Schema.org as a Description Logic

Andre Hernich[1], **Carsten Lutz**[2], **Ana Ozaki**[1] and **Frank Wolter**[1]
[1]University of Liverpool, UK [2]University of Bremen, Germany
{andre.hernich, anaozaki, wolter}@liverpool.ac.uk 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* $\mathsf{Movie}(a)$, $\mathsf{name}(a, \text{'}\mathsf{avatar}\text{'})$, $\mathsf{director}(a, b)$, $\mathsf{name}(b, \text{'}\mathsf{Cam}\text{'})$.

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 $(A_1 \sqcup \cdots \sqcup A_n)^\mathcal{I} = A_1^\mathcal{I} \cup \cdots \cup A_n^\mathcal{I}$. 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, $\mathcal{O} \models \mathsf{ran}(r) \sqsubseteq A_1 \sqcup \cdots \sqcup A_n$ if there is a role name s and a range restriction $\mathsf{ran}(s) \sqsubseteq B_1 \sqcup \cdots \sqcup B_m \in \mathcal{O}$ such that $\mathcal{O}_R \models r \sqsubseteq s$ and $\mathcal{O}_C \models B_j \sqsubseteq A_1 \sqcup \cdots \sqcup A_n$ 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

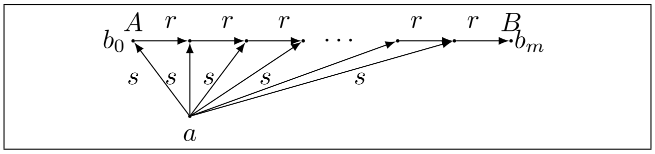

Figure 1: Data instance $\mathcal{A}_m$.

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* $F \sqcap A_i \sqcap \prod_{j \neq i} \neg A_j$ (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.

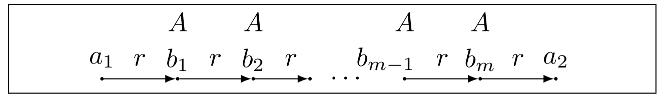

Figure 2: Data instance $\mathcal{A}'_m$.

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 $b_0, b_1, \ldots, b_n$ 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

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

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 $1 \leq i, j \leq 2$ 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 $A \equiv \{a_1, \ldots, a_n\}$ with $n \geq 2$ or contains an inclusion $F \sqsubseteq A_1 \sqcup \cdots \sqcup A_n$ such that there are at least two concept names in $\{A_1, \ldots, A_n\}$ 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 $\mathsf{N_E}(\mathcal{O})$. 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.

Appendix

A Proofs for Section 2

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.

Proof. The upper bounds are straightforward. For example, for the Π_2^p-upper bound regarding combined complexity, given a data instance $\mathcal{A}$ for $\mathcal{O}$ and $q(\vec{a})$, guess a model $\mathcal{I}$ with domain $\mathsf{Ind}(\mathcal{A})$, check in polynomial time whether $\mathcal{I}$ is a model of $\mathcal{O}$ and $\mathcal{A}$ and call an NP-oracle to check $\mathcal{I} \not\models q(\vec{a})$.

For the Π_2^p-lower bound, we give a reduction from 2QBF validity. Consider a 2QBF $\forall x_1 \dots x_m \exists y_1 \dots y_n \, \varphi$, where φ is a 3CNF over clauses $c_1, \dots, c_k$. We construct a schema.org-ontology $\mathcal{O}$ and Boolean CQ q with concept names $X_1, \dots, X_m$, and $C_1, \dots, C_k$, role names $V_1, V_2, V_3, r_1, \dots, r_m$, and enumeration individuals $\{0, 1\}$. For each clause c_i, we denote by v_i^j $(1 \leq j \leq 3)$ the variable appearing in the jth literal of c_i, and we let S_i denote the set of tuples in $\{0,1\}^3$ representing the seven truth assignments for (v_i^1, v_i^2, v_i^3) which satisfy c_i. Define the ontology $\mathcal{O}$ by setting

$$\begin{aligned} \mathcal{O} \;\; = \;\; & \{\mathsf{ran}(r_i) \sqsubseteq E, E \equiv \{0,1\} \mid 1 \leq i \leq m\} \cup \\ & \{\mathsf{ran}(r_i) \sqsubseteq X_i \mid 1 \leq i \leq m\} \end{aligned}$$

The ontology ensures that for any data instance $\mathcal{A}$ containing $r_i(f_i, d_i)$ in any model $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{A}$ we have $0 \in X_i^{\mathcal{I}}$ or $1 \in X_i^{\mathcal{I}}$. Thus, intuitively, $\mathcal{O}$ ensures that a truth assignment is selected for variable x_i. We encode φ using the data instance $\mathcal{A}_\varphi$ defined as follows:

$$\begin{aligned} \mathcal{A}_\varphi \;\; = \;\; & \{V_j(a_i^b, b_j) \mid b = (b_1, b_2, b_3) \in S_i, j = 1,2,3\} \cup \\ & \{C_i(a_i^b) \mid b = (b_1, b_2, b_3) \in S_i\} \cup \\ & \{r_i(f_i, d_i) \mid 1 \leq i \leq m\} \end{aligned}$$

Now the CQ q checks whether the selected truth assignment can be extended to a model of φ. q is defined as the conjunction of

$$\bigwedge_{1 \leq i \leq k} (C_i(z_i) \wedge V_1(z_i, v_i^1) \wedge V_2(z_i, v_i^2) \wedge V_3(z_i, v_i^3))$$

and

$$\bigwedge_{1 \leq \ell \leq m} X_\ell(x_\ell)$$

It is straightforward to show that $\forall x_1 \dots x_m \exists y_1 \dots y_n \, \varphi$ is valid iff $\mathcal{O}, \mathcal{A} \models q$. The NP lower bound regarding data complexity is proved, for example, in the CSP encoding of Theorem 10. ❑

B Proofs for Section 3

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

Proof. We show that $\mathcal{O} \models \mathsf{ran}(r) \sqsubseteq A_1 \sqcup \dots \sqcup A_n$ iff $(*)$ there exists a role name s such that $\mathcal{O}_R \models r \sqsubseteq s$ and a range restriction $\mathsf{ran}(s) \sqsubseteq B_1 \sqcup \dots \sqcup B_m \in \mathcal{O}$ such that $\mathcal{O}_C \models B_j \sqsubseteq A_1 \sqcup \dots \sqcup A_n$ for all $1 \leq j \leq m$, where $\mathcal{O}_C$ is the set of atomic concept inclusions in $\mathcal{O}$. Other basic schema.org-inclusions are consider similarly.

Clear, if $(*)$ holds, then $\mathcal{O} \models \mathsf{ran}(r) \sqsubseteq A_1 \sqcup \dots \sqcup A_n$. Conversely, assume that $(*)$ does not hold. Define an interpretation $\mathcal{I}$ with $\Delta^{\mathcal{I}} = \{a, b\}$ by setting

- $(c, d) \in s^{\mathcal{I}}$ iff $c = a$ and $d = b$ and $\mathcal{O}_R \models r \sqsubseteq s$;
- $c \in A^{\mathcal{I}}$ for all concept names A;
- for every s with $\mathcal{O}_R \models r \sqsubseteq s$ and $\mathsf{ran}(s) \sqsubseteq B_1 \sqcup \dots \sqcup B_m \in \mathcal{O}$ pick B_j such that $\mathcal{O}_C \not\models B_j \sqsubseteq A_1 \sqcup \dots \sqcup A_n$. Let $b \in B^{\mathcal{I}}$ for all concept names B with $\mathcal{O}_C \models B_j \sqsubseteq B$.

It is readily checked that $\mathcal{I}$ is a model of $\mathcal{O}$ with $b \in \mathsf{ran}(r)^{\mathcal{I}}$ and $b \notin A_1^{\mathcal{I}} \cup \dots \cup A_n^{\mathcal{I}}$. ❑

We use the following notation. A *match* π for a quantifier-free CQ $q = q(\vec{x}, \vec{a})$ in an interpretation $\mathcal{I}$ is a mapping from the set of terms $\mathsf{term}(q)$ of q to $\Delta^{\mathcal{I}}$ such that the following holds:

- $\pi(a) = a^{\mathcal{I}}$ for all $a \in \mathsf{N_I}$;
- If $A(t) \in q$, then $\pi(t) \in A^{\mathcal{I}}$;
- If $r(t, t') \in q$, then $(\pi(t), \pi(t') \in r^{\mathcal{I}}$.

If this is the case, we write $\mathcal{I} \models_\pi q$.

Given a data instance $\mathcal{A}$ and datalog program Π, we denote by $\mathcal{I}_{\Pi,\mathcal{A}}$ the *minimal interpretation* that is a model of $\mathcal{A}$ and satisfies all rules in Π. Note that $\Delta^{\mathcal{I}_{\Pi,\mathcal{A}}} = \mathsf{Ind}(\mathcal{A})$.

We prove the following result as a preparation for the proof of Theorem 8.

Proposition 20. *For every ontology-mediated query $(\mathcal{O}, q(\vec{x}))$ with $\mathcal{O}$ a basic schema.org-ontology and q a CQ of qvar-size at most one, one can construct in polynomial time a non-recursive datalog-rewriting of $(\mathcal{O}, q(\vec{x}))$.*

Proof. Let $q(\vec{x}) = \exists \vec{y} q_0(\vec{x}, \vec{y}, \vec{b})$, Let $\vec{x} = x_1 \dots x_k$, $\vec{y} = y_1, \dots, y_m$, and $\vec{b} = b_1, \dots, b_n$. Let I_A and I_r be IDB predicates for any concept name A and role name r in q, and include in $\Pi_{\mathcal{O},\mathsf{basic}}$ the following rules:

- $I_A(x) \leftarrow B(x)$, for all B with $\mathcal{O}_C \models B \sqsubseteq A$;
- $I_A(x) \leftarrow r(y, x)$, for all r with $\mathcal{O} \models \mathsf{ran}(r) \sqsubseteq A$;
- $I_A(x) \leftarrow r(x, y)$, for all r with $\mathcal{O} \models \mathsf{dom}(r) \sqsubseteq A$;
- $I_r(x, y) \leftarrow s(x, y)$, for all s with $\mathcal{O}_R \models s \sqsubseteq r$.

Now let $q_0'(\vec{x}, \vec{y})$ result from $q_0(\vec{x}, \vec{y})$ by replacing every $A(t)$ in q_0 by $I_A(t)$ and every $r(t, t')$ in q_0 by $I_r(t, t')$. Define Π by adding to $\Pi_{\mathcal{O},\mathsf{basic}}$ the rule $\mathsf{goal}(\vec{x}) \leftarrow q_0'(\vec{x}, \vec{y})$. We show that Π is a rewriting of $(\mathcal{O}, q(\vec{x}))$.

To prove this, we require some preparation. Let $\mathcal{O}$ be a basic schema.org-ontology and Π be a datalog program containing all $\Pi_{\mathcal{O},\mathsf{basic}}$. Let $\mathcal{A}$ be a data instance and consider $\mathcal{I}_{\Pi,\mathcal{A}}$. Then we consider the following variant $\mathcal{J}_{\Pi,\mathcal{A}}$ of $\mathcal{I}_{\Pi,\mathcal{A}}$ in which we transfer the extensions of I_A and $\mathcal{I}_r$ to the concept names A and role names r, respectively:

- $\Delta^{\mathcal{J}_{\Pi,\mathcal{A}}} = \Delta^{\mathcal{I}_{\Pi,\mathcal{A}}}$;

- $r^{\mathcal{J}_{\Pi,\mathcal{A}}} = I_r^{\mathcal{I}_{\Pi,\mathcal{A}}}$ for all role names r;
- $A^{\mathcal{J}_{\Pi,\mathcal{A}}} = I_A^{\mathcal{I}_{\Pi,\mathcal{A}}}$ for all concept names A.

Observation 1 Let $\mathcal{O}$ be a basic schema.org-ontology and Π be a program containing $\Pi_{\mathcal{O},\mathsf{basic}}$. Let $\mathcal{A}$ be a data instance. For a set $X \subseteq \mathsf{Ind}(\mathcal{A})$ let A_a, $a \in X$, be concept names such that $a \notin A^{\mathcal{J}_{\Pi,\mathcal{A}}}$ for all $a \in X$. Then there exists a model $\mathcal{J}$ of $\mathcal{O}$ and $\mathcal{A}$ with

- $\Delta^{\mathcal{J}} = \Delta^{\mathcal{J}_{\Pi,\mathcal{A}}}$;
- $r^{\mathcal{J}} = r^{\mathcal{J}_{\Pi,\mathcal{A}}}$ for all role names r;
- $A^{\mathcal{J}} \supseteq A^{\mathcal{J}_{\Pi,\mathcal{A}}}$ for all concept names A;

such that $a \notin A_a^{\mathcal{J}}$ for all $a \in X$.

Using Observation 1, we now show that Π is a rewriting of $(\mathcal{O}, q(\vec{x}))$.

Clearly, if $\vec{a} \in \Pi(\mathcal{A})$, then $\mathcal{O}, \mathcal{A} \models q(\vec{a})$. Conversely, assume that $\vec{a} \notin \Pi(\mathcal{A})$. Let $\vec{a} = a_1, \dots, a_k$. Consider the minimal model $\mathcal{I}_{\Pi,\mathcal{A}}$ of $\mathcal{A}$. We have $\mathcal{I}_{\Pi,\mathcal{A}} \not\models \mathsf{goal}(\vec{a})$. Let q_0^- be the result of removing from q_0 all atoms $A(t)$ with A a concept name. Let H be the set of all matches π of q_0^- in $\mathcal{J}_{\Pi,\mathcal{A}}$ with $\pi(x_i) = a_i$ for $1 \leq i \leq k$.

If H is empty then expand $\mathcal{J}_{\Pi,\mathcal{A}}$ to a model $\mathcal{J}$ of $\mathcal{O}$ by leaving the interpretation of role names fixed and setting $A^{\mathcal{J}} = \mathsf{Ind}(\mathcal{A})$ for all concept names A. Then $\mathcal{J} \not\models \exists \vec{y}\; q_0^-(\vec{a}, \vec{b})$ and so $\mathcal{J} \not\models q(\vec{a})$. Thus, $\mathcal{O}, \mathcal{A} \not\models q(\vec{a})$, as required.

