Schema.org as a Description Logic

Andre Hernich[1], **Carsten Lutz**[2], **Ana Ozaki**[1] and **Frank Wolter**[1]
[1]University of Liverpool, UK
{andre.hernich, anaozaki, wolter}@liverpool.ac.uk
[2]University of Bremen, Germany
clu@informatik.uni-bremen.de

Abstract

Schema.org is an initiative by the major search engine providers Bing, Google, Yahoo!, and Yandex that provides a collection of ontologies which webmasters can use to mark up their pages. Schema.org comes without a formal language definition and without a clear semantics. We formalize the language of Schema.org as a Description Logic (DL) and study the complexity of querying data using (unions of) conjunctive queries in the presence of ontologies formulated in this DL (from several perspectives). While querying is intractable in general, we identify various cases in which it is tractable and where queries are even rewritable into FO queries or datalog programs.

1 Introduction

The Schema.org initiative was launched in 2011 and is supported today by Bing, Google, Yahoo!, and Yandex. In the spirit of the Semantic Web, it provides a collection of ontologies that establish a standard vocabulary to mark up website content with metadata (https://schema.org/). In particular, web content that is generated from structured data as found in relational databases is often difficult to recover for search engines and Schema.org markup elegantly solves this problem. The markup is used by search engines to more precisely identify relevant pages, to provide richer search results, and to enable new applications. Schema.org is experiencing very rapid adoption and is used today by more than 15 million webpages including all major ones [Guha, 2013].

Schema.org does neither formally specify the language in which its ontologies are formulated nor does it provide a formal semantics for the published ontologies. However, the provided ontologies are extended and updated frequently and follow an underlying language pattern. This pattern and its meaning is described informally in natural language. Schema.org adopts a class-centric representation enriched with binary relations and datatypes, similar in spirit to description logics (DLs) and to the OWL family of ontology languages; the current version includes 622 classes and 891 binary relations. Partial translations into RDF and into OWL are provided by the linked data community. Based on the informal descriptions at https://schema.org/ and on the mentioned translations, Patel-Schneider [2014] develops an ontology language for Schema.org with a formal syntax and semantics that, apart from some details, can be regarded as a fragment of OWL DL.

In this paper, we abstract slightly further and view the Schema.org ontology language as a DL, in line with the formalization by Patel-Schneider. Thus, what Schema.org calls a *type* becomes a concept name and a *property* becomes a role name. The main characteristics of the resulting 'Schema.org DL' are that (i) the language is very restricted, allowing only inclusions between concept and role names, domain and range restrictions, nominals, and datatypes; (ii) ranges and domains of roles can be restricted to *disjunctions* of concept names (possibly mixed with datatypes in range restrictions) and nominals are used in 'one-of enumerations' which also constitute a form of disjunction. While Point (i) suggests that the Schema.org DL is closely related to the tractable profiles of OWL2, because of Point (ii) it does actually not fall into any of them. There is a close connection to the DL-Lite family of DLs [Calvanese *et al.*, 2007], and in particular to the DL-Lite$^{\mathcal{H}}_{\mathsf{bool}}$ variant [Artale *et al.*, 2009]. However, DL-Lite$^{\mathcal{H}}_{\mathsf{bool}}$ admits existential restriction, negation, conjunction, and free use of disjunction whereas the Schema.org DL allows no existential quantification and includes nominals and datatypes. We use the term *schema.org-ontology* to refer to ontologies formulated in the Schema.org DL; in contrast, 'Schema.org 2015' refers to the concrete collection of ontologies provided at https://schema.org/ as of end of April, 2015.

Our main aim is to investigate the complexity of *querying data in the presence of schema.org-ontologies*, where the data is the markup that was extracted from webpages. While answering queries over such data is the main reasoning task that arises in Schema.org applications and the Schema.org initiative specifies a format for the data in terms of so-called *items*, no information is given on what form of querying is used. We consider conjunctive queries (CQs) and unions of conjunctive queries (UCQ), a basic querying mechanism that is ubiquitous in relational database systems and research, and that also can be viewed as a core of the Semantic Web query language SPARQL. In particular, we also consider CQs and UCQs without quantified variables since these are not allowed in the relevant SPARQL entailment regimes [Glimm and Krötzsch, 2010]. We often view a pair $(\mathcal{O}, q)$ that consists of a schema.org-ontology and an actual query as a compound query called an *ontology-mediated query (OMQ)*.

We start with the observation that evaluating OMQs is intractable in general, namely Π_2^p-complete in combined complexity and coNP-complete in data complexity. In the main part of the paper, we therefore have two aims: (i) identify large and practically useful classes of OMQs with lower combined and data complexity, and (ii) investigate in how far it is possible to obtain a *full classification* of each schema.org ontology or each OMQ according to its data complexity. While the utility of aim (i) is obvious, we note that aim (ii) is also most useful from a user's perspective as it clarifies the complexity of every *concrete ontology or OMQ* that might be used in an actual application. Apart from classical tractability (that is, PTime), we are particularly interested in the rewritability of OMQs into first-order (FO) queries (actually: UCQs) and into datalog programs. One reason is that this allows to implement querying based on relational database systems and datalog engines, taking advantage of those systems' efficiency and maturity. Another reason is that there is significant research on how to efficiently answer UCQs and datalog queries in cluster computing models such as MapReduce [Afrati and Ullman, 2011; 2012], a natural framework when processing web-scale data.

For both aims (i) and (ii) above, we start with analyzing *basic* schema.org ontologies in which enumeration definitions ('one of' expressions) and datatypes are disallowed. Regarding aim (i), we show that all OMQs which consist of a basic schema.org-ontology and a CQ q of qvar-size two (the restriction of q to quantified variables is a disjoint union of queries with at most two variables each) are datalog-rewritable in polynomial time and can be evaluated in PTime in combined complexity. This result trivially extends to basic schema.org-ontologies *with* datatypes, but does not hold for unrestricted schema.org-ontologies. In the latter case, we establish the same tractability results for OMQs with CQs that do not contain any quantified variables.

Regarding aim (ii), we start with classifying each single schema.org-ontology $\mathcal{O}$ according to the data complexity of *all* OMQs $(\mathcal{O}, q)$ with q a UCQ. We establish a dichotomy between AC^0 and coNP in the sense that for each ontology $\mathcal{O}$, either all these OMQs are in AC^0 or there is one OMQ that is coNP-hard. The dichotomy comes with a transparent syntactic characterization and is decidable in PTime. Though beautiful, however, it is of limited use in practice since most interesting ontologies are of the intractable kind. Therefore, we also consider an even more fine-grained classification on the level of OMQs, establishing a useful connection to constraint satisfaction problems (CSPs) in the spirit of [Bienvenu *et al.*, 2014b]. It turns out that even for basic schema.org-ontologies and for ontologies that consist exclusively of enumeration definitions, a complexity classification of OMQs implies a solution to the dichotomy conjecture for CSPs, a famous open problem [Feder and Vardi, 1998; Bulatov, 2011]. However, the CSP connection can also be used to obtain positive results. In particular, we show that it is decidable in NExpTime whether an OMQ based on a schema.org-ontology and a restricted form of UCQ is FO-rewritable and, respectively, datalog-rewritable. We also establish a PSpace lower bound for this problem.

