjoint union.*

The description logic $\mathcal{ALC}$ presented here is a bit too simple and inexpressive to be useful in many applications. In fact, the languages of the OWL 2 family include a wealth of additional expressive means, selected such that satisfiability (and several other relevant reasoning problems) are still decidable, but the expressive power is more satisfactory. In the literature, a large number of decription logics have been considered that balance expressive power and computational complexity in different ways. Although many of them do not admit characterizations as clean as Theorem 1, there are often intimiate relationships with suitably modified notions of bisimulation [29]. It should also be mentioned that several widely-used DLs such as $\mathcal{EL}$, DL-Lite, and their extensions do not include negation, disjunction, and universal quantification [5, 17]. Rather than to modal logic, these languages are more closely related to Datalog and its extension with existential quantification in the rule heads, known as tuple-generating dependencies, existential rules, and Datalog$^{\pm}$ [14, 33].

2 Ontology-Mediated Queries

Using DL ontologies for data access is a very active branch of research, see for example [8, 16, 26] for some recent surveys. In DLs, data is commonly stored in a so-called *ABox* (another historical name, standing for "assertional Box"), which is simply a finite set of ground facts of the form $A(a)$ and $r(a, b)$ called *assertions*, where a, b are FO constants. Note that also ABoxes use only unary and dyadic predicates. This is appropriate for example for web data represented in RDF [35]. It is less appropriate for data that stems from traditional database systems, whose schemas often use relations of high arity; schema mappings have been proposed as a workaround, see [16]. The data stored in an ABox is considered (potentially) incomplete, that is, additional assertions except those explicitly stated in the ABox might be true. This is reflected in the semantics of query answering. Before giving details, we first introduce some relevant query languages.

A *conjunctive query (CQ)* takes the form $q = \exists \mathbf{x}\, \varphi(\mathbf{x}, \mathbf{y})$ with $\mathbf{x}, \mathbf{y}$ tuples of variables and φ a conjunction of atoms of the form $A(x)$ and $r(x, y)$ that uses only variables from $\mathbf{x} \cup \mathbf{y}$. The variables in $\mathbf{x}$ are called *answer variables*, the *arity* of q is the length of $\mathbf{x}$, and q is *Boolean* if it has arity zero. A *union of conjunctive queries (UCQ)* takes the form $q_1 \vee \cdots \vee q_k$ where $q_1, \ldots, q_k$ are CQs with the same answer variables. An *atomic query (AQ)* is a conjunctive query of the form $A(x)$ and a *Boolean atomic query (BAQ)* is a conjunctive query of the form $\exists x\, A(x)$.

As an example, consider the ABox

$$\mathsf{AirTicket}(\mathsf{offer12}) \quad \mathsf{class}(\mathsf{offer12}, \mathsf{j6}) \quad \mathsf{BusinessClass}(\mathsf{j6}),$$

the TBox given above, and the query $q = \exists y\, \mathsf{Ticket}(x) \wedge \mathsf{issuedBy}(x, y) \wedge \mathsf{RegularAirline}(y)$ asking to return all tickets issued by a regular airline. The domain knowledge in the TBox adds additional facts to the data such as that the ticket named offer12 is issued by some airline, and that this airline cannot be a budget airline, thus must be a regular airline, which allows to return offer12 as an answer to the query.

A (finite or infinite) set of assertions can be viewed as a relational structure in an obvious way. A relational structure $\mathfrak{A}$ is a *model* of an ABox $\mathcal{A}$ if it can be obtained from $\mathcal{A}$ by extending it with additional assertions, possibly involving additional constants. $\mathfrak{A}$ is a model of a TBox $\mathcal{T}$ is it satisfies (the FO translations of) all concept inclusions in $\mathcal{T}$. A tuple of constants $\mathbf{a}$ is an *answer* to a query q in $\mathfrak{A}$ if $\mathfrak{A} \models q[\mathbf{a}]$ in the standard sense of FO logic. Moreover, $\mathbf{a}$ is a *certain answer* to q in an ABox $\mathcal{A}$ given a TBox $\mathcal{T}$ if $\mathbf{a}$ is an answer to q in all models of $\mathcal{A}$ and $\mathcal{T}$. In the above example, offer12 is a certain answer.