If H is not empty, let $H(t) = \{\pi(t) \mid \pi \in H\}$ for any term t in q. Note that $H(t)$ is a singleton $\{t^{\mathcal{I}}\}$ if $t = a_i$ or $t = b_i$.

Claim 1. There exists a term t in q such that for all $a \in H(t)$ there exists a concept name A with $A(t) \in q_0$ and $a \notin A^{\mathcal{J}_{\Pi,\mathcal{A}}}$.

Proof of Claim 1. Assume no such t exists. Then choose for every term t in q a member a_t of $H(t)$ such that $a_t \in A^{\mathcal{J}_{\Pi,\mathcal{A}}}$ for all $A(t)$ in q_0. Define a mapping π_0 from the set of terms of t to $\mathcal{J}_{\Pi,\mathcal{A}}$ by setting $\pi_0(t) = a_t$. Using the condition that q has qvar-size at most 1, it is readily checked that π_0 is a match for q in $\mathcal{J}_{\Pi,\mathcal{A}}$ with $\pi(x_i) = a_i$ for $1 \leq i \leq k$. It follows that $\mathcal{I}_{\Pi,\mathcal{A}} \models q_0'(\vec{a})$ and so $\mathcal{I}_{\Pi,\mathcal{A}} \models \mathsf{goal}(\vec{a})$. We have derived a contradiction.

Consider now a term t from q_0 such that for every $a \in H(t)$ there exists a concept name A_a with $A_a(t) \in q_0$ and $a \notin A_a^{\mathcal{J}_{\Pi,\mathcal{A}}}$. By Observation 1 there exists a model $\mathcal{J}$ of $\mathcal{O}$ and $\mathcal{A}$ with

- $\Delta^{\mathcal{J}} = \Delta^{\mathcal{J}_{\Pi,\mathcal{A}}}$;
- $r^{\mathcal{J}} = r^{\mathcal{J}_{\Pi,\mathcal{A}}}$ for all role names r;
- $A^{\mathcal{J}} \supseteq A^{\mathcal{J}_{\Pi,\mathcal{A}}}$ for all concept names A;

such that $a \notin A_a^{\mathcal{J}}$ for all $a \in X$. It follows that $\mathcal{J} \not\models q(\vec{a})$ and so $\mathcal{O}, \mathcal{A} \not\models q(\vec{a})$, as required. ❑

Theorem 8 For every ontology-mediated query $(\mathcal{O}, q(\vec{x}))$ with $\mathcal{O}$ a basic schema.org-ontology and q a CQ of qvar-size at most 2 one can construct in polynomial time a datalog-rewriting of $(\mathcal{O}, q(\vec{x}))$.

Proof. Assume $\mathcal{O}$ and $q(\vec{x}) = \exists \vec{y} q_0(\vec{x}, \vec{y}, \vec{b})$. We employ the program $\Pi_{\mathcal{O},\mathsf{basic}}$ from the proof of Proposition 20. Also, for two sets X_1 and X_2 of concept names we use a program $\Pi_{X_1 \sqcup X_2}$ with intensional predicate $I_{X_1 \sqcup X_2}(x)$ such that for any data instance $\mathcal{A}$ and $a \in \mathsf{Ind}(\mathcal{A})$,

$$\mathcal{O}, \mathcal{A} \models \bigwedge_{A \in X_1} A(a) \vee \bigwedge_{A \in X_2} A(a) \Leftrightarrow \Pi_{X_1 \sqcup X_2} \models I_{X_1 \sqcup X_2}(a).$$

Denote by mCC the set of maximal connected components $\{v, w\}$ of quantified variables $v \neq w$ in q. We assume a fixed ordering v, w of any member of mCC. For any quantified variable v in q we set $X_v = \{A \mid A(v) \in q_0\}$. Let $\vec{z}_{v,w}$ be the variable in $\vec{x} \cup \vec{y}$ without v, w and take, for each $\{v, w\} \in \mathsf{mCC}$, the IDBs $P_{v,w}(v, w, \vec{z}_{v,w}, \vec{b})$, $X_{v,w}(v, \vec{z}_{v,w}, \vec{b})$, and $Y_{v,w}(w, \vec{z}_{v,w}, \vec{b})$. Insert in the rewriting Π the program $\Pi_{\mathcal{O},\mathsf{basic}}$ as well as all rules

$$P_{v,w}(v, w, \vec{z}_{v,w}, \vec{b}) \leftarrow q_0' \wedge I_{X_v \sqcup X_w}(v) \wedge I_{X_v \sqcup X_w}(w)$$

where q_0' results from q_0 by removing all unary atoms $A(t)$ from q_0, replacing all $r(t, t')$ by $I_r(t, t')$, and where $\{v, w\} \in \mathsf{mCC}$. Intuitively, $P_{v,w}(v, w, \vec{z}_{v,w}, \vec{b})$ collects the potential matches for v, w. If, in addition, $X_v(v, \vec{z}_{v,w}, \vec{b})$ and $Y_{v,w}(w, \vec{z}_{v,w}, \vec{b})$ hold, then one has actually found a match for v, w. We model the propagation of the 'colors' $X_{v,w}(v, \vec{z}_{v,w}, \vec{b})$ and $Y_{v,w}(w, \vec{z}_{v,w}, \vec{b})$ by inserting in Π for any $\{v, w\} \in \mathsf{mCC}$

$$\begin{aligned}
X_{v,w}(v, \vec{z}_{v,w}, \vec{b}) &\leftarrow P_{v,w}(v, w, \vec{z}_{v,w}, \vec{b}) \wedge \bigwedge_{A \in X_v} I_{\mathcal{O},A}(v) \\
Y_{v,w}(w, \vec{z}_{v,w}, \vec{b}) &\leftarrow P_{v,w}(v, w, \vec{z}_{v,w}, \vec{b}) \wedge \bigwedge_{A \in X_w} I_{\mathcal{O},A}(w) \\
X_{v,w}(w, \vec{z}_{v,w}, \vec{b}) &\leftarrow P_{v,w}(v, w, \vec{z}_{v,w}, \vec{b}) \wedge X_{v,w}(v, \vec{z}_{v,w}, \vec{b}) \\
Y_{v,w}(v, \vec{z}_{v,w}, \vec{b}) &\leftarrow P_{v,w}(v, w, \vec{z}_{v,w}, \vec{b}) \wedge Y_{v,w}(w, \vec{z}_{v,w}, \vec{b})
\end{aligned}$$

The first of the two recursive rules says that if the only option to possibly avoid a match for v, w is to color $(v, \vec{z}_{v,w}, \vec{b})$ with $X_{v,w}$, then the only way to possibly avoid a match for v, w is to color $(w, \vec{z}_{v,w}, \vec{b})$ with $X_{v,w}$ (because otherwise one would have to color $(w, \vec{x}_{v,w}, \vec{b})$ with $Y_{v,w}$). The second recursive rule can be understood analogously with X and Y swapped. Finally we insert in Π the following goal-rule $\mathsf{goal}(\vec{x}) \leftarrow q_0''$, where q_0'' is obtained from q_0 by replacing all $r(t, t')$ by $I_r(t, t')$, all $A(t)$ that do not participate in any $\{v, w\} \in \mathsf{mCC}$ by $I_A(t)$, and for all $\{v, w\} \in \mathsf{mCC}$ all $A(v)$ by $X_{v,w}(v, \vec{z}_{v,w}, \vec{b})$ and all $A(w)$ by $Y_{v,w}(w, \vec{z}_{v,w}, \vec{b})$, respectively.

The program Π is as required. Assume $q(\vec{x}) = \exists \vec{y}\; q_0(\vec{x}, \vec{y}, \vec{b})$. Let $\vec{x} = x_1 \dots x_k$, $\vec{y} = y_1, \dots, y_m$, and $\vec{b} = b_1, \dots, b_n$. It is straightforward to show that if $\vec{a} \in \Pi(\mathcal{A})$, then $\mathcal{O}, \mathcal{A} \models q(\vec{a})$.

Conversely, assume that $\vec{a} \notin \Pi(\mathcal{A})$. Then $\mathcal{I}_{\Pi,\mathcal{A}} \not\models \exists \vec{y} q_0''(\vec{a}, \vec{y}, \vec{b})$. Let H be the set of all matches π of q_0' in $\mathcal{J}_{\Pi,\mathcal{A}}$ with $\pi(x_i) = a_i$ for $1 \leq i \leq k$ and such that for any $\{v, w\} \in \mathsf{mCC}$ we have $(\pi(v), \pi(w), \pi(\vec{x}), \vec{b}) \in P_{v,w}^{\mathcal{I}_{\Pi,\mathcal{A}}}$. If H is empty then one can show similarly to the proof of Proposition 20 that $\mathcal{O}, \mathcal{A} \not\models q(\vec{a})$, as required.

If H is not empty, let for any connected component $\{v, w\} \in \mathsf{mCC}$, $H(v, w) = \{(\pi(v), \pi(w) \mid \pi \in H\}$ and let for any term t that does not participate in any $\{v, w\} \in \mathsf{mCC}$, $H(t) = \{\pi(t) \mid \pi \in H\}$.

Claim 1. (a) there exists a term t in q that does not participate in any $\{v, w\} \in \mathsf{mCC}$ such that for all $a \in H(t)$ there exists a concept name A with $A(t) \in q_0$ and $a \notin I_A^{\mathcal{I}_{\Pi,\mathcal{A}}}$ or (b) there exists $\{v, w\} \in \mathsf{mCC}$ such that for all $(a, b) \in H(v, w)$ either $(a, \vec{a}, \vec{b}) \notin X_{v,w}^{\mathcal{I}_{\Pi,\mathcal{A}}}$ or $(b, \vec{a}, \vec{b}) \notin Y_{v,w}^{\mathcal{I}_{\Pi,\mathcal{A}}}$.

The proof is similar of Claim 1 is similar to the proof of Claim 1 in Proposition 20.

Now, if (a) holds we can proceed similarly to the proof of Proposition 20 and construct a model $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{A}$ such that $\mathcal{I} \not\models q(\vec{a})$. If (b) holds, then we can construct a model $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{A}$ with $\mathcal{I} \not\models q(\vec{a})$ by picking $\{v, w\} \in \mathsf{mCC}$ such that for all $(a, b) \in H(v, w)$ either $(a, \vec{a}, \vec{b}) \notin X_{v,w}^{\mathcal{I}_{\Pi,\mathcal{A}}}$ or $(b, \vec{a}, \vec{b}) \notin Y_{v,w}^{\mathcal{I}_{\Pi,\mathcal{A}}}$ and ensuring that for every $(a, b) \in H(v, w)$ we either have $a \notin A^{\mathcal{I}}$ for some $A \in X_v$ or $b \notin A^{\mathcal{I}}$ for some $A \in X_w$. ❑

B.1 Proofs for Rewritings of UCQs

Proposition 21. *For basic schema.org-ontologies $\mathcal{O}$ and quantifier-free UCQs $q(x)$ with one variable it is coNP-hard to decide $\mathcal{O}, \mathcal{A} \models q(a)$ even for instance data $\mathcal{A}$ with only one individual a.*

Proof. The proof is by reduction of satisfiability of propositional formulas in CNF. Let $\varphi = \bigwedge_{i \leq n} c_i$ be a conjunction of propositional clauses in variables $v_1, \ldots, v_m$. We represent the formula φ in a basic schema.org-ontology $\mathcal{O}$ as follows:

- the concept names A_j and $\overline{A}_j$, $j \leq m$, are used to encode a positive or a negative literal, respectively, on variable v_j;
- to represent the clauses $c_1, \ldots, c_n$ we use role names $r_1, \ldots, r_n$ and define a range restriction for each role with the concept names corresponding to the literals of the clause. For example, if $c_1 = v_1 \vee \neg v_2$ then we define $\mathsf{ran}(r_1) \sqsubseteq A_1 \sqcup \overline{A}_2$.

Then, for a data instance $\mathcal{A} = \{r_1(a, a), \ldots, r_n(a, a)\}$ and $q(x) = (A_1(x) \wedge \overline{A}_1(x)) \vee \cdots \vee (A_m(x) \wedge \overline{A}_m(x))$ we have that $\mathcal{O}, \mathcal{A} \models q(a)$ iff φ is unsatisfiable. We constructed $\mathcal{O}$ and q over a single variable x such that, on data instances with a single individual a, deciding $\mathcal{O}, \mathcal{A} \models q(a)$ is coNP-hard. ❑

Evaluating a datalog rewriting over a data instance with a single individual is tractable. Thus, if there is a datalog rewriting of $(\mathcal{O}, q(x))$, then it cannot be constructed in polynomial time.

B.2 Proof of Theorem 9

We show the following result.

Theorem 9 Let $\mathcal{O}$ be a basic minimized schema.org-ontology. Then there exists a Boolean CQ q in the language of $\mathcal{O}$ such that $(\mathcal{O}, q)$ is coNP-hard in data complexity iff there exists $F \sqsubseteq A_1 \sqcup \cdots \sqcup A_n \in \mathcal{O}$ with $n \geq 2$. Otherwise every $(\mathcal{O}, q)$ with q a UCQ is FO-rewritable in polynomial time.

Observe that if there exists no $F \sqsubseteq A_1 \sqcup \cdots \sqcup A_n \in \mathcal{O}$ with $n \geq 2$, when we can rewrite every $(\mathcal{O}, q)$ with q a UCQ using the rewriting given in Proposition 20.

In the converse direction we modify a hardness proof given in [Lutz and Wolter, 2012]. The modification is required since we want to show that q can be chosen in such a way that only concept and role names in $\mathcal{O}$ are used. This is not the case in the proof given in [Lutz and Wolter, 2012].

Assume $\mathcal{O}$ is minimized and there exists $F \sqsubseteq A_0 \sqcup \cdots \sqcup A_k \in \mathcal{O}$ with $k \geq 1$. Then $F \in \{\mathsf{dom}(r), \mathsf{ran}(r)\}$ and we assume w.l.o.g. that $F = \mathsf{ran}(r)$. We want to construct a Boolean CQ q in the language of $\mathcal{O}$ such that $(\mathcal{O}, q)$ is coNP-hard in data complexity. To this end, we first construct a Boolean UCQ with these properties and then discuss the modifications required to obtain a Boolean CQ.