Detailed proofs are provided in the full version at http://cgi.csc.liv.ac.uk/~frank/publ/publ.html.

2 Preliminaries

Let $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ be countably infinite and mutually disjoint sets of *concept names*, *role names*, and *individual names*. Throughout the paper, concepts names will be denoted by $A, B, C, \ldots$, role names by $r, s, t, \ldots$, and individual names by $a, b, c, \ldots$.

A schema.org-ontology consists of concept inclusions of different forms, role inclusions, and enumeration definitions. A *concept inclusion* takes the form $A \sqsubseteq B$ (*atomic concept inclusion*), $\mathsf{ran}(r) \sqsubseteq A_1 \sqcup \cdots \sqcup A_n$ (*range restriction*), or $\mathsf{dom}(r) \sqsubseteq A_1 \sqcup \cdots \sqcup A_n$ (*domain restriction*). A *role inclusion* takes the form $r \sqsubseteq s$.

Example 1. *The following are examples of concept inclusions and role inclusions (last line) in Schema.org 2015:*

$$\begin{aligned}
\mathsf{Movie} &\sqsubseteq \mathsf{CreativeWork}\\
\mathsf{ran}(\mathsf{musicBy}) &\sqsubseteq \mathsf{Person} \sqcup \mathsf{MusicGroup}\\
\mathsf{dom}(\mathsf{musicBy}) &\sqsubseteq \mathsf{Episode} \sqcup \mathsf{Movie} \sqcup \mathsf{RadioSeries} \sqcup \mathsf{TVSeries}\\
\mathsf{sibling} &\sqsubseteq \mathsf{relatedTo}
\end{aligned}$$

We now define enumeration definitions. Fix a set $\mathsf{N_E} \subseteq \mathsf{N_I}$ of *enumeration individuals* such that both $\mathsf{N_E}$ and $\mathsf{N_I} \setminus \mathsf{N_E}$ are infinite. An *enumeration definition* takes the form $A \equiv \{a_1, \ldots, a_n\}$ with $A \in \mathsf{N_C}$ and $a_1, \ldots, a_n \in \mathsf{N_E}$.

Example 2. *An enumeration definition in Schema.org 2015 is* $\mathsf{Booktype} \equiv \{\mathsf{ebook}, \mathsf{hardcover}, \mathsf{paperback}\}$.

A *datatype* $\mathcal{D} = (D, \Delta^{\mathcal{D}})$ consists of a *datatype name* D and a non-empty set of *data values* $\Delta^{\mathcal{D}}$. Examples of datatypes in Schema.org 2015 are Boolean, Integer, and Text. We assume that datatype names and data values are distinct from the symbols in $\mathsf{N_C} \cup \mathsf{N_R} \cup \mathsf{N_I}$ and that there is an arbitrary but fixed set DT of datatypes such that $\Delta^{\mathcal{D}_1} \cap \Delta^{\mathcal{D}_2} = \emptyset$ for all $\mathcal{D}_1 \neq \mathcal{D}_2 \in \mathsf{DT}$.

To accommodate datatypes in ontologies, we generalize range restrictions to *range restrictions with datatypes*, which are inclusions of the form $\mathsf{ran}(r) \sqsubseteq A_1 \sqcup \cdots \sqcup A_n$ with $A_1, \ldots, A_n$ concept names or datatype names from DT.

Example 3. *A range restriction with datatypes in Schema.org 2015 is* $\mathsf{ran}(\mathsf{acceptsReservation}) \sqsubseteq \mathsf{Boolean} \sqcup \mathsf{Text}$

A *schema.org-ontology* $\mathcal{O}$ is a finite set of concept inclusions (including range restrictions with datatypes), role inclusions, and enumeration definitions. We denote by $\mathsf{N_C}(\mathcal{O})$ the set of concept names in $\mathcal{O}$, by $\mathsf{N_R}(\mathcal{O})$ the set of role names in $\mathcal{O}$, and by $\mathsf{N_E}(\mathcal{O})$ the set of enumeration individuals in $\mathcal{O}$.

A *data instance* $\mathcal{A}$ is a finite set of

- *concept assertions* $A(a)$ where $A \in \mathsf{N_C}$ and $a \in \mathsf{N_I}$;
- *role assertions* $r(a, b)$ where $r \in \mathsf{N_R}$, $a \in \mathsf{N_I}$ and $b \in \mathsf{N_I} \cup \bigcup_{\mathcal{D} \in \mathsf{DT}} \Delta^{\mathcal{D}}$.

We say that $\mathcal{A}$ is a data instance *for the ontology* $\mathcal{O}$ if $\mathcal{A}$ contains no enumeration individuals except those in $\mathsf{N_E}(\mathcal{O})$. We use $\mathsf{Ind}(\mathcal{A})$ to denote the set of all individuals (including datatype elements) in $\mathcal{A}$.

Example 4. *Examples for assertions are* $\mathsf{Movie}(a)$, $\mathsf{name}(a, \text{'avatar'})$, $\mathsf{director}(a, b)$, $\mathsf{name}(b, \text{'Cam'})$.

Let $\mathcal{O}$ be a schema.org-ontology. An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ *for* $\mathcal{O}$ consists of a non-empty set $\Delta^\mathcal{I}$ disjoint from $\bigcup_{\mathcal{D}\in\mathsf{DT}} \Delta^\mathcal{D}$ and with $\Delta^\mathcal{I} \cap \mathsf{N_E} = \mathsf{N_E}(\mathcal{O})$, and a function $\cdot^\mathcal{I}$ that maps

- every concept name A to a subset $A^\mathcal{I}$ of $\Delta^\mathcal{I}$,
- every role name r to a subset $r^\mathcal{I}$ of $\Delta^\mathcal{I} \times \Delta^{\mathcal{I},\mathsf{DT}}$, where $\Delta^{\mathcal{I},\mathsf{DT}} = \Delta^\mathcal{I} \cup \bigcup_{\mathcal{D}\in\mathsf{DT}} \Delta^\mathcal{D}$;
- every individual name $a \in (\mathsf{N_I} \setminus \mathsf{N_E}) \cup \mathsf{N_E}(\mathcal{O})$ to some $a^\mathcal{I} \in \Delta^\mathcal{I}$ such that $a^\mathcal{I} = a$ for all $a \in \mathsf{N_E}(\mathcal{O})$.