An *ontology mediated query (OMQ)* is a triple $(\mathcal{T}, \Sigma, q)$ where $\mathcal{T}$ is a TBox, Σ a data signature (that is, a set of concept and role names that can occur in the ABox), and q an actual query such as a CQ or an AQ. Note that $\mathcal{T}$ might introduce symbols that do not occur in the data, in this way enriching the vocabulary available for formulating the query q. An OMQ $(\mathcal{T}, \Sigma, q)$ is Boolean if q is. We use $(\mathcal{L}, \mathcal{Q})$ to denote the OMQ language which consists of all OMQs $(\mathcal{T}, \Sigma, q)$ with $\mathcal{T}$ formulated in the DL $\mathcal{L}$ and q formulated in the query language $\mathcal{Q}$. The most common choices for $\mathcal{Q}$ are AQs, CQs, and UCQs, giving raise for example to the OMQ languages $(\mathcal{ALC}, \text{AQ})$ and $(\mathcal{ALC}, \text{CQ})$. BAQs are somewhat less common, but, as we will see, constitute very natural cases when establishing the connection between OMQs and CSPs.

Given the exposition above, it is natural to ask in which way adding an ontology to a database query affects the complexity of query answering, and how the expressive power of OMQs relates to the expressive power of more standard database query languages. For simplicity, we will mostly concentrate on Boolean OMQs. As in the case of conventional databases, there are several complexity measures that one might study. In particular, combined complexity considers both the ABox and the OMQ as an input while data complexity assumes the OMQ to be fixed and considers only the ABox to be the input. Since OMQs tend to be small compared to the actual data,[3] data complexity is often viewed as the

[3] Very large ontologies such as SNOMED CT clearly constitute an exception and suggest other complexity measures such as treating both the ABox and the TBox as an input, but not the actual query.

more natural measure. It is not hard to see that there are OMQs that are coNP-complete in data complexity. For example, the following OMQ encodes non-3-colorability:

$$\mathcal{T} = \{\top \sqsubseteq R \sqcup G \sqcup B, \quad R \sqcup \exists r.R \sqsubseteq D, \quad G \sqcup \exists r.G \sqsubseteq D, \quad B \sqcup \exists r.B \sqsubseteq D\}$$
$$\Sigma = \{r\}$$
$$q = \exists x\, D(x).$$

Here, $\top$ is an abbreviation for a tautology such as $A \sqcup \neg A$ and the concept name D signals a defect. Recall that, without ontologies, the data complexity of FO queries such as CQs and UCQs is extremely low, namely in AC_0. By adding ontologies, we have thus transitioned from highly efficient to intractable. This seems unacceptable, but rarely causes unsolvable problems in practice because real-world ontologies do not encode combinatorial problems such as 3-colorability. But how to separate the tractable cases from the intractable ones in a theoretically clean way? A brute-force way is to replace $\mathcal{ALC}$ with a very restricted ontology language such as the DLs $\mathcal{EL}$ and DL-Lite mentioned above, resulting in PTime-completeness in data complexity. While this is acceptable for some applications, it is unacceptable for others where negation and disjunction are needed for proper modeling, though typically in a harmless way. It is tempting but seems impossible to charaterize what “harmless” means in a syntactic way. To achieve maximum flexibility and to circumvent syntactic characterizations, a non-uniform approach was advocated in [30] whose aim is to classify the exact complexity of *every* OMQ within a given OMQ language.

3 Constraint Satisfaction Problems and MMSNP

There is an interesting connection between OMQs and constraint satisfaction problems (CSP) which puts OMQs into a more general perspective and allows to obtain a number of interesting results regarding their expressive power and (non-uniform) complexity. This connection also extends in a very natural way to the logical generalization MMSNP of CSPs introduced in a seminal paper of Feder and Vardi [19], and it provides new motivation for studying this logic. In this section, we briefly introduce CSPs and MMSNP.