The construction of the queries is based on a reduction of the complement of $2+2$-SAT, a variant of propositional satisfiability introduced by Schaerf [Schaerf, 1993]. A $2+2$ clause is of the form $c = (u_0 \vee u_1 \vee \neg u_2 \vee \neg u_3)$, where each of u_l, $l \leq 3$, is a propositional letter or a truth constant $0, 1$. A $2+2$ formula is a finite conjunction of $2+2$ clauses. Now, $2+2$-SAT is the problem of deciding whether a given $2+2$ formula is satisfiable. It is shown in [Schaerf, 1993] that $2+2$-SAT is an NP-complete problem.

Assume $\varphi = c^0 \wedge \cdots \wedge c^n$ is a 2+2-formula in propositional letters $v_0, \ldots, v_m$ and let $c^i = u_0^i \vee u_1^i \vee \neg u_2^i \vee \neg u_3^i$ for $i \leq n$. Our first aim is to define a data instance $\mathcal{A}_\varphi$ and a Boolean UCQ q such that φ is unsatisfiable iff $\mathcal{O}, \mathcal{A}_\varphi \models q$. To start we represent the formula φ in the data instance $\mathcal{A}_\varphi$ as follows:

- the individual names $v_0, \ldots, v_m$ represent variables and the individual names $0, 1$ represent truth constants;
- the individual names c_l^i and b_l^i are used to encode the four literals of each $2+2$ clause c^i, where $i \leq n$ and $l \leq 3$;
- for $i \leq n$ and $l \leq 3$ we use the assertions
$$r(c_l^i, b_l^i), r(b_l^i, u_l^i), r(c_l^i, u_l^i)$$
and
$$r(c_0^i, c_1^i), r(c_1^i, c_2^i), r(c_2^i, c_3^i)$$
to associate the literals c_l^i of a clause c^i to the variable/truth constant u_l^i.

We further extend $\mathcal{A}_\varphi$ to enforce a truth value for each variable v_i, $i \leq m$. To this end, add to $\mathcal{A}_\varphi$ the data instances $\mathcal{A}_i = \{r(f_i, a_i)\}$ for $i \leq m$. Intuitively, $\mathcal{A}_i$ is used to generate a truth value for the variable v_i, where we interpret v_i as true if the query $A_0(a_i)$ is satisfied and as false if any of the queries $A_j(a_i)$, $1 \leq j \leq k$, is satisfied. Finally we extend $\mathcal{A}_\varphi$ by

- linking variables v_i to a_i by adding assertions $r(v_i, a_i)$ for all $i \leq m$;
- to ensure that 0 and 1 have the expected truth values, add the individuals $0'$ and $1'$ with the assertions $A_0(0')$ and $A_1(1')$. Also, add to $\mathcal{A}_\varphi$ the assertions $r(1, 1')$ and $r(0, 0')$.

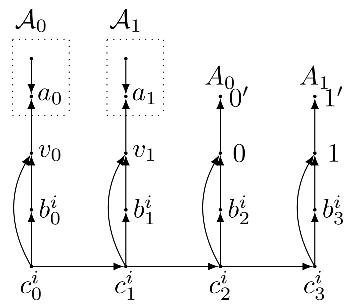

Figure 3: Encoding of a clause $c^i = v_0 \vee v_1 \vee \neg 0 \vee \neg 1$.

Figure 3 illustrates the encoding of a clause $c^i = v_0 \vee v_1 \vee \neg 0 \vee \neg 1$. Consider the Boolean UCQ (we omit existential quantifiers and do not distribute conjunctions over disjunctions)

$$q_0 = \bigwedge_{0 \leq i \leq 2} r(x_i, x_{i+1}) \wedge \bigwedge_{0 \leq i \leq 3} \psi_i$$

where

- $\psi_i = r(x_i, z_i) \wedge r(z_i, y_i) \wedge r(x_i, y_i) \wedge \mathsf{ff}_i(y_i)$ for $i = 0, 1$ and
- $\psi_i = r(x_i, z_i) \wedge r(z_i, y_i) \wedge r(x_i, y_i) \wedge \mathsf{tt}_i(y_i)$ for $i = 2, 3$ and

where

$$\begin{aligned} \mathsf{tt}_i(y_i) &= r(y_i, w_i) \wedge A_0(w_i) \\ \mathsf{ff}_i(y_i) &= r(y_i, w_i) \wedge (\bigvee_{1 \leq j \leq k} A_j(w_i)) \end{aligned}$$

Then $\mathcal{O}, \mathcal{A}_\varphi \models q_0$ iff q_0 is not satisfiable, as required. Note that the 'triangles' using b_l^i in $\mathcal{A}_\varphi$ and using z_i in q_0 are used to ensure that in a match for q_0 in $\mathcal{A}_\varphi$ the only possible matches for (w_0, w_1, w_2, w_3) are $(a_0^i, a_1^i, a_2^i, a_3^i)$ with $r(u_j^i, a_j^i) \in \mathcal{A}_\varphi$, $j \leq 3$, $i \leq n$. In the equivalence proof this is required since we can have inclusions such as $\mathsf{dom}(r) \sqsubseteq A_i \in \mathcal{O}$ for $i \leq n$ in $\mathcal{O}$ which force all A_i to be true in every individual in the domain of A_i.

We now show how to improve the result from Boolean UCQs to CQs. To this end we change the encoding of 'false' from $\mathsf{ff}_i(y_i)$ to

$$\mathsf{ff}'_i(y_i) = \bigwedge_{1 \leq l \leq k} r(y_i, y'_i) \wedge r(y'_i, w_l^i) \wedge A_l(w_l^i)$$

To ensure the match condition discussed above we also modify $\mathsf{tt}_i(y_i)$ to

$$\mathsf{tt}'_i(y_i) = r(y_i, y'_i) \wedge r(y'_i, w_i) \wedge A_0(w_i)$$

We modify $\mathcal{A}_\varphi$ correspondingly:

- for $i \leq m$, remove from $\mathcal{A}_\varphi$ the assertions $r(v_i, a_i)$;
- remove from $\mathcal{A}_\varphi$ the assertions $A_0(0')$ and $A_1(1')$;
- add to $\mathcal{A}_\varphi$ the assertions $r(0', 0'')$, $r(1', 1'')$, $A_0(0'')$ and $A_1(1'')$;
- for $1 \leq j \leq k$, add to $\mathcal{A}_\varphi$ the assertion $A_j(e_j)$ for fresh individual names e_j;

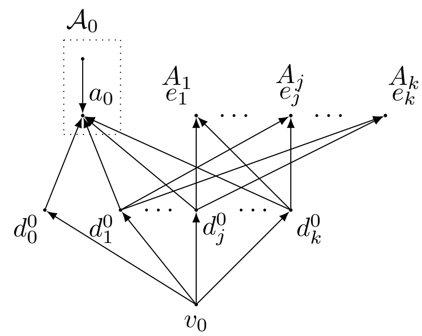

Figure 4: Modified encoding of v_0 occurring in a clause $c^i = v_0 \vee v_1 \vee \neg 0 \vee \neg 1$.

- for $i \leq m$ and $j \leq k$, add to $\mathcal{A}_\varphi$ the assertions $r(v_i, d_j^i)$ for fresh individual names d_j^i;
- for $i \leq m$ and $1 \leq j \leq k$, add to $\mathcal{A}_\varphi$ the assertions $r(d_0^i, a_i)$, $r(d_j^i, a_i)$ and

$$r(d_j^i, e_1), \ldots, r(d_j^i, e_{j-1}), r(d_j^i, e_{j+1}), \ldots, r(d_j^i, e_k).$$

This finishes the modified construction. Figure 4 illustrates the modified encoding of v_0 in a clause $c^i = v_0 \vee v_1 \vee \neg 0 \vee \neg 1$.

B.3 Proof of Theorem 10

When studying the complexity of CSP($\mathcal{B}$) one can assume w.l.o.g. that $\mathfrak{B}$ is a core, that is, every automorphism is an isomorphism. It is useful to further assume that the template $\mathcal{B}$ *admits precoloring*, that is, for each $b \in \Delta^{\mathcal{B}}$ there is a concept name P_b such that $d \in P_b^{\mathcal{B}}$ iff $d = b$ [Cohen and Jeavons, 2006]. It is known that for every template $\mathcal{B}$, there is a template $\mathcal{B}'$ that admits precoloring such that CSP($\mathcal{B}$) and CSP($\mathcal{B}$) are mutually FO-reducible [Larose and Tesson, 2009].

Theorem 10 For every template $\mathcal{B}$ there exists a OMQ $(\mathcal{O}, q)$ where $\mathcal{O}$ is a basic schema.org-ontology $\mathcal{O}$ and q a Boolean acyclic UCQ such that the complement of CSP($\mathcal{B}$) and $(\mathcal{O}, q)$ are mutually FO-reducible.

Proof. Assume a template $\mathcal{B}$ over signature Σ of concept and role names is given such that for each $b \in \Delta^{\mathcal{B}}$ there is a concept name P_b such that $d \in P_b^{\mathcal{B}}$ iff $d = b$. Take a fresh role name s and the concept names P_b, $b \in \Delta^{\mathcal{B}}$, and set

$$\mathcal{O} = \{\mathsf{ran}(s) \sqsubseteq \bigsqcup_{b \in \Delta^{\mathcal{B}}} P_b\}$$

Define a UCQ q as the disjunction of (we omit existential quantifiers)

- $P_a(x) \wedge r(x, y) \wedge P_b(y)$ for all $r \in \Sigma$ such that $(a, b) \notin r^{\mathcal{B}}$;
- $P_a(x) \wedge B(x)$ for all $B \in \Sigma$ such that $a \notin B^{\mathcal{B}}$;
- $P_a(x) \wedge P_b(x)$ for all $a \neq b$.

We show that $(\mathcal{O}, q)$ and coCSP($\mathcal{B}$) (the complement of CSP($\mathcal{B}$)) are mutually FO-reducible.

($\Rightarrow$) Assume a data instance $\mathcal{A}$ containing assertions using symbols in Σ only is given. Let $\mathcal{A}'$ be the union of $\mathcal{A}$ and all $s(a,b)$ such that $a,b \in \mathsf{Ind}(\mathcal{A})$. We show:

Claim 1. $\mathcal{A} \rightarrow \mathcal{B}$ iff $\mathcal{O},\mathcal{A}' \not\models q$.

Assume h is a homomorphism from $\mathcal{A}$ to $\mathcal{B}$. Define a model $\mathcal{I}$ by setting

- $\Delta^{\mathcal{I}} = \mathsf{Ind}(\mathcal{A})$;
- $a \in P_b^{\mathcal{I}}$ iff $h(a) = b$;
- $(a,b) \in r^{\mathcal{I}}$ iff $r(a,b) \in \mathcal{A}$, for all $r \in \Sigma$;
- $a \in B^{\mathcal{I}}$ iff $B(a) \in \mathcal{A}$, for all $B \in \Sigma \setminus \{P_b \mid b \in \Delta^{\mathcal{B}}\}$;
- $(a,b) \in s^{\mathcal{I}}$ for all $a,b \in \mathsf{Ind}(\mathcal{A})$.

It is readily checked that $\mathcal{I}$ is a model of $\mathcal{O}$ and $\mathcal{A}'$ such that $\mathcal{I} \not\models q$. Thus, $\mathcal{O},\mathcal{A}' \not\models q$, as required.

Now assume that $\mathcal{O},\mathcal{A}' \not\models q$. Let $\mathcal{I}$ be a model of $\mathcal{O}$ and $\mathcal{A}'$ such that $\mathcal{I} \not\models q$. Define $h(a) = b$ if $a \in P_b^{\mathcal{I}}$. Using the condition that $\mathcal{I} \not\models q$ one can show that h is well defined and a Σ-homomorphism from $\mathcal{A}$ to $\mathcal{B}$.

($\Leftarrow$) Assume a data instance $\mathcal{A}$ for $\mathcal{O}$ is given. Remove from $\mathcal{A}$ all assertions involving individuals a such that neither $P_b(a) \in \mathcal{A}$ for any $b \in \Delta^{\mathcal{B}}$ nor $s(a',a) \in \mathcal{A}$ for any a' and add the assertions $s(a,a)$ for the remaining individuals a. Clearly we have $\mathcal{O},\mathcal{A} \models q$ iff $\mathcal{O},\mathcal{A}' \models q$ for the resulting data instance $\mathcal{A}'$. Let $\mathcal{A}''$ be the restriction of $\mathcal{A}'$ to Σ. One can show that $\mathcal{O},\mathcal{A}' \models q$ iff $\mathcal{A}'' \not\rightarrow \mathcal{B}$, as required. ❏

B.4 Proof of Theorem 11

We show the PSPACE lower bound for FO-rewritability in basic schema.org-ontologies using a 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 [Cosmadakis *et al.*, 1988], of certain FO-rewritability problems in ontology-based data access [Bienvenu *et al.*, 2013], and of FO-rewritability problems in consistent query answering [Lutz and Wolter, 2015]. Let $M = (Q,\Omega,\Gamma,\delta,q_0,q_{\mathsf{acc}},q_{\mathsf{rej}})$ be a DTM that solves a PSPACE-complete problem and $p(\cdot)$ its polynomial space bound. Here, Q is the set of states, Ω is the input alphabet, Γ the tape alphabet, $\delta : (Q \times \Gamma) \rightarrow \{L,N,R\} \times Q \times \Gamma$ the transition function, $q_0 \in Q$ the initial state, and q_{acc} q_{rej} the accepting and rejecting state, respectively. We assume that the transition function is total except on q_{acc} and q_{rej} where it is undefined for every tape symbol. The tape is assumed to be two-side infinite. We make the following additional assumptions on M. We assume that M never writes the blank symbol and with the *left (resp. right) end of the tape* we mean the first tape cell to the left (resp. right) of the head labeled with a blank. We also assume that M always terminates with the head on the right-most tape cell and that it never attempts to move left on the left-most end of the tape. Finally and most importantly, we assume that, when started in any (not necessarily initial) configuration C, then the computation of M terminates (this assumption is justified in [Lutz and Wolter, 2015]).