Note that we make the standard name assumption (and, therefore, unique name assumption) for individuals in $\mathsf{N_E}$. Individual names from $\mathsf{N_E}$ that do not occur in $\mathcal{O}$ are not interpreted by $\mathcal{I}$ to avoid enforcing infinite domains.

For an interpretation $\mathcal{I}$ and role name r, set $\mathsf{dom}(r)^\mathcal{I} = \{d \mid (d,d') \in r^\mathcal{I}\}$ and $\mathsf{ran}(r)^\mathcal{I} = \{d' \mid (d,d') \in r^\mathcal{I}\}$. To achieve uniform notation, set $D^\mathcal{I} = \Delta^\mathcal{D}$ for every datatype $(D, \Delta^\mathcal{D})$ in DT and $d^\mathcal{I} = d$ for every $d \in \Delta^\mathcal{D}$, $\mathcal{D} \in \mathsf{DT}$. For concept or datatype names $A_1, \dots, A_n$, set $(A_1 \sqcup \dots \sqcup A_n)^\mathcal{I} = A_1^\mathcal{I} \cup \dots \cup A_n^\mathcal{I}$. An interpretation $\mathcal{I}$ for an ontology $\mathcal{O}$ *satisfies* a (concept or role) inclusion $X_1 \sqsubseteq X_2 \in \mathcal{O}$ if $X_1^\mathcal{I} \subseteq X_2^\mathcal{I}$, an enumeration definition $A \equiv \{a_1, \dots, a_n\}$ if $A^\mathcal{I} = \{a_1, \dots, a_n\}$, a concept assertion $A(a)$ if $a^\mathcal{I} \in A^\mathcal{I}$, and a role assertion $r(a,b)$ if $(a^\mathcal{I}, b^\mathcal{I}) \in r^\mathcal{I}$. These satisfaction relationships are denoted with "$\models$", as in $\mathcal{I} \models X_1 \sqsubseteq X_2$.

An interpretation $\mathcal{I}$ for $\mathcal{O}$ is a *model* of $\mathcal{O}$ if it satisfies all inclusions and definitions in $\mathcal{O}$ and a *model* of a data instance $\mathcal{A}$ for $\mathcal{O}$ if it satisfies all assertions in $\mathcal{A}$. We say that $\mathcal{A}$ is *satisfiable* w.r.t. $\mathcal{O}$ if $\mathcal{O}$ and $\mathcal{A}$ have a common model. Let α be a concept or role inclusion, or an enumeration definition. We say that α *follows from* $\mathcal{O}$, in symbols $\mathcal{O} \models \alpha$, if every model of $\mathcal{O}$ satisfies α.

We introduce the query languages considered in this paper. A *term* t is either a member of $\mathsf{N_I} \cup \bigcup_{\mathcal{D}\in\mathsf{DT}} \Delta^\mathcal{D}$ or an *individual variable* taken from an infinite set $\mathsf{N_V}$ of such variables. A *first-order query (FOQ)* consist of a (domain-independent) first-order formula $\varphi(\vec{x})$ that uses unary predicates from $\mathsf{N_C} \cup \{D \mid (D,\mathcal{D}) \in \mathsf{DT}\}$, binary predicates from $\mathsf{N_R}$, and only terms as introduced above. The unary datatype predicates are built-ins that identify the elements of the respective datatype. We call $\vec{x}$ the *answer variables* of $\varphi(\vec{x})$, the remaining variables are called *quantified*. A query without answer variables is *Boolean*. A *conjunctive query (CQ)* is a FOQ of the form $\exists\vec{y}\,\varphi(\vec{x},\vec{y})$ where $\varphi(\vec{x},\vec{y})$ is a conjunction of atoms such that every answer variable x occurs in an atom that uses a symbol from $\mathsf{N_C} \cup \mathsf{N_R}$, that is, an answer variable x is not allowed to occur exclusively in atoms of the form $D(x)$ with D a datatype name (to ensure domain independence). A *union of conjunctive queries (UCQ)* is a disjunction of CQs. A CQ q can be regarded as a directed graph G^q with vertices $\{t \mid t \text{ term in } q\}$ and edges $\{(t,t') \mid r(t,t') \text{ in } q\}$. If G^q is acyclic and $r(t_1,t_2), s(t_1,t_2) \in q$ implies $r = s$, then q is an *acyclic CQ*. A UCQ is *acyclic* if all CQs in it are.

We are interested in querying data instances $\mathcal{A}$ using a UCQ $q(\vec{x})$ taking into account the knowledge provided by an ontology $\mathcal{O}$. A *certain answer to* $q(\vec{x})$ *in* $\mathcal{A}$ *under* $\mathcal{O}$ is a tuple $\vec{a}$ of elements of $\mathsf{Ind}(\mathcal{A})$ of the same length as $\vec{x}$ such that for every model $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{A}$, we have $\mathcal{I} \models q[\vec{a}]$. In this case, we write $\mathcal{O}, \mathcal{A} \models q(\vec{a})$.

Query evaluation is the problem to decide whether $\mathcal{O}, \mathcal{A} \models q(\vec{a})$. For the combined complexity of this problem, all of $\mathcal{O}, \mathcal{A}, q$, and $\vec{a}$ are the input. For the data complexity, only $\mathcal{A}$ and $\vec{a}$ are the input while $\mathcal{O}$ and q are fixed. It often makes sense to combine the ontology $\mathcal{O}$ and actual query $q(\vec{x})$ into an *ontology-mediated query (OMQ)* $Q = (\mathcal{O}, q(\vec{x}))$, which can be thought of as a compound overall query. We show the following by adapting techniques from [Eiter *et al.*, 1997] and [Bienvenu *et al.*, 2014b].