There are various equivalent definitions of CSPs [36]. We choose here to define them in terms of homomorphisms between relational structures. A *template* is a finite relational structure in some signature Σ; in contrast to the previous sections, the arity of predicates is unrestricted. Each template T defines the class of finite relational Σ-structures $\text{CSP}(T) = \{S \mid S \rightarrow T\}$ where $S \rightarrow T$ means that there is a homomorphism from S to T. The constraint satisfaction problem for template T is to decide, given a finite relational Σ-structure S, whether $S \in \text{CSP}(T)$. It is easy to see that every CSP is in NP and that there are CSPs that are NP-complete. An example is $\text{CSP}(K_3)$ where K_3 is the 3-clique and Σ contains only a single dyadic predicate r which represents edges in graphs; note that an undirected graph S is in $\text{CSP}(K_3)$ if and only if S is 3-colorable. Additional NP-complete problems that can be presented as CSPs include 3-satisfiability and integer programming on finite domains. Other NP-complete problems such as Hamilton cycle cannot be presented as CSPs because the class of their “yes”-instances is not closed under homomorphic pre-images. Of course, there are also CSPs of lower complexity such as $\text{CSP}(K_2)$, which is 2-colorability and thus PTime-complete.

Feder and Vardi have asked for a complete classification of the complexity of all CSPs, in particular for a precise delineation of the CSPs that can be solved in PTime from those that are NP-complete [19]. They have conjectured that a transparent such delineation is possible and that there is a dichotomy between PTime and NP for CSPs, that is, that every CSP is in PTime or NP-hard, unlike the NP-intermediate problems whose existence is established by Ladner's theorem [27]. While the general case is still open, a lot of progress has been made and the dichotomy conjecture has been confirmed for special cases such as undirected graphs [24] and for oriented cycles [18]. To connect the dichotomy question with the field of descriptive complexity, Feder and Vardi identified a fragment of monadic NP called MMSNP (for "monotone monadic strict NP") that corresponds rather closely to CSPs. A sentence of MMSNP takes the form

$$\exists X_1 \cdots \exists X_n \forall x_1 \cdots \forall x_m \, \varphi$$

where φ is a quantifier- and equality-free FO formula in which every atom $R(\mathbf{x})$ with $R \notin \{X_1, \ldots, X_n\}$ occurs only with negative polarity. For example, $\text{CSP}(K_3)$ is equivalent to the MMSNP formula

$$\begin{aligned}\exists R \, \exists G \, \exists B \, \forall x_1 \forall x_2 \; & (R(x_1) \vee G(x_1) \vee B(x_1)) \wedge \\ & \neg(R(x_1) \wedge r(x_1, x_2) \wedge R(x_2)) \wedge \\ & \neg(G(x_1) \wedge r(x_1, x_2) \wedge G(x_2)) \wedge \\ & \neg(B(x_1) \wedge r(x_1, x_2) \wedge B(x_2))\end{aligned}$$

where r is again the dyadic edge relation in input graphs. While MMSNP is more expressive than CSP (with non-monochromatic triangle[4] being a witnessing problem), a main result of Feder and Vardi shows that there is a dichotomy between PTime and NP for CSPs if and only if there is such a dichotomy for MMSNP. Thus, MMSNP can be viewed as a well-behaved extension of CSPs. There are several seemingly minor generalizations of MMSNP which destroy this property and result in non-dichotomy.

4 Ontologies, CSP, and MMSNP

The examples given above might already suggest to the reader that there is a connection between OMQs and CSPs. In fact, CSPs can be viewed as generalized coloring problems and it is not hard to adapt the OMQ from Section 2 expressing non-3-colorability to any CSP over a signature with only unary and dyadic predicates. Let T be a template over such a signature Σ. The OMQ $Q_T = (\mathcal{T}, \Sigma, q)$ is defined by setting $q = \exists x \, D(x)$ and including the following concept inclusions in $\mathcal{T}$:

- $\top \sqsubseteq A_{d_1} \sqcup \cdots \sqcup A_{d_k}$ when the universe of T is $U = \{d_1, \ldots, d_k\}$;
- $A_d \sqcap B \sqsubseteq D$ when $d \notin B^T$ for all $d \in U$ and concept names $A \in \Sigma$;
- $A_{d_1} \sqcap \exists r.A_{d_2}$ when $(d_1, d_2) \notin r^T$ for all $d_1, d_2 \in U$ and role names $r \in \Sigma$.

Note that Q_T is formulated in the OMQ language $(\mathcal{ALC}, BAQ)$. It is equivalent to the complement of T in the sense that for any finite Σ-structure (equivalently: Σ-ABox) $\mathcal{A}$, we have $\mathcal{A} \not\to T$ if and only if Q_T is true on $\mathcal{A}$. Conversely, it is possible to convert any OMQ Q from $(\mathcal{ALC}, \text{BAQ})$ into a CSP whose complement is equivalent to Q. As the template, one uses a structure that is similar to the structures emerging from filtration and type elimination constructions for modal and description logics [10]. In particular, 1-types are used as elements of the template, and this is sufficient only because of the essentially monadic nature of DLs.