Now let M be a TM that satisfies the conditions above and let $x \in \Omega^*$ be an input to M of length n. Our aim is to construct a basic schema.org-ontology $\mathcal{O}$ and Boolean UCQ q such that $(\mathcal{O},q)$ is *not* FO-rewritable iff M accepts x.

A fundamental idea of the reduction is that when M accepts x, then $(\mathcal{O},q)$ is not FO-rewritable because any FO-rewriting would have to query for longer and longer paths that represent the accepting computation of M on x, repeated over and over again; this clearly contradicts the *locality* of an FO-query. In the reduction, we use a very simple ontology

$$\mathcal{O} = \{\mathsf{ran}(s_0) \sqsubseteq B \sqcup B'\}$$

where B, B' are concept names. To understand the source for non-FO-rewritability that we build on, consider the OMQ $(\mathcal{O},q)$ with $\widehat{q} = B(x) \wedge r(x,y) \wedge B'(y)$ (see also Example 7). Non-FO-rewritability is witnessed by path-shaped data instances of the form

$$\begin{aligned}\mathcal{A}_m \quad := \quad & \{r(b_1,b_0), r(b_2,b_1), \dots, r(b_m,b_{m-1})\} \cup \\ & \{B'(b_0), B(b_m)\} \cup \\ & \{s_0(a_i,b_i) \mid 0 < i < m\}.\end{aligned}$$

In fact, it can be verified that $\mathcal{T},\mathcal{A}_m \models \widehat{q}$ for all $m > 0$, but whenever we drop an assertion from $\mathcal{A}_m$ resulting in data instance $\mathcal{A}'_m$, then $\mathcal{T},\mathcal{A}'_m \not\models \widehat{q}$. We are going to modify the above paths so that they describe a (repeated) accepting computation of M on x. To this end, the tape contents, the current state, and the head position are represented using the elements of $\Gamma \cup (\Gamma \times Q)$ as monadic relation symbols. Each constant on the path represents one tape cell of one configuration, the binary relation R is used to move between consecutive tape cells, the binary relation S is used to move between successor configurations inside the same computation, and the binary relation T is used to separate computations. To illustrate, suppose the computation of M on $x = ab$ consists of the two configurations qab and $aq'b$.[1] The corresponding path of length m that describes this computation (repeatedly) is

$$\begin{aligned}& B'(b_0), r(b_1,b_0), s(b_2,b_1), r(b_3,b_2), t(b_4,b_3), \\ & \quad r(b_5,b_4), \dots, r(b_m,b_{m-1}), B(b_m)\end{aligned}$$

with the additional assertions $(a,q)(c)$ for $c = b_0, b_4, b_8, \dots,$ $b(c)$ for $c = b_1, b_5, b_9, \dots,$ $a(c)$ for $c = b_2, b_6, b_{10}, \dots,$ and $(b,q')(c)$ for $c = b_3, b_7, b_{11}, \dots$. We now assemble the UCQ q. To ensure that every individual on the path is labeled with at least one symbol from $\Gamma \cup (\Gamma \times Q)$ (and since we now have three relations r,s,t instead of only a single one), we modify the query $\widehat{q}$ from above. While doing this, we also ensure that t-steps can only occur exactly after the accepting state was reached (we omit existential quantifiers):

(r-pr) $B(x) \wedge A(x) \wedge r(x,y) \wedge A'(y) \wedge B'(y)$, for all $A \in \Gamma \cup (\Gamma \times Q)$ and all $A' \in \Gamma \cup (\Gamma \times (Q \setminus \{q_{\mathsf{acc}}, q_{\mathsf{rej}}\}))$;

(s-pr) $B(x) \wedge A(x) \wedge s(s,y) \wedge A'(y) \wedge B'(y)$, for all $A \in \Gamma \cup (\Gamma \times Q)$ and all $A' \in \Gamma \cup (\Gamma \times (Q \setminus \{q_{\mathsf{acc}}, q_{\mathsf{rej}}\}))$;

(t-pr) $B(x) \wedge A(x) \wedge t(x,y) \wedge A'(y) \wedge B'(y)$ for all $A \in \Gamma \cup (\Gamma \times Q)$ and all $A' \in \Gamma \times \{q_{\mathsf{acc}}\}$.

[1] $uqv \in \Gamma^* Q \Gamma^*$ means that M is in state q, the tape left of the head is labeled with u, and starting from the head position, the remaining tape is labeled with v.

If we simply use the disjunction of the above three queries as the UCQ in our query evaluation problem, then that problem is not FO-rewritable. This is witnessed by paths as above in which every element is labeled with some role symbol from $\Gamma \cup (\Gamma \times Q)$. However, these labeled witness paths need not represent proper computations of M on x since the transition relation need not be satisfied, there need not be any state, etc. We fix these problems by including additional CQs in the UCQ q that discover 'defects' in the computation. These queries rule out labeled path that do not describe proper computations as witnesses for non-FO-rewritability of the defined query evaluation problem: paths with defects are 'yes'-instances, but can be identified by an FO-query. In fact, the following queries do not mention B and B' and thus are derived from $\mathcal{O} \cup \mathcal{A}$ if and only if they have a match in $\mathcal{A}$. They thus do not require any rewriting. The first set of additional CQs ensures that every tape cell has a unique label.

(uni) $A(x) \wedge A'(x)$ for all distinct $A, A' \in \Gamma \cup (\Gamma \times Q)$.

The next CQ enforces that there is not more than one head position per configuration:

(h1) $\bigwedge_{0 \leq l < i} \{r(x_l, x_{l+1})\} \wedge (a, q)(x_i) \wedge \bigwedge_{0 \leq l < j} r(y_l, y_{l+1}) \wedge (a', q')(y_j)$, for all $i < j < p(n)$, $(a, q), (a', q') \in \Gamma \times Q$, and $x_0 = y_0$.

and that there is at least one head position per configuration:

(h2) $r(x_0, x_1) \wedge \ldots \wedge r(x_{p(n)-2}, x_{p(n)-1}) \wedge a_1(x_0) \wedge \ldots \wedge a_{p(n)-1}(x_{p(n)-1})$, for all sequences $a_0, \ldots, a_{p(n)-1} \in \Gamma$.

We ensure that configurations have at most length $p(n)$ using the CQ

(l1) $r(x_0, x_1) \wedge \ldots \wedge r(x_{p(n)-1}, x_{p(n)})$.

We also ensure that configurations are not shorter than $p(n)$ (with the possible exception of the first configuration, which can be shorter):

(l2) $\rho(x_0, x_1) \wedge r(x_1, x_2) \wedge \ldots \wedge r(x_i, x_{i+1}) \wedge \rho'(x_{i+1}, x_{i+2})$ for all $i < p(n) - 1$ and $\rho, \rho' \in \{s, t\}$.

We now enforce that the transition function is respected and that the content of tape cells which are not under the head does not change. Let forbid denote the set of all tuples (A_1, A_2, A_3, A) with $A_i \in \Gamma \cup (\Gamma \times Q)$ such that whenever three consecutive tape cells in a configuration c are labeled with A_1, A_2, A_3, then in the successor configuration c' of c, the tape cell corresponding to the middle cell *cannot* be labeled with A:

(con) $r(x_0) \wedge r(x_0, x_1) \wedge \ldots \wedge r(x_{i-1}, x_i) \wedge s(x_i, y_0) \wedge t(y_0, y_1) \wedge \ldots \wedge r(y_{p(n)-i-3}, y_{p(n)-i-2}) \wedge A_3(y_{p(n)-i-2}) \wedge r(y_{p(n)-i-2}, y_{p(n)-i-1}) \wedge A_2(y_{p(n)-i-1}) \wedge r(y_{p(n)-i-1}, y_{p(n)-i}) \wedge A_1(y_{p(n)-i})$ for all $0 \leq i < p(n)$ and $(A_1, A_2, A_3, A) \in \mathsf{forbid}$.

It remains to set up the initial configuration. Recall that witness instances consist of repeated computations of M, which ideally we would all like to start in the initial configuration for input x. It does not seem possible to enforce this for the first computation in the instance, so we live with this computation starting in some unknown configuration, relying on our assumption that M terminates also when started in an arbitrary configuration. Then, we utilize the final states q_{acc} and q_{rej} to enforce that all computations in the instance except the first one must start with the initial configuration for x. Let $A_0^{(0)}, \ldots, A_{p(n)-1}^{(0)}$ be the monadic relation symbols that describe the initial configuration, i.e., when the input x is $x_0 \cdots x_{n-1}$, then $A_0^{(0)} = (x_0, q_0)$, $A_i^{(0)} = x_i$ for $1 \leq i < n$, and $A_i^{(0)} = x_i$ is the blank symbol for $n \leq i < p(n)$. Now take

(in) $\bigwedge_{0 \leq l < i} r(x_l, x_{l+1}) \wedge t(x_i, x_{i+1}) \wedge A(x_0)$ for all $0 \leq i < p(n)$ and $A \neq A_i^{(0)}$.

The query q is the UCQ defined by taking the union of all Boolean CQs given above. The following lemma establishes the correctness of our reduction.

Lemma 22. *$(\mathcal{O}, q)$ is* not *FO-rewritable iff M accepts x.*

Proof. "if". Assume that M accepts x. By using standard locality arguments (e.g., Hanf's Theorem), it is enough to show that there exist arbitrary large k and instances $\mathcal{A}_k$ with domain $\{a_0, b_0, \ldots, a_k, b_k\}$ such that

- for all $i, j \leq k$: if $\rho(b_i, b_j) \in I_k$ for some $\rho \in \{r, s, t\}$, then $i = j + 1$ or $j = i + 1$;
- The assertions involving s_0 in $\mathcal{A}_k$ are exactly $s_0(a_i, b_i)$ for $0 < i < m$;
- $\mathcal{O}, \mathcal{A}_k \models q$;
- $\mathcal{O}, \mathcal{A} \not\models q$, where $\mathcal{A}$ is the disjoint union of the data instances $\mathcal{A}_k^0$ and $\mathcal{A}_k^k$, where $\mathcal{A}_k^0$ is obtained from $\mathcal{A}_k$ by removing all facts involving b_0 and $\mathcal{A}_k^k$ is obtained from $\mathcal{A}_k$ by removing all facts involving b_k.

Assume $k > 0$ is given. Let $C_1, \ldots, C_m$ be a sequence of configurations of length $p(n)$ obtained by sufficiently often repeating the accepting computation of M on x so that $|C_1| + \cdots + |C_m| \geq k$. We can convert $C_1, \ldots, C_m$ into the desired witness data instance $\mathcal{A}_k$ in a straightforward way: introduce one individual name for each tape cell in each configuration and computation, use r to connect cells within the same configuration, s to connect configurations, and t to connect computations, and the symbols from $\Gamma \cup (\Gamma \times Q)$ to indicate the tape inscription, current state, and head position. We obtain instance data satisfying the conditions above by identifying the individuals with $b_0, \ldots, b_k$ assuming that b_0 stands for the first cell of the first configuration of C_1. Finally add the assertions $\{B'(b_0)\} \cup \{B(b_k))\} \cup \{s_0(a_i, b_i) \mid 0 < i < k\}$ to obtain $\mathcal{A}_k$. It can be verified that $\mathcal{A}_k$ is as required. To see that $\mathcal{O}, \mathcal{A}_k \models q$ observe that in any model $\mathcal{I}_k'$ of $\mathcal{A}_k$ there is some i with $0 \leq i < k$ such that $B'(b_i) \in \mathcal{I}_k'$ and $B(b_{i+1}) \in \mathcal{I}_k'$. To see that $\mathcal{O}, \mathcal{A} \not\models q$ for the disjoint union $\mathcal{A}$ of $\mathcal{A}_k^0$ and $\mathcal{A}_k^k$, observe that one obtains a model of $\mathcal{A}$ by satisfying B'

everywhere in the interpretation corresponding to $\mathcal{A}_k^0$ and B everywhere in the interpretation corresponding to $\mathcal{A}_k^k$.

"only if". Assume that $(\mathcal{O}, q)$ is not FO-rewritable. Note that all CQs in q that are distinct from (r-pr), (s-pr), and (t-pr) have a match in $\mathcal{A}$ iff they they are entailed by $\mathcal{O}, \mathcal{A}$. Thus, they are FO-rewritable. Now consider the following

Observation. Assume $\mathcal{A}$ is a data instance such that no CQ in q distinct from the CQs (r-pr), (s-pr), and (t-pr) has a match in $\mathcal{A}$. Then $\mathcal{O}, \mathcal{A} \models q$ iff there exists $k > 0$ such that the following condition $(*_k)$ holds: there are

$$\rho_0(b_1, b_0), \ldots, \rho_{k-1}(b_k, b_{k-1}), A_0(b_0), \ldots, A_k(b_k) \in \mathcal{A}$$

with $\rho_i \in \{r, s, t\}$ for all $i < k$ and $A_i \in \Gamma \cup (\Gamma \times Q)$ for all $i \leq k$ such that

- $B'(b_0) \in \mathcal{A}, B(b_0) \notin \mathcal{A}$,
- $B(b_k) \in \mathcal{A}_k, B'(b_k) \notin \mathcal{A}$,
- for all $0 < i < k$ there exists a such that $s_0(a, b_i) \in \mathcal{A}$,
- if $\rho_{i+1} \in \{r, s\}$, then $A_i \in \Gamma \cup (\Gamma \times (Q \setminus \{q_{\mathsf{acc}}, q_{\mathsf{rej}}\}))$,
- if $\rho_{i+1} = t$, then $A_i \in \Gamma \cup (\Gamma \times \{q_{\mathsf{acc}}\})$.

Clearly, for every $k > 0$ condition $(*_k)$ can be expressed in FO. Thus, if $(\mathcal{O}, q)$ is not FO-rewritable, then for every $k > 0$ there exists a data instance $\mathcal{A}$ satisfying $(*_k)$. Now let m_0 be the maximum number of steps M makes starting from any configuration of length $p(n)$ before entering the final state. One can prove that any $\mathcal{A}$ satifying $(*_k)$ for $k \geq 2m_0(p(n) + 1) + 1$ encodes an accepting computation of M for input x, as required. ❑

C Proofs for Section 4

In this section, we provide proofs of Theorems 12–14 from Section 4. Some of the results presented here also form the basis for the next section.