Theorem 5. *Query evaluation of CQs and UCQs under schema.org-ontologies is* Π_2^p*-complete in combined complexity. In data complexity, each OMQ* $(\mathcal{O}, q)$ *from this class can be evaluated in* coNP*; moreover, there is such a OMQ (with* q *a CQ) that is* coNP*-complete in data complexity.*

An OMQ $(\mathcal{O}, q(\vec{x}))$ is *FO-rewritable* if there is a FOQ $Q(\vec{x})$ (called an *FO-rewriting* of $(\mathcal{O}, q(\vec{x}))$) such that for every data instance $\mathcal{A}$ for $\mathcal{O}$ and all $\vec{a} \in \mathsf{Ind}(\mathcal{A})$, we have $\mathcal{O}, \mathcal{A} \models q(\vec{a})$ iff $\mathcal{I}_\mathcal{A} \models Q(\vec{a})$ where $\mathcal{I}_\mathcal{A}$ is the interpretation that corresponds to $\mathcal{A}$ (in the obvious way). We also consider *datalog-rewritability*, defined in the same way as FO-rewritability, but using datalog programs in place of FOQs. Using Rossman's homomorphism preservation theorem [Rossman, 2008], one can show that an OMQ $(\mathcal{O}, q(\vec{x}))$ with $\mathcal{O}$ a schema.org-ontology and $q(\vec{x})$ a UCQ is FO-rewritable iff it has a UCQ-rewriting iff it has a non-recursive datalog rewriting, see [Bienvenu *et al.*, 2014b] for more details. Since non-recursive datalog-rewritings can be more succinct than UCQ-rewritings, we will generally prefer the former.

3 Basic schema.org-Ontologies

We start with considering *basic* schema.org-ontologies, which are not allowed to contain enumeration definitions and datatypes. The results obtained for basic schema.org-ontologies can be easily extended to basic schema.org-ontologies with datatypes but do not hold for ontologies with enumeration definitions (as will be shown in the next section). In Schema.org 2015, 45 concept names from a total of 622 are defined using enumeration definitions, and hence are not covered by the results presented in this section.

We start with noting that the *entailment problem for basic schema.org-ontologies* is decidable in polynomial time. This problem is to check whether $\mathcal{O} \models \alpha$ for a given basic schema.org-ontology $\mathcal{O}$ and a given inclusion α of the form allowed in such ontologies. In fact, the algorithm is straightforward. For example, $\mathcal{O} \models \mathsf{ran}(r) \sqsubseteq A_1 \sqcup \dots \sqcup A_n$ if there is a role name s and a range restriction $\mathsf{ran}(s) \sqsubseteq B_1 \sqcup \dots \sqcup B_m \in \mathcal{O}$ such that $\mathcal{O}_R \models r \sqsubseteq s$ and $\mathcal{O}_C \models B_j \sqsubseteq A_1 \sqcup \dots \sqcup A_n$ for all $1 \leq j \leq m$, where $\mathcal{O}_R$ and $\mathcal{O}_C$ denote the set of role inclusions and atomic concept inclusions in $\mathcal{O}$.

Theorem 6. *The entailment problem for basic schema.org-ontologies is in* PTIME.

The hardness results reported in Theorem 5 crucially rely on existential quantification in the actual query. In fact, it follows from results in [Grau *et al.*, 2013; Kaminski *et al.*, 2014b] that given an OMQ $Q = (\mathcal{O}, q(\vec{x}))$ with $\mathcal{O}$ a basic schema.org-ontology and $q(\vec{x})$ a CQ without quantified variables, it is possible to construct a non-recursive datalog rewriting of Q

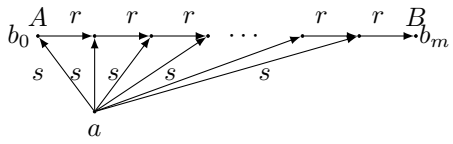

Figure 1: Data instance $\mathcal{A}_m$.

in polynomial time, and thus such OMQs can be evaluated in PTIME in combined complexity. We aim to push this bound further by admitting restricted forms of quantification.

A CQ q has *qvar-size* n if the restriction of q to quantified variables is a disjoint union of queries with at most n variables each. For example, quantifier-free CQs have qvar-size 0 and the following query $q(x, y)$ has qvar-size 1:

$$\exists z_1 \exists z_2 \bigwedge_{v \in \{x,y\}} (\mathsf{producedBy}(z_1, v) \wedge \mathsf{musicBy}(v, z_2))$$

The above consequences of the work by Grau, Kaminski, et al. can easily be extended to OMQs where queries have qvar-size one. In what follows, we consider qvar-size two, which is more subtle and where, in contrast to qvar-size one, reasoning by case distinction is required. The following example shows that there are CQs of qvar-size two for which no non-recursive datalog rewriting exists.

Example 7. *Let $\mathcal{O} = \{\mathsf{ran}(s) \sqsubseteq A \sqcup B\}$ and consider the following CQ of qvar-size two:*

$$q(x) = \exists x_1 \exists x_2 (s(x, x_1) \wedge A(x_1) \wedge r(x_1, x_2) \wedge B(x_2))$$

It is easy to see that $\mathcal{O}, \mathcal{A}_m \models q(a)$ for every data instance $\mathcal{A}_m$ with $m \geq 2$ as defined in Figure 1.

By applying locality arguments and using the data instances $\mathcal{A}_m$, one can in fact show that $(\mathcal{O}, q(x))$ is not FO-rewritable (note that removing one $r(b_i, b_{i+1})$ from $\mathcal{A}_m$ results in $q(a)$ being no longer entailed).

Theorem 8. *For every OMQ $(\mathcal{O}, q(\vec{x}))$ with $\mathcal{O}$ a basic schema.org-ontology and $q(\vec{x})$ a CQ of qvar-size at most two, one can construct a datalog-rewriting in polynomial time. Moreover, evaluating OMQs from this class is in* PTIME *in combined complexity.*

Applied to Example 7, the proof of Theorem 8 yields a datalog rewriting that consists of the rules

$$P(x_1, x_2, x) \leftarrow s(x, x_1) \wedge X_1(x_1) \wedge r(x_1, x_2) \wedge X_2(x_2)$$

where the X_i range over A, B, and $\exists y\, r(y, \cdot)$, plus

$$\begin{aligned}
I_A(x_1, x) &\leftarrow P(x_1, x_2, x) \wedge A(x_1) \\
I_B(x_2, x) &\leftarrow P(x_1, x_2, x) \wedge B(x_2) \\
I_A(x_2, x) &\leftarrow P(x_1, x_2, x) \wedge I_A(x_1, x) \\
I_B(x_1, x) &\leftarrow P(x_1, x_2, x) \wedge I_B(x_2, x) \\
\mathsf{goal}(x) &\leftarrow s(x, x_1) \wedge I_A(x_1, x) \wedge r(x_1, x_2) \wedge I_B(x_2, x).
\end{aligned}$$

The recursive rule for I_A (the one for I_B is dual) says that if the only option to possibly avoid a match for (x_1, x_2, x) is to color (x_1, x) with I_A, then the only way to possibly avoid a match for (x_1, x_2, x) is to color (x_2, x) with I_A (otherwise, since $\mathsf{ran}(s) \sqsubseteq A \sqcup B \in \mathcal{O}$, it would have to be colored with I_B which gives a match).