We use coCSP to denote the class of complements of CSPs and likewise for coMMSNP. The next result yields a strong connection between OMQs and the CSP world.

[4] The class of all undirected graphs whose nodes can be colored black and white such that neither the all-white triangle nor the all-black triangle admits a homormophism into the colored graph.

▶ **Theorem 2** ([9]). *The following have the same expressive power:*
1. *($\mathcal{ALC}$, BAQ) and coCSP;*
2. *($\mathcal{ALC}$, UCQ) and coMMSNP.*

Theorem 2 can be seen as clarifying the descriptive complexity of the fundamental OMQ languages ($\mathcal{ALC}$, BAQ) and ($\mathcal{ALC}$, UCQ). It also has immediate consequences for non-uniform data complexity: classifying the complexity of all OMQs from these OMQ languages is equivalent to classifying the complexity of CSPs with only unary and dyadic predicates. It is known that there is a dichotomy between PTime and NP for such CSPs if and only if there is such a dichotomy for unrestricted CSPs [19]. Consequently, the mentioned OMQ languages have a dichotomy between PTime and coNP if and only if the Feder-Vardi conjecture holds. Other results from CSP research carry over as well. Before we proceed with harvesting the fruits of Theorem 2, we give some remarks on extensions and variations of the theorem, all substantiated in [9]:

- the theorem also provides insight on the expressive power of OMQs from the perspective of more traditional database query languages; in fact, coMMSNP can be seen as a notational variant of monadic disjunctive Datalog, which thus has the same expressive power as ($\mathcal{ALC}$, UCQ);
- $\mathcal{ALC}$ can be replaced with several other standard description logics such as $\mathcal{ELU}$, $\mathcal{ALCI}$ and $\mathcal{SHI}$, without invalidating the theorem;
- there are some standard features of DLs whose addition to $\mathcal{ALC}$ breaks Theorem 2, for example: (1) if $\mathcal{ALC}$ is extended with forms of counting such as funtional roles or number restrictions, then the resulting class of OMQs provably has no dichotomy between PTime and coNP; (2) the extension of $\mathcal{ALC}$ with transitive roles increases the expressive power beyond coCSP/coMMSNP, but it is not known whether the dichotomy holds or fails;
- while the equivalences given in Theorem 2 are effective, there are substantial differences in succinctness; whereas the translation from coCSP/coMMSNP to OMQs is polynomial, the converse translation is exponential and must be superpolynomial unless ExpTime $\subseteq$ coNP/poly; for OMQs based on the mild extension $\mathcal{ALCI}$ of $\mathcal{ALC}$, only a double exponential translation to coCSP/coMMSNP is known;
- argueably, the practically most important query languages are AQs and CQs; OMQs based on AQs correspond to a slightly generalized (and well-understood) form of CSPs with multiple templates and a single constant symbol. There is indeed no known natural counterpart to OMQs based on conjunctive queries on the CSP/MMSNP side.

An important and very active topic of OMQ research is to rewrite OMQs into equivalent queries that are formulated in traditional database query languages, in this way enabling the use of conventional database systems for efficient OMQ answering. Particularly important target query languages include FO (aka SQL) and Datalog. Rewriting into these languages is not always possible, witnessed for example by the OMQ from Section 2 that expresses non-3-colorability, which can be expressed neither in FO nor in Datalog [1]. However, the simple structure of real-world ontologies gives hope that rewriting will be possible in many practically relevant cases, and there is experimental evidence supporting this hope [23, 39, 40]. Ideally, one would like to have an approach for rewritings OMQs that is complete in the sense that it finds a (preferably small and simple) rewriting if there is one and otherwise reports non-existance. For OMQ languages such as ($\mathcal{ALC}$, AQ) and ($\mathcal{ALC}$, UCQ), this is non-trivial to attain. Interesting initial results can be carried over from the CSP world.

It is known that FO-definability and Datalog-definability of coCSPs are decidable [28, 7, 21]. These results can be lifted to multi-template CSPs with a single constant symbol [9].

Theorem 2 and its adaptation to $(\mathcal{ALC}, \mathrm{AQ})$ thus yields the upper bounds in the following theorem; the lower bounds are established by reductions from a tiling problem.