Let $\mathcal{O}$ be a schema.org-ontology. By $\mathcal{O}_{\mathsf{basic}}$ we will always denote the basic schema.org-ontology obtained from $\mathcal{O}$ by dropping all enumeration definitions and viewing all datatype names as concept names.

A *basic concept* B is a concept name, an expression of the form $\mathsf{dom}(r)$ or $\mathsf{ran}(r)$ with r a role name, or a datatype name in DT. Let $\mathsf{BC}(\mathcal{O})$ be the set of all basic concepts constructed from concept names and role names in $\mathcal{O}$, and datatype names in DT.

An *item type over* $\mathcal{O}$ (or *item type* if $\mathcal{O}$ is understood) is

- an enumeration individual in $\mathsf{N_E}(\mathcal{O})$,
- a datatype in DT, or
- the symbol $\star$.

Given an interpretation $\mathcal{I}$, an $a \in \Delta^{\mathcal{I},\mathrm{DT}}$, and an item type t over $\mathcal{O}$, we say that *a has type t* (alternatively, *t is the type of a*, or *a realizes t*) if

- $t \in \mathsf{N_E}(\mathcal{O})$ and $a = t$,
- $t \in \mathsf{DT}$ and $a \in \Delta^t$, or
- $t = \star$ and $a \in \Delta^{\mathcal{I}} \setminus \mathsf{N_E}$.

For all $B \in \mathsf{BC}(\mathcal{O})$, let $\mathsf{IT}_{\mathcal{O}}(B)$ be the set of all item types t over $\mathcal{O}$ that satisfy the following conditions:

- if $B \in \mathsf{N_C}$, then $t \notin \mathsf{DT}$, and $t \in X$ for all $C \equiv X \in \mathcal{O}$ with $\mathcal{O}_{\mathsf{basic}} \models B \sqsubseteq C$;
- if B is the name of a datatype $\mathcal{D} \in \mathsf{DT}$, then $t = \mathcal{D}$;
- if $B = f(r)$ for some $r \in \mathsf{N_R}$ and $f \in \{\mathsf{dom}, \mathsf{ran}\}$, then the following are true:
  - if $f = \mathsf{dom}$, then $t \notin \mathsf{DT}$; and
  - for all $f(s) \sqsubseteq C_1 \sqcup \cdots \sqcup C_k \in \mathcal{O}$ with $\mathcal{O}_{\mathsf{basic}} \models r \sqsubseteq s$ there is $i \in \{1, \ldots, k\}$ with $t \in \mathsf{IT}_{\mathcal{O}}(C_i)$.

It is straightforward to check that $\mathsf{IT}_{\mathcal{O}}(B)$ can be computed in polynomial time from $\mathcal{O}$ and B. For this, recall from Section 3 that the entailment problem for basic schema.org-ontologies is decidable in polynomial time.

We are now ready to prove Theorem 12.

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

Proof. Observe that $\mathcal{O}$ is incoherent iff $\mathsf{IT}_{\mathcal{O}}(B) = \emptyset$ for some $B \in \mathsf{BC}(\mathcal{O})$. Furthermore, $\mathsf{IT}_{\mathcal{O}}(B)$ can be computed in polynomial time for each basic concept B. ❑

Let $\mathcal{A}$ be a data instance for $\mathcal{O}$. For each $a \in \mathsf{Ind}(\mathcal{A})$, let $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$ be the set of all item types t over $\mathcal{O}$ such that

- for all $B \in \mathsf{BC}(\mathcal{O})$ with $\mathcal{I}_{\mathcal{A}} \models B(a)$ we have $t \in \mathsf{IT}_{\mathcal{O}}(B)$; and
- if $a \in \mathsf{N_E}$, then $t = a$.

The following lemma shows that $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$ consists of precisely those item types that are realizable by $a^{\mathcal{I}}$ in some model $\mathcal{I}$ of $\mathcal{O} \cup \mathcal{A}$.

Lemma 23. *Let $\mathcal{O}$ be a schema.org-ontology, and let $\mathcal{A}$ be a data instance for $\mathcal{O}$.*

1. *Let $\mathcal{I}$ be a model of $\mathcal{O} \cup \mathcal{A}$, let $a \in \mathsf{Ind}(\mathcal{A})$, and let t be the item type of $a^{\mathcal{I}}$. Then, $t \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$.*
2. *Let $t_a \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$ for each $a \in \mathsf{Ind}(\mathcal{A})$. Then, there is a model $\mathcal{I}$ of $\mathcal{O} \cup \mathcal{A}$ such that t_a is the item type of $a^{\mathcal{I}}$ for each $a \in \mathsf{Ind}(\mathcal{A})$. Furthermore, if $t_a = \star$, then $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ for all $b \in \mathsf{Ind}(\mathcal{A})$ with $a \neq b$.*

Proof. *Ad 1:* To prove $t \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$, we have to show that for all $B \in \mathsf{BC}(\mathcal{O})$ with $\mathcal{I}_{\mathcal{A}} \models B(a)$ we have $t \in \mathsf{IT}_{\mathcal{O}}(B)$; and if $a \in \mathsf{N_E}$, then $t = a$.

First of all, since $a^{\mathcal{I}}$ has the item type t, we have that $a \in \mathsf{N_E}$ implies $t = a^{\mathcal{I}} = a$.

Next, let $B \in \mathsf{BC}(\mathcal{O})$ be such that $\mathcal{I}_{\mathcal{A}} \models B(a)$. We have to show that $t \in \mathsf{IT}_{\mathcal{O}}(B)$. To this end, we distinguish the following three cases:

- *Case 1:* $B \in \mathsf{N_C}$. Since $\mathcal{I}_{\mathcal{A}} \models B(a)$ and datatype values are not allowed to occur in concepts, we have $a \notin \bigcup_{\mathcal{D} \in \mathrm{DT}} \Delta^{\mathcal{D}}$ and hence $t \notin \mathrm{DT}$. Now, let $C \equiv X \in \mathcal{O}$ be such that $\mathcal{O}_{\mathsf{basic}} \models B \sqsubseteq C$. Since $\mathcal{I}$ is a model of $\mathcal{O} \cup \mathcal{A}$ and $\mathcal{I}_A \models B(a)$, we have $\mathcal{I} \models C(a)$. Hence, $a^{\mathcal{I}} \in X$, which implies $t = a^{\mathcal{I}} \in X$.
- *Case 2: B is a datatype name in* DT. Let $\mathcal{D} \in \mathsf{DT}$ be such that $\mathcal{D} = (B, \Delta^{\mathcal{D}})$. Since $\mathcal{I}_{\mathcal{A}} \models B(a)$, we have that $a \in \Delta^{\mathcal{D}}$. Hence, $t = \mathcal{D}$.

- *Case 3: $B = f(r)$ for some $r \in \mathsf{N_R}$, $f \in \{\mathsf{dom}, \mathsf{ran}\}$.* First of all, if $f = \mathsf{dom}$, then by $\mathcal{I}_A \models B(a)$ and the definition of interpretation, we have $a \notin \bigcup_{\mathcal{D} \in \mathsf{DT}} \Delta^{\mathcal{D}}$ and hence $t \notin \mathsf{DT}$. Next, let $f(s) \sqsubseteq C_1 \sqcup \cdots \sqcup C_k \in \mathcal{O}$ be such that $\mathcal{O}_{\mathsf{basic}} \models r \sqsubseteq s$. From $\mathcal{I}_{\mathcal{A}} \models B(a)$ and $\mathcal{O}_{\mathsf{basic}} \models r \sqsubseteq s$, we obtain $a^{\mathcal{I}} \in f(s)^{\mathcal{I}}$. Hence, there is an $i \in \{1, \ldots, k\}$ with $\mathcal{I} \models C_i(a)$. We can now prove $t \in \mathsf{IT}_{\mathcal{O}}(C_i)$ similarly to cases 1 and 2 above.

Ad 2: For all $a \in \mathsf{Ind}(\mathcal{A})$, let θ_a be the smallest set of basic concepts over $\mathcal{O}$ such that for all $B \in \mathsf{BC}(\mathcal{O})$,

- if $\mathcal{I}_{\mathcal{A}} \models B(a)$, then $B \in \theta_a$;
- if $B \in \theta_a$ and $\mathcal{O}_{\mathsf{basic}} \models B \sqsubseteq C_1 \sqcup \cdots \sqcup C_k$, then θ_a contains all C_i with $t_a \in \mathsf{IT}_{\mathcal{O}}(C_i)$.

Since $t \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$, we have $t \in X$ for all $B \equiv X \in \mathcal{O}$ with $B \in \theta_a$, and $t \in \mathsf{DT}$ iff $t = (B, \Delta^t)$ and $B \in \theta_a$. It is straightforward to construct a model $\mathcal{I}$ of $\mathcal{O} \cup \mathcal{A}$ such that for each $a \in \mathsf{Ind}(\mathcal{A})$,

- $a^{\mathcal{I}} = t_a$ if $t_a \in \mathsf{N_E}$, $a^{\mathcal{I}} \in \Delta^{t_a}$ if $t_a \in \mathsf{DT}$, and $a^{\mathcal{I}} = a$ if $t_a = \star$;
- for all $B \in \mathsf{BC}(\mathcal{O})$, we have $\mathcal{I} \models B(a)$ iff $B \in \theta_a$.

In particular, t_a is the item type of $a^{\mathcal{I}}$. Furthermore, if $t_a = \star$, then $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ for all $b \in \mathsf{Ind}(\mathcal{A})$ with $a \neq b$. ❑

It follows that $\mathcal{A}$ is unsatisfiable w.r.t $\mathcal{O}$ iff $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$ is empty for some $a \in \mathsf{Ind}(\mathcal{A})$. Since for each $B \in \mathsf{BC}(\mathcal{O})$, $\mathsf{IT}_{\mathcal{O}}(B)$ is computable in polynomial time from $\mathcal{O}$, we have that $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$ is computable in polynomial time from $\mathcal{O}$ and $\mathcal{A}$ for each $a \in \mathsf{Ind}(\mathcal{A})$. Hence, unsatisfiability of $\mathcal{A}$ w.r.t. $\mathcal{O}$ is decidable in PTime. Even stronger, the following lemma implies that unsatisfiability is definable in non-recursive datalog. For every concept $B \in \mathsf{BC}(\mathcal{O})$, define a relational atom $\mathsf{at}_B(x)$ as follows: if $B = A \in \mathsf{N_C}$, then $\mathsf{at}_B(x) = A(x)$; if $B = \mathsf{dom}(r)$, then $\mathsf{at}_B(x) = r(x, z)$; and if $B = \mathsf{ran}(r)$, then $\mathsf{at}_B(x) = r(z, x)$.

Lemma 24. *For any schema.org-ontology $\mathcal{O}$ and item type t over $\mathcal{O}$, one can compute in polynomial time a non-recursive datalog program Π_t such that for any data instance $\mathcal{A}$ for $\mathcal{O}$, $\Pi_t(\mathcal{A})$ is the set of all $a \in \mathsf{Ind}(\mathcal{A})$ with $t \notin \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$.*

Proof. Let Π_t be the datalog program containing the rules

- $\mathsf{goal}(x) \leftarrow \mathsf{at}_B(x)$ for $B \in \mathsf{BC}(\mathcal{O})$ with $t \notin \mathsf{IT}_{\mathcal{O}}(B)$;
- $\mathsf{goal}(x) \leftarrow x = a$ for all $a \in \mathsf{N_E}(\mathcal{O})$ with $t \neq a$;
- $\mathsf{goal}(x) \leftarrow r(y, x) \wedge D(x)$ for all $\mathcal{D} = (D, \Delta^{\mathcal{D}}) \in \mathsf{DT}$ with $\mathcal{D} \neq t$.

It it easy to verify that if $b \in \mathsf{Ind}(\mathcal{A})$ satisfies the body of a rule in Π_t, then $t \notin \mathsf{IT}_{\mathcal{O},\mathcal{A}}(b)$. Hence, $\Pi_t(\mathcal{A})$ has the desired property. Since $\mathsf{IT}_{\mathcal{O}}(\cdot)$ can be computed in polynomial time, Π_t can be computed in polynomial time. ❑

It is now easy to prove Theorem 13 using Lemma 24.

Theorem 13 (restated). *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 $\Pi(\mathcal{A}) \neq \emptyset$.*

Proof. The program Π can be obtained from the programs Π_t in Lemma 24 as follows. Let $\mathsf{goal}_t(x)$ be the goal predicate of Π_t. Then, Π contains the program Π_t for each item type t, and the following additional rule

$$\mathsf{goal}() \leftarrow \bigwedge_t \mathsf{goal}_t(x),$$

where t ranges over all the items types over $\mathcal{O}$. Then, $\Pi(\mathcal{A})$ is non-empty iff $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(a) = \emptyset$ for some $a \in \mathsf{Ind}(\mathcal{A})$. The latter holds precisely if $\mathcal{A}$ is unsatisfiable w.r.t. $\mathcal{O}$. ❑

We conclude this section by proving Theorem 14. Recall the notion of a *conflict assertion* from Section 4.

Theorem 14 (restated). *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 $\mathcal{A}$.*

Proof. For concept names A, Π contains the following rules:

- for all minimal $S \subseteq \mathsf{BC}(\mathcal{O})$ such that $A \in S$ and $\bigcap_{C \in S} \mathsf{IT}_{\mathcal{O}}(C) = \emptyset$, the rule $\mathsf{goal}(x) \leftarrow \bigwedge_{C \in S} \mathsf{at}_C(x)$;
- $\mathsf{goal}(x) \leftarrow A(x) \wedge x = a$ for all $a \in \mathsf{N_E}(\mathcal{O})$ with $a \notin \mathsf{IT}_{\mathcal{O}}(A)$.