Theorem 8 can easily be extended to basic schema.org-ontologies enriched with datatypes. For schema.org-ontologies $\mathcal{O}$ that also contain enumeration definitions, the rewriting is sound but not necessarily complete, and can thus be used to compute approximate query answers.

Interestingly, Theorem 8 cannot be generalized to UCQs. This follows from the result shown in the full version that for basic schema.org-ontologies $\mathcal{O}$ and quantifier-free UCQs $q(x)$ (even without role atoms), the problem $\mathcal{O}, \mathcal{A} \models q(a)$ is coNP-hard regarding combined complexity for data instances $\mathcal{A}$ with a single individual a. We also note that it is not difficult to show (and follows from FO-rewritability of instance queries in DL-Lite$^{\mathcal{H}}_{\mathsf{bool}}$ [Artale *et al.*, 2009]) that given an OMQ $(\mathcal{O}, q(\vec{x}))$ with $\mathcal{O}$ a basic schema.org-ontology and $q(\vec{x})$ a *quantifier-free* UCQ, one can construct an FO-rewriting in exponential time, and thus query evaluation is in AC^0 in data complexity.

We now classify basic schema.org-ontologies $\mathcal{O}$ according to the data complexity of evaluating OMQs $(\mathcal{O}, q)$ with q a UCQ (or CQ). It is convenient to work with *minimized* ontologies where for all inclusions $F \sqsubseteq A_1 \sqcup \cdots \sqcup A_n \in \mathcal{O}$ and all $i \leq n$, there is a model $\mathcal{I}$ of $\mathcal{O}$ and a $d \in \Delta^{\mathcal{I}}$ such that d *satisfies* $F \sqcap A_i \sqcap \bigsqcap_{j \neq i} \neg A_j$ (defined in the usual way). Every schema.org-ontology can be rewritten in polynomial time into an equivalent minimized one. We establish the following dichotomy theorem.

Theorem 9. *Let $\mathcal{O}$ be a minimized basic schema.org-ontology. If there exists $F \sqsubseteq A_1 \sqcup \cdots \sqcup A_n \in \mathcal{O}$ with $n \geq 2$, then there is a Boolean CQ q that uses only concept and role names from $\mathcal{O}$ and such that $(\mathcal{O}, q)$ is* coNP*-hard in data complexity. Otherwise, a given OMQ $(\mathcal{O}, q)$ with q a UCQ can be rewritten into a non-recursive datalog-program in polynomial time (and is thus in AC^0 in data complexity).*

The proof of the second part of Theorem 9 is easy: if there are no $F \sqsubseteq A_1 \sqcup \cdots \sqcup A_n \in \mathcal{O}$ with $n \geq 2$, then $\mathcal{O}$ essentially is already a non-recursive datalog program and the construction is straightforward. The proof of the hardness part is obtained by extending the corresponding part of a dichotomy theorem for $\mathcal{ALC}$-ontologies of depth one [Lutz and Wolter, 2012]. The main differences between the two theorems are that (i) for basic schema.org-ontologies, the dichotomy is decidable in PTIME (whereas decidability is open for $\mathcal{ALC}$), (ii) the CQs in coNP-hard OMQs use only concept and role names from $\mathcal{O}$ (this is not possible in $\mathcal{ALC}$), and (iii) the dichotomy is between AC^0 and coNP whereas for $\mathcal{ALC}$ OMQs can be complete for PTIME, NL, etc.

By Theorem 9, disjunctions in domain and range restrictions are the (only!) reason that query answering is non-tractable for basic schema.org-ontologies. In Schema.org 2015, 14% of all range restrictions and 20% of all domain restrictions contain disjunctions.

In Theorem 9, we have classified the data complexity of *ontologies*, quantifying over the actual queries. In what follows, we aim to classify the data complexity of every OMQ. This problem turns out to be much harder and, in fact, we show that a classification of the data complexity of OMQs based on basic schema.org-ontologies and UCQs implies a classification of constraint satisfaction problems according to their complexity

(up to FO-reductions), a famous open problem that is the subject of significant ongoing research [Feder and Vardi, 1998; Bulatov, 2011].

A *signature* is a set of concept and role names (also called *symbols*). Let $\mathcal{B}$ be a finite interpretation that interprets only the symbols from a finite signature Σ. The *constraint satisfaction problem CSP*($\mathcal{B}$) is to decide, given a data instance $\mathcal{A}$ over Σ, whether there is a homomorphism from $\mathcal{A}$ to $\mathcal{B}$. In this context, $\mathcal{B}$ is called the *template* of CSP($\mathcal{B}$).

Theorem 10. *For every template $\mathcal{B}$, one can construct in polynomial time an OMQ $(\mathcal{O}, q)$ with $\mathcal{O}$ a basic schema.org-ontology and q a Boolean acyclic UCQ such that the complement of CSP($\mathcal{B}$) and $(\mathcal{O}, q)$ are mutually FO-reducible.*

Theorem 18 below establishes the converse direction of Theorem 10 for unrestricted schema.org-ontologies and a large class of (acyclic) UCQs. From Theorem 18, we obtain a NEXPTIME-upper bound for deciding FO-rewritability and datalog-rewritability of a large class of OMQs (Theorem 19 below). It remains open whether this bound is tight, but we can show a PSPACE lower bound for FO-rewritability using a reduction of the word problem of PSPACE Turing machines. The proof uses the ontology $\mathcal{O}$ and data instances $\mathcal{A}_m$ from Example 7 and is similar to a PSPACE lower bound proof for FO-rewritability in consistent query answering [Lutz and Wolter, 2015] which is, in turn, based on a construction from [Cosmadakis *et al.*, 1988].

Theorem 11. *It is* PSPACE*-hard to decide whether a given OMQ $(\mathcal{O}, q)$ with $\mathcal{O}$ a basic schema.org-ontology and q a Boolean acyclic UCQ is FO-rewritable.*

4 Incoherence and Unsatisfiability

In the subsequent section, we consider unrestricted schema.org ontologies instead of basic ones, that is, we add back enumeration definitions and datatypes. The purpose of this section is to deal with a complication that arises from this step, namely the potential presence of inconsistencies.