▶ **Theorem 3** ([9]). *In $(\mathcal{ALC}, BAQ)$ and $(\mathcal{ALC}, AQ)$, FO-rewritability and Datalog-rewritability are decidable and* NExpTime*-complete.*

In principle, the techniques in [28] also allow the construction of concrete FO-rewritings, and the result from [7] that Datalog-definability of coCSPs implies definability by Datalog programs of width three together with the canonical Datalog programs constructed by Feder and Vardi in [19] give a way to construct concrete Datalog-rewritings. However, naively applying such approaches will hardly be practical. More insight into the shape and construction of rewritings might be gained from the theory of obstructions, which have received significant attention in the CSP world.

A set of relational structures Γ is an *obstruction set* for a relational structure T (all in signature Σ) if for all finite Σ-structures S, we have $S \not\rightarrow T$ if and only if $O \rightarrow S$ for some $O \in \Gamma$. Obstructions are important because if an OMQ Q is equivalent to the complement of CSP(T) and Γ is an obstruction set of T, then $\bigvee_{O \in \Gamma} q_O$ is a (potentially infinitary) rewriting of Q where q_O is the Boolean CQ whose graph is O. Therefore, the CSP result that FO-definability of coCSP(T) implies the existence of finite set of finite tree-shaped obstructions for T [4, 28] translates into the OMQ result that any FO-rewritable OMQ has an FO-rewriting which is a UCQ that consists of tree-shaped CQs. Such results can potentially be used to guide the design of algorithm that construct rewritings and to study the succinctness of rewritings. Many other results about obstructions are available. For example, Datalog-rewritability is related to the existence of (an infinite set of) obstructions of bounded treewidth [19]. It is interesting to note that allowing answer variables in OMQs (which is roughly the same as extending CSPs with constants) changes the shape of obstructions in non-trivial ways, see [2, 9].

Theorem 3 only mentions atomic queries, but neither CQs nor UCQs. In fact, FO-definability and Datalog-definability of the complements of MMSNP sentences does not seem to have been studied in the CSP literature. We have recently observed that the problem is harder than for CSPs.

▶ **Theorem 4** ([13]). *In $(\mathcal{ALC}, CQ)$ and in coMMSNP, FO-rewritability and Datalog-rewritability are* 2NExpTime*-hard.*

In very recent (and yet unpublished) work with Cristina Feier, we were able to show that FO-rewritability and monadic Datalog rewritability are decidable for $(\mathcal{ALC}, \mathrm{CQ})$ and coMMSNP, with a 2NExpTime upper bound.

5 Summary

The connection between OMQs and CSPs brings interesting new results and techniques for OMQs. It also provides additional motivation for studying CSPs and underlines the importance of MMSNP which, despite having been the subject of some very interesting studiessuch as [32, 11, 12], has so far received much less attention than CSP. The motivation provided by the OMQ connection does not even stop at MMSNP. Some natural OMQ languages such as (GF, UCQ) with GF denoting the guarded fragment of FO have the same expressive power as an extension of MMSNP known as MMSNP_2 or GMSNP. Intuititively, the transition from MMSNP to GMSNP corresponds to replacing monadicity with guardedness, which increases expressive power [9]. Very little is known about the computational properties of this extended language.

There are other classes of database problems that are potentially quite closely connected to CSPs and related formalisms. In principle, it is interesting to consider the CSP connection for all querying problems that are, implicitly or explicitly, based on a certain answers semantics. One example is view-based query processing, for which a CSP connection has been established in [15]. Another one is *consistent query answering (CQA)*, where a query is answered over a set of databases that emerges from repairing a given database which violates its integrity constraints [3]. First observations on the connection between CSP and CQA have been made in [20, 31].

References

1 Foto N. Afrati, Stavros S. Cosmadakis, and Mihalis Yannakakis. On datalog vs. polynomial time. *J. Comput. Syst. Sci.*, 51(2):177–196, 1995.

2 Bogdan Alexe, Balder ten Cate, Phokion G. Kolaitis, and Wang Chiew Tan. Characterizing schema mappings via data examples. *ACM Trans. Database Syst.*, 36(4):23, 2011.

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. *Eur. J. Comb.*, 29(4):796–820, 2008.

5 Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of IJCAI*, pages 364–369, 2005.

6 Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2nd edition, 2010.