For role names r, Π contains the following rules:

- for all minimal $S \subseteq \mathsf{BC}(\mathcal{O})$ such that $\mathsf{dom}(r) \in S$ and $\bigcap_{C \in S} \mathsf{IT}_{\mathcal{O}}(C) = \emptyset$, the rule $\mathsf{goal}(x, y) \leftarrow r(x, y) \wedge \bigwedge_{C \in S} \mathsf{at}_C(x)$;
- for all minimal $S \subseteq \mathsf{BC}(\mathcal{O})$ such that $\mathsf{ran}(r) \in S$ and $\bigcap_{C \in S} \mathsf{IT}_{\mathcal{O}}(C) = \emptyset$, the rule $\mathsf{goal}(x, y) \leftarrow r(x, y) \wedge \bigwedge_{C \in S} \mathsf{at}_C(y)$;
- $\mathsf{goal}(x, y) \leftarrow r(x, y) \wedge x = a$ for all $a \in \mathsf{N_E}(\mathcal{O})$ with $a \notin \mathsf{IT}_{\mathcal{O}}(\mathsf{dom}(r))$;
- $\mathsf{goal}(x, y) \leftarrow r(x, y) \wedge y = a$ for all $a \in \mathsf{N_E}(\mathcal{O})$ with $a \notin \mathsf{IT}_{\mathcal{O}}(\mathsf{ran}(r))$;
- $\mathsf{goal}(x, y) \leftarrow r(x, y) \wedge D(y)$ if $\mathcal{D} = (D, \Delta^{\mathcal{D}}) \in DT$ and $\mathcal{D} \notin \mathsf{IT}_{\mathcal{O}}(\mathsf{ran}(r))$.

It is easy to verify that Π has the desired properties. ❑

D Proof of Theorem 15

This section provides a detailed proof of Theorem 15. We first show that over schema.org-ontologies, quantifier-free CQs can be polynomially rewritten into datalog-programs. Polynomial time combined complexity of answering quantifier-free CQs q w.r.t. schema.org-ontologies $\mathcal{O}$ will then follow from the structure of the rewriting of $(\mathcal{O}, q)$.

To simplify technical constructions, we assume that $\mathsf{N_E}(\mathcal{O}) \subseteq \Delta^{\mathcal{I}_{\mathcal{A}}}$ and $a_i \in A^{\mathcal{I}_{\mathcal{A}}}$ whenever $A \equiv \{a_1, \ldots, a_n\} \in \mathcal{O}$, $1 \leq i \leq n$. All results in this section also hold without this assumption. For instance, in the datalog programs constructed in Lemmas 26 and 29 below, we can "simulate" this by adding rules $A(a_i) \leftarrow \mathsf{true}$ for all $A \equiv \{a_1, \ldots, a_k\} \in \mathcal{O}$ and $1 \leq i \leq k$; the resulting programs are non-recursive iff the original programs are. For the definitions of $\mathcal{O}_{\mathsf{basic}}$, the set $\mathsf{BC}(\mathcal{O})$ of basic concepts of a schema.org-ontology $\mathcal{O}$, the concept of an item type, and the set $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(\cdot)$, we refer the reader to Section C.

We start by considering queries of the form $A(x)$. Note that rewritings as constructed in Section 3 do not work here, because enumeration types in the ontology might force us to map an individual to an enumeration individual for which we can derive A. For example, let $\mathcal{O} = \{B \equiv \{b_1, b_2\}\}$ and $\mathcal{A} = \{B(a), A(b_1), A(b_2)\}$. Then, $\mathcal{O}, \mathcal{A} \models A(a)$ holds, although $A(a)$ is not true in $\mathcal{A}$ (i.e., $A(a)$ is not true after removing the enumeration definition from $\mathcal{O}$). Nevertheless, there is a simple non-recursive datalog rewriting of $A(x)$:[2]

$$\begin{aligned}
\mathsf{goal}(x) &\leftarrow A(x),\\
\mathsf{goal}(x) &\leftarrow B(x) \wedge \mathsf{goal}_1(x) \wedge \mathsf{goal}_2(x),\\
\mathsf{goal}_i(x) &\leftarrow \mathsf{neq}(x, b_i)\\
\mathsf{goal}_i(x) &\leftarrow A(b_i).
\end{aligned}$$

Here, $\mathsf{goal}_i(a)$ checks that if a can have the item type b_i (i.e., if it is not the case that $a \in \{b_1, b_2\}$ and $a \neq b_i$), then $A(b_i)$ holds. Thus, $\mathsf{goal}(a)$ is true iff the data instance contains $A(a)$, or a cannot have the item type $\star$ and for the remaining item types $t \in \{b_1, b_2\}$ that are possible for a the data instance contains $A(t)$. It is easy to check that the latter is equivalent to $\mathcal{O}, \mathcal{A} \models A(a)$.

We now describe the construction for arbitrary schema.org-ontologies. Before doing this, we show under which conditions one can derive an atom $A(a)$ from a data instance $\mathcal{A}$ and a schema.org-ontology $\mathcal{O}$.

Lemma 25. *Let $\mathcal{A}$ be a data instance for $\mathcal{O}$, $A \in \mathsf{N_C}$, and $a \in \mathsf{N_I}$. Suppose that $\mathcal{O} \cup \mathcal{A}$ is satisfiable. Then, $\mathcal{O}, \mathcal{A} \models A(a)$ iff one of the following applies:*

1. $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \models A(a)$*; or*
2. $\star \notin \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$*, and for all* $t \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$ *there exists a* $b \in \mathsf{Ind}(\mathcal{A})$ *with* $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \models A(b)$ *and* $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(b) \subseteq \{t\}$.

Proof. *"Only if"* We prove the contrapositive. Suppose that $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \not\models A(a)$, and that one of the following applies:

1. $\star \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$; or
2. there exists a $t \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$ such that for all $b \in \mathsf{Ind}(\mathcal{A})$ with $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \models A(b)$ we have $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(b) \not\subseteq \{t\}$.

We show that $\mathcal{O}, \mathcal{A} \not\models A(a)$. To this end, we show that there is a model $\mathcal{I}$ of $\mathcal{O} \cup \mathcal{A}$ with $\mathcal{I} \not\models A(a)$.

First, assume that $\star \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$. Since $\mathcal{O} \cup \mathcal{A}$ is satisfiable, we can pick an element $t_b \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(b)$ for each $b \in \mathsf{Ind}(\mathcal{A})$. By Lemma 23, there is a model $\mathcal{I}$ of $\mathcal{O} \cup \mathcal{A}$ such that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ for all $b \in \mathsf{Ind}(\mathcal{A}) \setminus \{a\}$. Together with $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \not\models A(a)$, this implies $\mathcal{I} \not\models A(a)$.

Next, assume that there exists a $t \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$ such that for all $b \in \mathsf{Ind}(\mathcal{A})$ with $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \models A(b)$ we have $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(b) \not\subseteq \{t\}$. By Lemma 23, there is a model $\mathcal{I}$ of $\mathcal{O} \cup \mathcal{A}$ such that $a^{\mathcal{I}}$ has item type t, and each $b \in \mathsf{Ind}(\mathcal{A})$ with $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \models A(b)$ has an item type distinct from t. In particular, each $b \in \mathsf{Ind}(\mathcal{A})$ with $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \models A(b)$ is assigned to an individual distinct from $a^{\mathcal{I}}$. Together with $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \not\models A(a)$, this implies $\mathcal{I} \not\models A(a)$.

"If" Clearly, $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \models A(a)$ implies $\mathcal{O}, \mathcal{A} \models A(a)$. Assume now that $\star \notin \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$, and that for all $t \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$ there exists a $b \in \mathsf{Ind}(\mathcal{A})$ with $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \models A(b)$ and $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(b) \subseteq \{t\}$. We show that $\mathcal{O}, \mathcal{A} \models A(a)$. To this end, let $\mathcal{I}$ be a model of $\mathcal{O} \cup \mathcal{A}$, and let t be the item type of $a^{\mathcal{I}}$. By Lemma 23, we have $t \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$. Hence, by our assumption, there is a $b \in \mathsf{Ind}(\mathcal{A})$ with $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \models A(b)$ and $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(b) \subseteq \{t\}$. Fix such a b. Since $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(b) \subseteq \{t\}$ and $t \neq \star$, we have $a^{\mathcal{I}} = b^{\mathcal{I}}$. Together with $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \models A(b)$, this implies $\mathcal{I} \models A(a)$. ❑

We are now ready for giving the construction of a non-recursive datalog rewriting of atomic queries $A(x)$ w.r.t. arbitrary schema.org-ontologies.

Lemma 26. *For every schema.org-ontology $\mathcal{O}$ and every $A \in \mathsf{N_C}$, one can construct in polynomial time a non-recursive datalog-rewriting of $(\mathcal{O}, A(x))$.*

Proof. As mentioned at the end of Section 4 it suffices to construct a rewriting that works for data instances $\mathcal{A}$ such that $\mathcal{O} \cup \mathcal{A}$ is satisfiable. Let $\mathcal{A}$ be a data instance for $\mathcal{O}$ and $a \in \mathsf{Ind}(\mathcal{A})$. By Lemma 25 we have $\mathcal{O}, \mathcal{A} \models A(a)$ iff one of the following applies:

1. $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \models A(a)$; or
2. $\star \notin \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$, and for all $t \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$ there exists a $b \in \mathsf{Ind}(\mathcal{A})$ with $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \models A(b)$ and $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(b) \subseteq \{t\}$.

The datalog program constructed below implements the above checks.

By Proposition 20, we can compute in polynomial time a non-recursive datalog rewriting Π_A of $(\mathcal{O}_{\mathsf{basic}}, A(x))$. Let $\mathsf{certain}_A$ be the goal predicate of Π_A. Furthermore, by Lemma 24, for every item type t over $\mathcal{O}$ we can compute in polynomial time a non-recursive datalog program Π_t such that for any data instance $\mathcal{A}$ for $\mathcal{O}$, $\Pi_t(\mathcal{A})$ is the set of all $a \in \mathsf{Ind}(\mathcal{A})$ with $t \notin \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$. Let $\overline{\mathsf{it}}_t$ be the goal predicate of Π_t.

Now, consider the non-recursive datalog program Π containing Π_A and Π_t, for every item type t over $\mathcal{O}$, and the following additional rules:

1. $\mathsf{goal}(x) \leftarrow \mathsf{certain}_A(x)$;
2. $\mathsf{goal}(x) \leftarrow \bigwedge_{t \in \mathsf{N_E}(\mathcal{O}) \cup \{\star\}} R_t(x)$;
3. $R_t(x) \leftarrow \overline{\mathsf{it}}_t(x)$ for all item types t over $\mathcal{O}$;
4. $R_t(x) \leftarrow \mathsf{certain}_A(y) \wedge \bigwedge_{t' \in \mathsf{N_E}(\mathcal{O}) \setminus \{t\}} \overline{\mathsf{it}}_{t'}(y)$ for all item types $t \in \mathsf{N_E}(\mathcal{O})$.[3]

Here, goal and R_t, for each item type t over $\mathcal{O}$, are fresh unary IDB predicates. Clearly, Π can be computed in polynomial time from $\mathcal{O}$. The characterisation of $\mathcal{O}, \mathcal{A} \models A(a)$ at the beginning of the proof implies that for every data instance $\mathcal{A}$ for $\mathcal{O}$ and every $a \in \mathsf{Ind}(\mathcal{A})$, we have $a \in \Pi(\mathcal{A})$ iff $\mathcal{O}, \mathcal{A} \models A(a)$. ❑

[2] To simplify the presentation, we omit the straightforward rules for deriving $\mathsf{neq}(b_1, b_2)$, $\mathsf{neq}(b_2, b_1)$ as well as $B(b_1)$ and $B(b_2)$.

[3] Technically, we would have to add atoms to the body to "cover" the variable x. We can easily do this by first adding rules that define the unary predicate of all individual names in $\mathcal{A}$, and then using this unary predicate to "cover" x.

Next, we deal with atomic role queries. We first prove an auxiliary lemma, Lemma 28, which states under which conditions one can derive an atom $r(a,b)$ from a data instance $\mathcal{A}$ and a schema.org-ontology $\mathcal{O}$. The lemma is based on the following notion of path.

Definition 27. Let $\mathcal{O}$ be a schema.org-ontology, $\mathcal{A}$ a data instance for $\mathcal{O}$, $a,b \in \mathsf{Ind}(\mathcal{A})$, and t_a, t_b item types over $\mathcal{O}$. An (r,a,b,t_a,t_b)*-path in* $\mathcal{O},\mathcal{A}$ is a sequence $c_0, c_1, \ldots, c_n \in \mathsf{Ind}(\mathcal{A})$, for some $n \geq 1$, such that

- $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \models r(c_{i-1}, c_i)$ for each $i \in \{1, \ldots, n\}$,
- $\mathsf{IT}'_{\mathcal{O},\mathcal{A}}(c_0) \subseteq \{t_a\}$,
- $\mathsf{IT}'_{\mathcal{O},\mathcal{A}}(c_i) \subseteq \{t_a, t_b\}$ for each $i \in \{1, \ldots, n-1\}$, and
- $\mathsf{IT}'_{\mathcal{O},\mathcal{A}}(c_n) \subseteq \{t_b\}$.

Here, we let $\mathsf{IT}'_{\mathcal{O},\mathcal{A}}(c) := \mathsf{IT}_{\mathcal{O},\mathcal{A}}(c)$ for $c \in \mathsf{Ind}(\mathcal{A}) \setminus \{a,b\}$, and $\mathsf{IT}'_{\mathcal{O},\mathcal{A}}(c) := \{t_c\}$ for $c \in \{a,b\}$.

Lemma 28. *Let $\mathcal{A}$ be a data instance for $\mathcal{O}$ such that $\mathcal{O} \cup \mathcal{A}$ is satisfiable. Let $r \in \mathsf{N_R}$, $a \in \mathsf{N_I}$, and $b \in \mathsf{N_I} \cup \bigcup_{\mathcal{D} \in \mathsf{DT}} \Delta^{\mathcal{D}}$. Then, $\mathcal{O},\mathcal{A} \models r(a,b)$ iff one of the following applies:*

1. *$\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \models r(a,b)$; or*
2. *$\mathsf{IT}_{\mathcal{O},\mathcal{A}}(a) \subseteq \mathsf{N_E}(\mathcal{O})$, $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(b) \subseteq \mathsf{N_E}(\mathcal{O})$, and for all $t_a \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$ and $t_b \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(b)$ there exists an (r,a,b,t_a,t_b)-path in $\mathcal{O},\mathcal{A}$.*

Proof. *"If"* If $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \models r(a,b)$, then $\mathcal{O},\mathcal{A} \models r(a,b)$. In the following, we assume that $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(a), \mathsf{IT}_{\mathcal{O},\mathcal{A}}(b) \subseteq \mathsf{N_E}(\mathcal{O})$, and that for all $t_a \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$ and $t_b \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(b)$ there exists an (r,a,b,t_a,t_b)-path in $\mathcal{O},\mathcal{A}$.