A symbol $X \in \mathsf{N_C} \cup \mathsf{N_R}$ is *incoherent* in an ontology $\mathcal{O}$ if $X^{\mathcal{I}} = \emptyset$ for all models $\mathcal{I}$ of $\mathcal{O}$. An ontology $\mathcal{O}$ is *incoherent* if some symbol is incoherent in $\mathcal{O}$. The problem with incoherent ontologies $\mathcal{O}$ is that there are clearly data instances $\mathcal{A}$ that are unsatisfiable w.r.t. $\mathcal{O}$. Incoherent ontologies can result from the UNA for enumeration individuals such as in $\mathcal{O} = \{A \equiv \{a\}, B \equiv \{b\}, A \sqsubseteq B\}$, which has no model (if $a \neq b$) and thus any symbol is incoherent in $\mathcal{O}$; they can also arise from interactions between concept names and datatypes such as in $\mathcal{O}' = \{\mathsf{ran}(r) \sqsubseteq \mathsf{Integer}, \mathsf{ran}(s) \sqsubseteq A, r \sqsubseteq s\}$ with $A \in \mathsf{N_C}$, in which r is incoherent since $\Delta^{\mathcal{I}} \cap \Delta^{\mathsf{Integer}} = \emptyset$ in any model $\mathcal{I}$ of $\mathcal{O}'$. Using Theorem 6, one can show the following.

Theorem 12. *Incoherence of schema.org-ontologies can be decided in PTime.*

We now turn to inconsistencies that arise from combining an ontology $\mathcal{O}$ with a particular data instance $\mathcal{A}$ for $\mathcal{O}$. As an example, consider $\mathcal{O} = \{A \equiv \{a\}, B \equiv \{b\}\}$ and $\mathcal{A} = \{A(c), B(c)\}$. Although $\mathcal{O}$ is coherent, $\mathcal{A}$ is unsatisfiable w.r.t. $\mathcal{O}$. Like incoherence, unsatisfiability is decidable in polynomial time. In fact, we can even show the following stronger result.

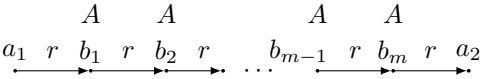

Figure 2: Data instance $\mathcal{A}'_m$.

Theorem 13. *Given a schema.org-ontology $\mathcal{O}$, one can compute in polynomial time a non-recursive datalog program Π such that for any data instance $\mathcal{A}$ for $\mathcal{O}$, $\mathcal{A}$ is unsatisfiable w.r.t. $\mathcal{O}$ iff $\Pi(\mathcal{A}) \neq \emptyset$.*

In typical schema.org applications, the data is collected from the web and it is usually not acceptable to simply report back an inconsistency and stop processing the query. Instead, one would like to take maximum advantage of the data despite the presence of an inconsistency. There are many semantics for inconsistent query answering that can be used for this purpose. As efficiency is paramount in schema.org applications, our choice is the pragmatic intersection repair (IAR) semantics which avoids CONP-hardness in data complexity [Lembo *et al.*, 2010; Rosati, 2011; Bienvenu *et al.*, 2014a]. A *repair* of a data instance $\mathcal{A}$ w.r.t. an ontology $\mathcal{O}$ is a maximal subset $\mathcal{A}' \subseteq \mathcal{A}$ that is satisfiable w.r.t. $\mathcal{O}$. We use $\mathsf{rep}_{\mathcal{O}}(\mathcal{A})$ to denote the set of all repairs of $\mathcal{A}$ w.r.t. $\mathcal{O}$. The idea of IAR semantics is then to replace $\mathcal{A}$ with $\bigcap_{\mathcal{A}' \in \mathsf{rep}_{\mathcal{O}}(\mathcal{A})} \mathcal{A}'$. In other words, we have to remove from $\mathcal{A}$ all assertions that occur in some minimal subset $\mathcal{A}' \subseteq \mathcal{A}$ that is unsatisfiable w.r.t. $\mathcal{O}$. We call such an assertion a *conflict assertion*.

Theorem 14. *Given a schema.org-ontology $\mathcal{O}$ and concept name A (resp. role name r), one can compute a non-recursive datalog program Π such that for any data instance $\mathcal{A}$ for $\mathcal{O}$, $\Pi(\mathcal{A})$ is the set of all $a \in \mathsf{Ind}(\mathcal{A})$ (resp. $(a, b) \in \mathsf{Ind}(\mathcal{A})^2$) such that $A(a)$ (resp. $r(a, b)$) is a conflict assertion in $\mathcal{A}$.*

By Theorem 14, we can adopt the IAR semantics by simply removing all conflict assertions from the data instance before processing the query. Programs from Theorem 14 become exponential in the worst case, but we expect them to be small in practical cases. In the remainder of the paper, we assume that ontologies are coherent and that $\mathcal{A}$ is satisfiable w.r.t. $\mathcal{O}$ if we query a data instance $\mathcal{A}$ using an ontology $\mathcal{O}$.

5 Unrestricted schema.org-Ontologies

We aim to lift the results from Section 3 to unrestricted schema.org-ontologies. Regarding Theorem 8, it turns out that quantified variables in CQs are computationally much more problematic when there are enumeration definitions in the ontology. In fact, one can expect positive results only for quantifier-free CQs, and even then the required constructions are quite subtle.

Theorem 15. *Given an OMQ $Q = (\mathcal{O}, q)$ with $\mathcal{O}$ a schema.org-ontology and q a quantifier-free CQ, one can construct in polynomial time a datalog-rewriting of Q. Moreover, evaluating OMQs in this class is in* PTIME *in combined complexity. The rewriting is non-recursive if $q = A(x)$.*

The following example illustrates the construction of the datalog program. Let $\mathcal{O} = \{A \equiv \{a_1, a_2\}\}$ and $q() = r(a_1, a_2)$. Observe that $\mathcal{O}, \mathcal{A}'_m \models q()$ for every data instance $\mathcal{A}'_m$ defined in Figure 2. Similarly to Example 7, one can use the data instances $\mathcal{A}'_m$ to show that $(\mathcal{O}, q())$ is not FO-rewritable.

A datalog-rewriting of $(\mathcal{O}, q())$ is given by the program Π_{a_1,a_2} which contains the rules

$$\begin{aligned}
\mathsf{goal}() &\leftarrow r(a_1,a_2)\\
\mathsf{goal}() &\leftarrow r(a_1,x) \wedge \mathsf{path}_A(x,y) \wedge r(y,a_2)\\
\mathsf{path}_A(x,y) &\leftarrow r(x,y) \wedge A(x) \wedge A(y)\\
\mathsf{path}_A(x,y) &\leftarrow \mathsf{path}_A(x,z) \wedge \mathsf{path}_A(z,y).
\end{aligned}$$

Given a data instance $\mathcal{A}$, the program checks whether there is an r-path from a_1 to a_2 in $\mathcal{A}$ with inner nodes in A. If $b_0, b_1, \dots, b_n$ is such a path, then in all models $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{A}$ there is an $i < n$ with $(b_{i-1}^{\mathcal{I}}, b_i^{\mathcal{I}}) = (a_1, a_2)$, hence $\mathcal{I} \models q()$. Otherwise, we obtain a model $\mathcal{I}$ with $\mathcal{I} \not\models q()$ by assigning a_1 to all individual names b with $A(b) \in \mathcal{A}$ that are reachable from a_1 by a path with inner nodes in A, and an individual $\neq a_1$ to all other individual names in $\mathcal{A}$.