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

8 Meghyn Bienvenu and Magdalena Ortiz. Ontology-mediated query answering with data-tractable description logics. In *Proc. of Reasoning Web*, volume 9203 of *LNCS*, pages 218–307. Springer, 2015.

9 Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. *ACM Trans. Database Syst.*, 39(4):33:1–33:44, 2014.

10 Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge University Press, 2001.

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 Víctor Dalmau. Datalog and constraint satisfaction with infinite templates. *J. Comput. Syst. Sci.*, 79(1):79–100, 2013.

13 Pierre Bourhis and Carsten Lutz. Containment in monadic disjunctive datalog, mmsnp, and expressive description logics. *Submitted*, 2016.

14 Andrea Calì, Georg Gottlob, Thomas Lukasiewicz, and Andreas Pieris. Datalog+/-: A family of languages for ontology querying. In *Proc. of Datalog Reloaded*, volume 6702 of *LNCS*, pages 351–368. Springer, 2010.

15 Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. View-based query processing and constraint satisfaction. In *Proc. of LICS*, pages 361–371. IEEE Computer Society, 2000.

16 Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, and Riccardo Rosati. Ontologies and databases: The DL-Lite approach. In *Reasoning Web*, volume 5689 of *LNCS*, pages 255–356, 2009.

17 Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

18 Tomás Feder. Classification of homomorphisms to oriented cycles and of k-partite satisfiability. *SIAM J. Discrete Math.*, 14(4):471–480, 2001.

19 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.

20 Gaëlle Fontaine. Why is it hard to obtain a dichotomy for consistent query answering? *ACM Trans. Comput. Log.*, 16(1):7:1–7:24, 2015.

21 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`, 2009.

22 Valentin Goranko and Martin Otto. *Handbook of Modal Logic*, chapter Model Theory of Modal Logic, pages 255–325. Elsevier, 2006.

23 Peter Hansen, Carsten Lutz, Inanç Seylan, and Frank Wolter. Efficient query rewriting in the description logic $\mathcal{EL}$ and beyond. In *Proc. of IJCAI*, pages 3034–3040. AAAI Press, 2015.

24 Pavol Hell and Jaroslav Nesetril. On the complexity of H-coloring. *J. Comb. Theory, Ser. B*, 48(1):92–110, 1990.

25 Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: the making of a web ontology language. *J. Web Sem.*, 1(1):7–26, 2003.

26 Roman Kontchakov and Michael Zakharyaschev. An introduction to description logics and query rewriting. In *Proc. of Reasoning Web*, volume 8714 of *LNCS*, pages 195–244. Springer, 2014.

27 Richard E. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171, 1975.

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

29 Carsten Lutz, Robert Piro, and Frank Wolter. Description logic TBoxes: Model-theoretic characterizations and rewritability. In *Proc. of IJCAI*, pages 983–988, 2011.

30 Carsten Lutz and Frank Wolter. Non-uniform data complexity of query answering in description logics. In *Proc. of KR*, 2012.

31 Carsten Lutz and Frank Wolter. On the relationship between consistent query answering and constraint satisfaction problems. In *Proc. of ICDT*, pages 363–379, 2015.

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

33 Marie-Laure Mugnier and Michaël Thomazo. An introduction to ontology-based query answering with existential rules. In *Proc. of Reasoning Web*, volume 8714 of *LNCS*, pages 245–278. Springer, 2014.

34 W3C OWL Working Group. *OWL 2 Web Ontology Language: Document Overview.* W3C Recommendation, 2009. Available at `http://www.w3.org/TR/owl2-overview/`.

35 W3C RDF Working Group. *RDF Primer.* W3C Recommendation, 2004. Available at `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/`.

36 Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence).* Elsevier Science Inc., 2006.

37 Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artif. Intell.*, 48(1):1–26, 1991.

38 Kent Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *J. of the Am. Med. Inf. Ass.*, 2000.

39 Despoina Trivela, Giorgos Stoilos, Alexandros Chortaras, and Giorgos B. Stamou. Optimising resolution-based rewriting algorithms for OWL ontologies. *J. Web Sem.*, 33:30–49, 2015.

40 Yujiao Zhou, Bernardo Cuenca Grau, Yavor Nenov, Mark Kaminski, and Ian Horrocks. Pagoda: Pay-as-you-go ontology query answering using a datalog reasoner. *J. Artif. Intell. Res. (JAIR)*, 54:309–367, 2015.