To show that $\mathcal{O},\mathcal{A} \models r(a,b)$, let $\mathcal{I}$ be a model of $\mathcal{O} \cup \mathcal{A}$. For each $c \in \mathsf{Ind}(\mathcal{A})$, let t_c be the item type realized by $c^{\mathcal{I}}$. By Lemma 23, we have $t_c \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(c)$.

Now, let $c_0, c_1, \ldots, c_n$ be an (r,a,b,t_a,t_b)-path in $\mathcal{O},\mathcal{A}$. Then, $\mathcal{I} \models r(c_i, c_{i+1})$ for all $i < n$, $\mathsf{IT}'_{\mathcal{O},\mathcal{A}}(c_0) \subseteq \{t_a\}$, $\mathsf{IT}'_{\mathcal{O},\mathcal{A}}(c_n) \subseteq \{t_b\}$, and $\mathsf{IT}'_{\mathcal{O},\mathcal{A}}(c_i) \subseteq \{t_a, t_b\}$ for each $i \leq n$. Let i be the smallest index $\leq n$ with $t_{c_i} = t_b$. Then, $t_{c_{i-1}} = t_a$, and hence $(a^{\mathcal{I}}, b^{\mathcal{I}}) = (t_a, t_b) = (c_{i-1}^{\mathcal{I}}, c_i^{\mathcal{I}})$. Since $\mathcal{I} \models r(c_{i-1}, c_i)$, this implies $\mathcal{I} \models r(a,b)$.

This shows that $r(a,b)$ is true in every model of $\mathcal{O} \cup \mathcal{A}$, hence $\mathcal{O},\mathcal{A} \models r(a,b)$.

"Only if" We prove the contrapositive. Suppose that $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \not\models r(a,b)$, and that one of the following applies:

- *Case 1:* $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(a) \not\subseteq \mathsf{N_E}(\mathcal{O})$.
- *Case 2:* $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(b) \not\subseteq \mathsf{N_E}(\mathcal{O})$.
- *Case 3:* Cases 1 and 2 do not apply, and there are $t_a \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$ and $t_b \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(b)$ such that there is no (r,a,b,t_a,t_b)-path in $\mathcal{O},\mathcal{A}$.

Case 1: In this case, there is a $t_a \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$ with $t_a \notin \mathsf{N_E}$. Note that this implies $t_a = \star$. Since $\mathcal{O} \cup \mathcal{A}$ is satisfiable, we can pick an element $t_c \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(c)$ for each $c \in \mathsf{Ind}(\mathcal{A}) \setminus \{a\}$. By Lemma 23, there is a model $\mathcal{I}$ of $\mathcal{O} \cup \mathcal{A}$ such that

1. $a^{\mathcal{I}}$ realizes t, and
2. if $t_a = \star$, then $a^{\mathcal{I}} \neq c^{\mathcal{I}}$ for all $c \in \mathsf{Ind}(\mathcal{A})$ with $a \neq c$.

Without loss of generality, we may assume that $\mathcal{I}$ is an inclusion-minimal model. In such a model, we have $\mathcal{I} \models r(a,b)$ iff $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \models r(c,d)$ for $c,d \in \mathsf{Ind}(\mathcal{A})$ with $(a^{\mathcal{I}}, b^{\mathcal{I}}) = (c^{\mathcal{I}}, d^{\mathcal{I}})$. Since $t_a = \star$, and by the properties of $\mathcal{I}$ stated above, there is no $c \in \mathsf{Ind}(\mathcal{A})$ with $c^{\mathcal{I}} = a^{\mathcal{I}}$, and therefore $\mathcal{I} \not\models r(a,b)$. As $\mathcal{I}$ is a model of $\mathcal{O} \cup \mathcal{A}$, this implies $\mathcal{O},\mathcal{A} \not\models r(a,b)$.

Case 2: Similar to case 1.

Case 3: Let $t_a \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a) \subseteq \mathsf{N_E}(\mathcal{O})$ and $t_b \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(b) \subseteq \mathsf{N_E}(\mathcal{O})$ such that there is no (r,a,b,t_a,t_b)-path in $\mathcal{O},\mathcal{A}$. Consider the directed graph $G = (V,E)$ defined as follows:

- the vertex set V consists of all the elements $c \in \mathsf{Ind}(\mathcal{A})$ with $\mathsf{IT}'_{\mathcal{O},\mathcal{A}}(c) \subseteq \{t_a, t_b\}$;
- the edge set E consists of all pairs $(c,d) \in V \times V$ with $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \models r(c,d)$.

Here, $\mathsf{IT}'_{\mathcal{O},\mathcal{A}}(c)$ is defined as in Definition 27. Let $S \subseteq V$ consist of all those $c \in V$ with $\mathsf{IT}'_{\mathcal{O},\mathcal{A}}(c) \subseteq \{t_a\}$, and let S^* be the set of all $c \in V$ that are reachable from S in G. Since there is no (r,a,b,t_a,t_b)-path in $\mathcal{O},\mathcal{A}$, we have

$$t_a \in \mathsf{IT}'_{\mathcal{O},\mathcal{A}}(c) \quad \text{for all } c \in S^*. \tag{1}$$

Since all those $c \in V$ with $t_b \notin \mathsf{IT}'_{\mathcal{O},\mathcal{A}}(c)$ occur in $S \subseteq S^*$, we also have

$$t_b \in \mathsf{IT}'_{\mathcal{O},\mathcal{A}}(c) \quad \text{for all } c \in V \setminus S^*. \tag{2}$$

Also note that $a \in S \subseteq S^*$ and $b \in V \setminus S^*$ (for the latter, recall that $\mathsf{IT}'_{\mathcal{O},\mathcal{A}}(b) = \{t_b\}$, hence the presence of b in S^* would imply an (r,a,b,t_a,t_b)-path in $\mathcal{O},\mathcal{A}$, contradicting our assumption). For each $c \in \mathsf{Ind}(\mathcal{A})$, let

- $t_c = t_a$ if $c \in S^*$,
- $t_c = t_b$ if $c \in V \setminus S^*$,
- $t_c \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(c) \setminus \{t_a, t_b\}$ if $c \in \mathsf{Ind}(\mathcal{A}) \setminus V$.

Notice that there is no edge (c,d) in G with $(t_c, t_d) = (t_a, t_b)$, because by the definition of the t_c every such edge (c,d) must have the property that $c \in S^*$ and $d \in V \setminus S^*$, but by the definition of S^* there is no edge from S^* to $V \setminus S^*$.

By Lemma 23, there is a model $\mathcal{I}$ of $\mathcal{O} \cup \mathcal{A}$ such that for each $c \in \mathsf{Ind}(\mathcal{A})$,

- $c^{\mathcal{I}}$ realizes t_c, and
- if $t_c = \star$, then $c^{\mathcal{I}} \neq d^{\mathcal{I}}$ for all $d \in \mathsf{Ind}(\mathcal{A})$ with $c \neq d$.

Without loss of generality, we may assume that $\mathcal{I}$ is an inclusion-minimal such model. In such a model, we have $\mathcal{I} \models r(a,b)$ iff there is an edge (c,d) in G with $(c^{\mathcal{I}}, d^{\mathcal{I}}) = (a^{\mathcal{I}}, b^{\mathcal{I}}) = (t_a, t_b)$. But from our construction of the t_c and the properties of $\mathcal{I}$ it follows that there is no edge (c,d) in G with $(c^{\mathcal{I}}, d^{\mathcal{I}}) = (t_a, t_b)$. Consequently, $\mathcal{I} \not\models r(a,b)$, which implies $\mathcal{O},\mathcal{A} \not\models r(a,b)$, as desired. ❏

Lemma 29. *For every schema.org-ontology $\mathcal{O}$ and every $r \in \mathsf{N_R}$, one can construct in polynomial time a datalog-rewriting of $(\mathcal{O}, r(x,y))$.*

Proof. As mentioned at the end of Section 4 it suffices to construct a rewriting that works for data instances $\mathcal{A}$ such that $\mathcal{O} \cup \mathcal{A}$ is satisfiable. Clearly, if $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \models r(a,b)$ for a data instance $\mathcal{A}$ for $\mathcal{O}$ and $a, b \in \mathsf{Ind}(\mathcal{A})$, then $\mathcal{O}, \mathcal{A} \models r(a,b)$. On the other hand, if $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \not\models r(a,b)$, then we need to check that there always exists an assertion $r(c,d) \in \mathcal{A}$ such that c and d are forced onto the same individuals as a and b, respectively. By Lemma 28 we have that $\mathcal{O}, \mathcal{A} \models r(a,b)$ iff one of the following applies:

1. $\mathcal{O}_{\mathsf{basic}}, \mathcal{A} \models r(a,b)$; or
2. $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(a) \subseteq \mathsf{N_E}(\mathcal{O})$, $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(b) \subseteq \mathsf{N_E}(\mathcal{O})$, and for all $t_a \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(a)$ and $t_b \in \mathsf{IT}_{\mathcal{O},\mathcal{A}}(b)$ there exists an (r, a, b, t_a, t_b)-path in $\mathcal{O}, \mathcal{A}$.

We now construct a datalog program implementing these checks. As in Lemma 26, we start by computing a non-recursive datalog rewriting of $(\mathcal{O}_{\mathsf{basic}}, r(x,y))$, and non-recursive datalog programs Π_t, for each item type t over $\mathcal{O}$, as guaranteed by Lemma 24. By Proposition 20 and Lemma 24, this is possible in polynomial time. Let $\mathsf{certain}_r$ and $\overline{\mathsf{it}}_t$ be the goal predicates of Π_A and Π_t, respectively.

Now, let Π be the datalog program containing Π_r and Π_t, for each item type t over $\mathcal{O}$, and the following additional sets of rules. First, for all $t, t' \in \mathsf{N_E}(\mathcal{O})$, we include the following rules to check the existence of an (r, x, y, t, t')-path in $\mathcal{O}, \mathcal{A}$:[4]

1. $S_{t,t'}(x,u) \leftarrow \bigwedge_{t'' \neq t} \overline{\mathsf{it}}_{t''}(u)$;
   $S_{t,t'}(x,u) \leftarrow x = u$;
2. $V_{t,t'}(x,y,u) \leftarrow \bigwedge_{t'' \notin \{t,t'\}} \overline{\mathsf{it}}_{t''}(u)$;
   $V_{t,t'}(x,y,u) \leftarrow x = u$;
   $V_{t,t'}(x,y,u) \leftarrow y = u$;
3. $P_{t,t'}(x,y,u) \leftarrow S_{t,t'}(x,v) \wedge \mathsf{certain}_r(v,u) \wedge V_{t,t'}(x,y,u)$;
4. $P_{t,t'}(x,y,u) \leftarrow P_{t,t'}(x,y,v) \wedge \mathsf{certain}_r(v,u) \wedge V_{t,t'}(x,y,u)$;
5. $\mathsf{goal}_{t,t'}(x,y) \leftarrow P_{t,t'}(x,y,u) \wedge \bigwedge_{t'' \neq t'} \overline{\mathsf{it}}_{t''}(u)$;
6. $\mathsf{goal}_{t,t'}(x,y) \leftarrow P_{t,t'}(x,y,u) \wedge u = y$.

The rules in 1 add to $S_{t,t'}(x,\cdot)$ all u with $\mathsf{IT}'_{\mathcal{O},\mathcal{A}}(u) \subseteq \{t\}$ (i.e., $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(u) \subseteq \{t\}$ or $u = x$). Similarly, the rules in 2 add to $V_{t,t'}(x,y,\cdot)$ all u with $\mathsf{IT}'_{\mathcal{O},\mathcal{A}}(u) \subseteq \{t,t'\}$ (i.e., $\mathsf{IT}_{\mathcal{O},\mathcal{A}}(u) \subseteq \{t,t'\}$ or $u \in \{x,y\}$). The rules in 3 and 4 add to $P_{t,t'}(x,y,\cdot)$ all u such that there is an r-path starting in an element in $S_{t,t'}(x,\cdot)$, ending in u, and having all its intermediate vertices in $V_{t,t'}(x,y,\cdot)$. Finally, rules 5 and 6 check whether some element in $P_{t,t'}(x,y,\cdot)$ is the endpoint of a (r,x,y,t,t')-path. In particular, $\mathsf{goal}_{t,t'}(x,y)$ is true iff there is an (r,x,y,t,t')-path in $\mathcal{O}, \mathcal{A}$.

We now use the above rules to construct the final rewriting. To this end, we add the following rules:

- $\mathsf{goal}(x,y) \leftarrow \mathsf{certain}_r(x,y)$;
- $\mathsf{goal}(x,y) \leftarrow \bigwedge_{t,t' \in \mathsf{N_E}(\mathcal{O})} R_{t,t'}(x,y)$;
- $R_{t,t'}(x,y) \leftarrow \overline{\mathsf{it}}_t(x)$ and $R_{t,t'}(x,y) \leftarrow \overline{\mathsf{it}}_{t'}(y)$ for all item types t, t' over $\mathcal{O}$;
- $R_{t,t'}(x,y) \leftarrow \mathsf{goal}_{t,t'}(x,y)$ for all $t, t' \in \mathsf{N_E}(\mathcal{O})$.

This finishes the construction of the program. Clearly, Π can be computed in polynomial time. The characterization of $\mathcal{O}, \mathcal{A} \models r(a,b)$ at the beginning of the proof implies that $(a,b) \in \Pi(\mathcal{A})$ iff $\mathcal{O}, \mathcal{A} \models r(a,b)$. ❏

[4]To simplify the presentation, we sometimes omit covering variables that occur in the head of a rule, but as before we can easily do this by first adding rules that define the unary predicate of all individual names in $\mathcal{A}$, and then using this unary predicate to "cover" the variables that occur in the head but not in the body of a rule.