We now modify the datalog program to obtain a rewriting of the OMQ $(\mathcal{O}, q'(x,y))$ with $q(x,y) = r(x,y)$. First, we include in Π_r the rules $A(a_1) \leftarrow \mathsf{true}$, $A(a_2) \leftarrow \mathsf{true}$, and

$$\begin{aligned}
\mathsf{goal}(x,y) &\leftarrow r(x,y)\\
\mathsf{goal}(x,y) &\leftarrow A(x) \wedge A(y) \wedge \textstyle\bigwedge_{1\le i,j\le 2} R_{a_i,a_j}(x,y)
\end{aligned}$$

We want to use the latter rule to check that (1) in every model, x and y have to be identified with an individual in $\{a_1, a_2\}$, and (2) for all $i, j \in \{1,2\}$, all models that identify x and y with a_i and a_j satisfy $r(a_i, a_j)$. Notice that $r(x,y)$ is false in a model of $\mathcal{O}$ and $\mathcal{A}$ iff $\mathcal{A}$ does not contain $r(x,y)$ and (1) or (2) is violated. To implement (2), we add the rules:

$$\begin{aligned}
R_{a_i,a_j}(x,y) &\leftarrow \mathsf{neq}(x,a_i) \qquad R_{a_i,a_j}(x,y) \leftarrow \mathsf{neq}(y,a_j)\\
R_{a_i,a_j}(x,y) &\leftarrow \mathsf{goal}(a_i,a_j)\\
\mathsf{neq}(a_1,a_2) &\leftarrow \mathsf{true} \qquad\qquad \mathsf{neq}(a_2,a_1) \leftarrow \mathsf{true}.
\end{aligned}$$

The first row checks admissibility of the assignment $x, y \mapsto a_i, a_j$: if x is one of the enumeration individuals in $\{a_1, a_2\}$ and $a_i \neq x$, then there is no model that identifies x with a_i, hence the statement (2) above is trivially true. Similarly for y and a_j. It remains to add rules 3 and 4 from Π_{a_1,a_2} and

$$\mathsf{goal}(a_i,a_j) \leftarrow r(a_i,x) \wedge \mathsf{path}_A(x,y) \wedge r(y,a_j)$$

for $1 \le i, j \le 2$ and $i \neq j$.

Theorem 15 is tight in the sense that evaluating CQs with a single atom and a single existentially quantified variable, as well as quantifier-free UCQs, is coNP-hard in data complexity. For instance, let $\mathcal{O} = \{\mathsf{dom}(e) \sqsubseteq A,\ \mathsf{ran}(e) \sqsubseteq A,\ A \equiv \{r,g,b\}\}$. Then, an undirected graph $G = (V,E)$ is 3-colorable iff $\mathcal{O}, \{e(v,w) \mid (v,w) \in E\} \not\models \exists x\, e(x,x)$. Alternatively, one may replace the query by $r(r,r) \vee r(g,g) \vee r(b,b)$. In fact, one can prove the following variant of Theorem 10 which shows that classifying OMQs with ontologies using *only* enumeration definitions and quantifier-free UCQs according to their complexity is as hard as CSP.

Theorem 16. *Given a template $\mathcal{B}$, one can construct in polynomial time an OMQ $(\mathcal{O}, q)$ where $\mathcal{O}$ only contains enumeration definitions and q is a Boolean variable-free UCQ such that the complement of CSP($\mathcal{B}$) and $(\mathcal{O}, q)$ are mutually FO-reducible.*

We now turn to classifying the complexity of ontologies and of OMQs, starting with a generalization of Theorem 9 to unrestricted schema.org-ontologies.

Theorem 17. *Let $\mathcal{O}$ be a coherent and minimized schema.org-ontology. If $\mathcal{O}$ contains an enumeration definition $A \equiv \{a_1, \dots, a_n\}$ with $n \ge 2$ or contains an inclusion $F \sqsubseteq A_1 \sqcup \dots \sqcup A_n$ such that there are at least two concept names in $\{A_1, \dots, A_n\}$ and $\mathcal{O} \not\models F \sqsubseteq A \sqcup \bigsqcup_{(D,\Delta^{\mathcal{D}}) \in DT} D$ for any A with $A \equiv \{a\} \in \mathcal{O}$, then $(\mathcal{O}, q)$ is coNP-hard for some Boolean CQ q. Otherwise every $(\mathcal{O}, q)$ with q a UCQ is FO-rewritable (and thus in AC^0 in data complexity).*

Note that, in contrast to Theorem 9, being in AC^0 does not mean that no 'real disjunction' is available. For example, for $\mathcal{O} = \{\mathsf{ran}(r) \sqsubseteq A \sqcup B, A \sqsubseteq C, B \sqsubseteq C, C \equiv \{c\}\}$ and $\mathcal{A} = \{r(a,b)\}$ we have $\mathcal{O}, \mathcal{A} \models A(b) \vee B(b)$ and neither $A(b)$ nor $B(b)$ are entailed. This type of choice does not effect FO-rewritability, since it is restricted to individuals that must be identified with a unique individual in $\mathsf{N}_\mathsf{E}(\mathcal{O})$. Note that, for the hardness proof, we now need to use a role name that possibly does not occur in $\mathcal{O}$. For example, for $\mathcal{O} = \{A \equiv \{a_1, a_2\}\}$ there exists a Boolean CQ q such that $(\mathcal{O}, q)$ is NP-hard, but a fresh role name is required to construct q.

We now consider the complexity of single OMQs and show a converse of Theorems 10 and 16 for schema.org-ontologies and UCQs that are *qvar-acyclic*, that is, when all atoms $r(t,t')$ with neither of t, t' a quantified variable are dropped, then all CQs in it are acyclic. We use *generalized CSPs with marked elements* in which instead of a single template $\mathcal{B}$, one considers a finite set Γ of templates whose signature contains, in addition to concept and role names, a finite set of individual names. Homomorphisms have to respect also the individual names and the problem is to decide whether there is a homomorphism from the input interpretation to some $\mathcal{B} \in \Gamma$. It is proved in [Bienvenu *et al.*, 2014b] that there is a dichotomy between PTIME and NP for standard CSPs if, and only if, there is such a dichotomy for generalized CSPs with marked elements.

Theorem 18. *Given an OMQ $(\mathcal{O}, q)$ with $\mathcal{O}$ a schema.org-ontology and q a qvar-acyclic UCQ, one can compute in exponential time a generalized CSP with marked elements Γ such that $(\mathcal{O}, q)$ and the complement of CSP(Γ) are mutually FO-reducible.*

The proof uses an encoding of qvar-acyclic queries into concepts in the description logic $\mathcal{ALCIUO}$ that extends $\mathcal{ALC}$ by inverse roles, the universal role, and nominals. It extends the the template constructions of [Bienvenu *et al.*, 2014b] to description logics with nominals. It is shown in [Bienvenu *et al.*, 2014b] that FO-definability and datalog definability of the complement of generalized CSPs with marked elements are NP-complete problems. Thus, we obtain the following result as a particularly interesting consequence of Theorem 18.

Theorem 19. *FO-rewritability and datalog-rewritability of OMQs $(\mathcal{O}, q)$ with $\mathcal{O}$ a schema.org-ontology and q a qvar-acyclic UCQ are decidable in* NEXPTIME.

6 Practical Considerations

In this paper, we have introduced a novel description logic motivated by Schema.org and studied the complexity of the

resulting querying problems from various angles. From a practical perspective, a central observation is that intractability is caused by the combination of disjunction in the ontology (in domain/range restrictions and, with $\{a, b\} \equiv \{a\} \sqcup \{b\}$, in enumeration definitions) and quantification in the query. For practical feasibility, one thus has to tame the interaction between these features.

One may speculate that professional users of Schema.org such as the major search engine providers take a pragmatic approach and essentially ignore disjunction. However, the results in this paper show that one can do better without compromising tractability when the query contains no quantified variables (Theorem 15). For basic ontologies, it is even possible to handle some queries with quantified variables (Theorem 8); in fact, we believe that the restriction to qvar-size 2 is a mild one from a practical perspective. It is also interesting to observe that the datalog-rewritings constructed in the proofs of these two theorems are sound if applied to unrestricted CQs and can be seen as tractable approximations that go beyond simply ignoring disjunction.

Another practically interesting way to address intractability is to require suitable forms of completeness of the data. For example, whenever the data contains an assertion $r(a, b)$ and there is a range restriction $\mathsf{ran}(r) \sqsubseteq A_1 \sqcup \cdots \sqcup A_n$ in the ontology, one could require that $A_i(b)$ is also in the data, for some i. This could be easily implemented in existing Schema.org validators that webpage developers use to verify their annotations. If all disjunctions are 'disabled' in the described way, tractability is regained.

References

[Afrati and Ullman, 2011] F.N. Afrati and J.D. Ullman. Optimizing multiway joins in a map-reduce environment. *IEEE Trans. Knowl. Data Eng.*, 23(9):1282–1298, 2011.

[Afrati and Ullman, 2012] F.N. Afrati and J.D. Ullman. Transitive closure and recursive datalog implemented on clusters. In *EDBT*, pages 132–143, 2012.

[Artale *et al.*, 2009] A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The DL-Lite family and relations. *JAIR*, 36:1–69, 2009.

[Bienvenu *et al.*, 2013] M. Bienvenu, C. Lutz, and F. Wolter. First-order rewritability of atomic queries in Horn description logics. In *Proc. of IJCAI*, 2013.

[Bienvenu *et al.*, 2014a] M. Bienvenu, C. Bourgaux, and F. Goasdoué. Querying inconsistent description logic knowledge bases under preferred repair semantics. In *AAAI*, pages 996–1002, 2014.

[Bienvenu *et al.*, 2014b] M. Bienvenu, B. ten Cate, C. Lutz, and F. Wolter. Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. *ACM Trans. Database Syst.*, 39(4):33, 2014.

[Bulatov, 2011] A.A. Bulatov. On the CSP dichotomy conjecture. In *CSR*, pages 331–344, 2011.

[Calvanese *et al.*, 2007] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

[Cohen and Jeavons, 2006] D. Cohen and P. Jeavons. *The complexity of constraint languages*, Ch. 8. Elsevier, 2006.

[Cosmadakis *et al.*, 1988] S.S. Cosmadakis, H. Gaifman, P.C. Kanellakis, and M.Y. Vardi. Decidable optimization problems for database logic programs (preliminary report). In *STOC*, pages 477–490, 1988.

[Eiter *et al.*, 1997] T. Eiter, G. Gottlob, and H. Mannila. Disjunctive datalog. *ACM Trans. Database Syst.*, 22(3):364–418, 1997.

[Feder and Vardi, 1998] T. Feder and M.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.

[Glimm and Krötzsch, 2010] B. Glimm and M. Krötzsch. SPARQL beyond subgraph matching. In *ISWC*, volume 6496 of *LNCS*, pages 241–256. Springer, 2010.

[Grau *et al.*, 2013] B. Cuenca Grau, B. Motik, G. Stoilos, and I. Horrocks. Computing datalog rewritings beyond horn ontologies. In *IJCAI*, 2013.

[Guha, 2013] R.V. Guha. Light at the end of the tunnel? Invited Talk at ISWC 2013, https://www.youtube.com/watch?v=oFY-0QoxBi8.

[Kaminski *et al.*, 2014a] M. Kaminski, Y. Nenov, and B. Cuenca Grau. Computing datalog rewritings for disjunctive datalog programs and description logic ontologies. In *RR*, pages 76–91, 2014.

[Kaminski *et al.*, 2014b] M. Kaminski, Y. Nenov, and B. Cuenca Grau. Datalog rewritability of disjunctive datalog programs and its applications to ontology reasoning. In *AAAI*, pages 1077–1083, 2014.

[Larose and Tesson, 2009] B. Larose and P. Tesson. Universal algebra and hardness results for constraint satisfaction problems. *Theor. Comput. Sci.*, 410(18):1629–1647, 2009.

[Lembo *et al.*, 2010] D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D. Fabio Savo. Inconsistency-tolerant semantics for description logics. In *RR*, pages 103–117, 2010.

[Lutz and Wolter, 2012] C. Lutz and F. Wolter. Non-uniform data complexity of query answering in description logics. In *KR*, 2012.

[Lutz and Wolter, 2015] C. Lutz and F. Wolter. On the relationship between consistent query answering and constraint satisfaction problems. In *ICDT*, 2015.

[Patel-Schneider, 2014] P.F. Patel-Schneider. Analyzing schema.org. In *ISWC*, pages 261–276, 2014.

[Rosati, 2011] R. Rosati. On the complexity of dealing with inconsistency in description logic ontologies. In *IJCAI*, pages 1057–1062, 2011.

[Rossman, 2008] B. Rossman. Homomorphism preservation theorems. *J. ACM*, 55(3), 2008.

[Schaerf, 1993] A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *J. of Int. Infor. Sys.*, 689:508, 1993.