We obtain Theorem 15 as a corollary of Lemmas 26 and 29.

Theorem 15 (restated). *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; the rewriting is non-recursive if $q = A(x)$. Moreover, evaluating OMQs from this class is in* PTIME *in combined complexity.*

Proof. Let $\mathcal{O}$ be a schema.org-ontology, and $q(\bar{x})$ a quantifier-free CQ. For each concept name A and role name r in $q(\bar{x})$, let Π_A and Π_r be datalog rewritings of $A(x)$ and $r(x,y)$ with goal predicates goal_A and goal_r, respectively. By Lemmas 26 and 29, such rewritings can be computed in polynomial time. Now, a datalog rewriting of $(\mathcal{O}, q)$ is obtained as the datalog program containing Π_A and Π_r for each concept name A and role name r in $q(\bar{x})$, and the rule

$$\mathsf{goal}(\bar{x}) \leftarrow \phi, \tag{3}$$

where ϕ is obtained from $q(\bar{x})$ by replacing each concept name A by goal_A, and each role name r by goal_r.

Next, we argue that evaluating OMQs $(\mathcal{O}, q)$ with $\mathcal{O}$ a schema.org-ontology and q a quantifier-free query has PTime combined complexity. Given a schema.org-ontology $\mathcal{O}$, a data instance $\mathcal{A}$ for $\mathcal{O}$, a quantifier-free query q, and a tuple $\bar{a}$, we first construct the datalog program Π as described above. We then construct a new program $\Pi_{\bar{a}}$ obtained from Π by substituting $\bar{a}$ for $\bar{x}$ in (3), and replacing $\mathsf{goal}(\bar{a})$ with the unary goal predicate $\mathsf{goal}()$. Then, $\mathcal{O}, \mathcal{A} \models q(\bar{a})$ iff $\Pi_{\bar{a}}(\mathcal{A}) \neq \emptyset$. Inspecting the constructions of the programs Π_A and Π_r, we observe that each rule in $\Pi_{\bar{a}}$ has at most three variables. It follows that $\Pi_{\bar{a}}$ can be evaluated in polynomial time.

Note that, if $q = A(x)$, then Lemma 26 states that a non-recursive datalog rewriting of $(\mathcal{O}, q)$ can be computed in polynomial time. ❏

E Proof of Theorem 16

Theorem 16 For every 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.

Proof. As in the proof of Theorem 10, assume a template $\mathcal{B}$ over signature Σ of concept and role names is given such that for each $b \in \Delta^{\mathcal{B}}$ there is a concept name P_b such that $d \in P_b^{\mathcal{B}}$ iff $d = b$.

Take a fresh concept name A, set $\mathcal{O} = \{A \equiv \{b \mid b \in \Delta^{\mathcal{B}}\}\}$, and define the UCQ q as the disjunction of

- $r(b, b')$ for all $r \in \Sigma$ and $(b, b') \notin r^{\mathcal{B}}$;

- $B(b)$ for all $B \in \Sigma$ and $b \notin B^{\mathcal{B}}$.

We show that the complement of CSP($\mathcal{B}$) and $(\mathcal{O}, q)$ are mutually FO-reducible.

($\Rightarrow$) Assume a data instance $\mathcal{A}$ over Σ is given. We may assume that the individuals b, $b \in \Delta^{\mathcal{B}}$, do not occur in $\mathcal{A}$. If there exist $P_b(a), P_{b'}(a) \in \mathcal{A}$ with $b \neq b'$ then output $\mathcal{A} \not\to \mathcal{B}$. Otherwise replace exhaustively

- $B(a)$ by $B(b)$ if $P_b(a) \in \mathcal{A}$ and $B \in \Sigma \setminus \{P_b \mid b \in \Delta^{\mathcal{B}}\}$;
- $r(a_1, a_2)$ by $r(b, a_2)$ if $P_b(a_1) \in \mathcal{A}$;
- $r(a_1, a_2)$ by $r(a_1, b)$ if $P_b(a_2) \in \mathcal{A}$;

and remove all assertions involving some P_b from $\mathcal{A}$ and add $A(a)$ for all remaining individuals a. Denote by $\mathcal{A}'$ the resulting data instance. It is readily checked that $\mathcal{O}, \mathcal{A}' \models q$ iff $\mathcal{A} \not\to \mathcal{B}$.

($\Leftarrow$) Assume a data instance $\mathcal{A}$ is given. Remove from $\mathcal{A}$ all assertions involving individuals a distinct from b with $b \in \Delta^{\mathcal{B}}$ such that $A(a) \notin \mathcal{A}$. Clearly $\mathcal{O}, \mathcal{A} \models q$ iff $\mathcal{O}, \mathcal{A}' \models q$ for the resulting data instance $\mathcal{A}'$. Now add $P_b(b)$ to $\mathcal{A}'$ for all $b \in \Delta^{\mathcal{B}}$ and remove all assertions with concept or role names not in Σ. Denote the resulting data instance by $\mathcal{A}''$. One can show that $\mathcal{O}, \mathcal{A}' \models q$ iff $\mathcal{A}'' \not\to \mathcal{B}$. ❑

F Proof of Theorem 17

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 $n \geq 2$ 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 $\mathcal{O} \not\models F \sqsubseteq A \sqcup \bigsqcup_{(D, \Delta^{\mathcal{D}}) \in \mathrm{DT}} D$ 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).

Proof. Assume $\mathcal{O}$ is coherent and minimized and the conditions for NP-hardness are satisfied. If $\mathcal{O}$ contains an enumeration definition $A \equiv \{a_1, \dots, a_n\}$ with $n \geq 2$ we prove NP-hardness similarly to the hardness proof in Theorem 9. Differences are that in this case we do not attempt to work within the language of the given ontology $\mathcal{O}$ and that we use enumeration individuals *in* the query instead of existentially quantified variables.

For simplicity, we consider the case in which $A \equiv \{a_0, a_1\} \in \mathcal{O}$. The generalization to arbitrarily many enumeration individuals is straightforward using the ideas from the proof of Theorem 9.

Assume $\varphi = c^0 \wedge \dots \wedge c^n$ is a 2+2-formula in propositional letters $v_0, \dots, v_m$ and let $c^i = u_0^i \vee u_1^i \vee \neg u_2^i \vee \neg u_3^i$ for $i \leq n$. Our aim is to define an data instance $\mathcal{A}_\varphi$ and a Boolean CQ q such that φ is unsatisfiable iff $\mathcal{O}, \mathcal{A}_\varphi \models q$. We represent the formula φ in the data instance $\mathcal{A}_\varphi$ as follows. We use two enumeration individuals, a_0, a_1, all remaining individual names are from $\mathsf{N_I} \setminus \mathsf{N_E}$. In addition we use one fresh role name r. Now we take as in the proof of Theorem 9

- the individual names $v_0, \dots, v_m$ represent variables and the individual names $0, 1$ represent truth constants;
- the individual names c_l^i and b_l^i are used to encode the four literals of each $2+2$ clause c^i, where $i \leq n$ and $l \leq 3$;
- for $i \leq n$ and $l \leq 3$, the assertions
$$r(c_l^i, b_l^i), r(b_l^i, u_l^i), r(c_l^i, u_l^i)$$
and
$$r(c_0^i, c_1^i), r(c_1^i, c_2^i), r(c_2^i, c_3^i)$$
to associate the literals c_l^i of a clause c^i to the variable/truth constant u_l^i.

We further extend $\mathcal{A}_\varphi$ to enforce a truth value for each variable v_i, $i \leq m$. Intuitively, assertions $A(a_i')$ are used to generate a truth value (a_0 or a_1) for v_i, where we identify a_0 with true and a_1 with false. Thus add to $\mathcal{A}_\varphi$ the assertions $A(a_0'), \dots, A(a_k')$ and

- to link variables v_i to a_i' the assertions $r(v_i, a_i')$ for all $i \leq m$;
- to ensure that 0 and 1 have the expected truth values, add to $\mathcal{A}_\varphi$ the assertions $r(1, a_0)$ and $r(0, a_1)$.

Consider the Boolean UCQ (we omit existential quantifiers):

$$q_0 = \bigwedge_{0 \leq i \leq 2} r(x_i, x_{i+1}) \wedge \bigwedge_{0 \leq i \leq 3} \psi_i$$

where

- $\psi_i = r(x_i, z_i) \wedge r(z_i, y_i) \wedge r(x_i, y_i) \wedge \mathsf{ff}(y_i)$ for $i = 0, 1$ and
- $\psi_i = r(x_i, z_i) \wedge r(z_i, y_i) \wedge r(x_i, y_i) \wedge \mathsf{tt}(y_i)$ for $i = 2, 3$ and

where

$$\begin{aligned} \mathsf{tt}(y_i) &= r(y_i, a_1) \\ \mathsf{ff}(y_i) &= r(y_i, a_0) \end{aligned}$$

Then one can show that $\mathcal{O}, \mathcal{A}_\varphi \models q_0$ iff q_0 is not satisfiable.

Assume now that no enumeration definition $A \equiv \{a_1, \dots, a_n\}$ with $n \geq 2$ is in $\mathcal{O}$. Set $\mathsf{N_E}^C(\mathcal{O}) = \{C \mid C \equiv \{c\} \in \mathcal{O}\}$ and set $D_0 = \bigsqcup_{(D, \Delta^{\mathcal{D}}) \in \mathrm{DT}} D$. We prove the following

Claim 1. There exist $F_0 \in \{\mathsf{dom}(r), \mathsf{ran}(r)\}$ and $C_1, \dots, C_k \in \mathsf{N_E}^C(\mathcal{O}) \cup \mathsf{N_C}(\mathcal{O})$, $k \geq 2$, such that $\mathcal{O} \models F_0 \sqsubseteq C_1 \sqcup \dots \sqcup C_k \sqcup D_0$ and for $\mathcal{A} = \{r(a_1, b_1), \dots, r(a_k, b_k)\}$, where $a_i, b_i \in \mathsf{N_I} \setminus \mathsf{N_E}$ there exists a model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{O}$ such that

- if $F_0 = \mathsf{ran}(r)$, then $b_i \in C_i^{\mathcal{I}} \setminus \bigcup_{i \neq j} C_j^{\mathcal{I}}$ for $1 \leq i \leq k$ and
- if $F_0 = \mathsf{dom}(r)$, then $a_i \in C_i^{\mathcal{I}} \setminus \bigcup_{i \neq j} C_j^{\mathcal{I}}$ for $1 \leq i \leq k$.

Proof of Claim 1. Consider the following Condition ($*$): there exist $F' \in \{\mathsf{dom}(s), \mathsf{ran}(s)\}$ and $X \subseteq \mathsf{N_E}^C(\mathcal{O})$ of cardinality at least two, such that

- $\mathcal{O} \models F' \sqsubseteq (\bigsqcup_{C \in X} C) \sqcup D_0$ and
- $F' \sqcap C$ is satisfiable relative to $\mathcal{O}$ for all $C \in X$.

Clearly, if $(*)$ holds, then Claim 1 follows immediately. Now assume $(*)$ does not hold. Consider $F \sqsubseteq A_1 \sqcup \cdots \sqcup A_n \in \mathcal{O}$ such that there are at least two concept names in $\{A_1, \ldots, A_n\}$ and $\mathcal{O} \not\models F \sqsubseteq \{a\} \sqcup D_0$ for any enumeration individual a. Assume w.l.o.g. that $F = \mathsf{ran}(r)$ and that $A_1, \ldots, A_k$ are concept names and $A_{k+1}, \ldots, A_n$ are datatype names. We have

$$\mathcal{O} \models F \sqsubseteq (\bigsqcup_{C \in \mathsf{N_E}^C(\mathcal{O})} C) \sqcup A_1 \sqcup \cdots \sqcup A_k \sqcup D_0$$

By removing 'redundant' concepts starting with A_k and moving $C \in \mathsf{N_E}^C(\mathcal{O})$ vie A_1 we find $X_1 \subseteq \mathsf{N_E}^C(\mathcal{O})$ and $X_2 \subseteq \{A_1, \ldots, A_k\}$ such that

- $\mathcal{O} \models F \sqsubseteq (\bigsqcup_{C \in X_1} C) \sqcup (\bigsqcup_{A \in X_2} A) \sqcup D_0$;
- $\mathsf{ran}(r) \sqcap A \sqcap \neg(\bigsqcup_{C \in \mathsf{N_E}^C(\mathcal{O})} C) \sqcap \neg(\bigsqcup_{B \in X_2, B \neq A} B)$ is satisfiable relative to $\mathcal{O}$ for all $A \in X_2$;
- $\mathsf{ran}(r) \sqcap C \sqcap \neg(\bigsqcup_{A \in X_2} A)$ is satisfiable relative to $\mathcal{O}$ for all $C \in X_1$.

It follows from the conditions for NP-hardness in Theorem 17 that we have $|X_1 \cup X_2| \geq 2$: if $|X_1| = 1$ and $X_2 = \emptyset$, then we have found an enumeration individual a with $\mathcal{O} \not\models F \sqsubseteq \{a\} \sqcup \bigsqcup_{(D, \Delta^{\mathcal{D}}) \in \mathrm{DT}} D$. If $|X_2| = 1$ and $X_1 = \emptyset$, then $\mathcal{O} \models A_i \sqcup \bigsqcup_{(D, \Delta^{\mathcal{D}}) \in \mathrm{DT}} D$ which contradicts the condition that $\mathcal{O}$ is minimized, and that $F \sqsubseteq A_1 \sqcup \cdots \sqcup A_n \in \mathcal{O}$ with at least two concept names in $\{A_1, \ldots, A_n\}$.

Next observe that we cannot have both, $\mathcal{O} \models \mathsf{dom}(r) \sqsubseteq C$ for some $C \in X_1$ and $\mathcal{O} \models \mathsf{dom}(r) \sqsubseteq \bigsqcup_{A \in X_2} A$. Moreover, since $(*)$ does not hold, $\mathcal{O} \not\models \mathsf{dom}(r) \sqsubseteq (\bigsqcup_{C \in X_1} C)$ does not hold unless $\mathcal{O} \models \mathsf{dom}(r) \sqsubseteq C$ for some $C \in X_1$. Thus, we find models $\mathcal{I}_A$, $A \in X_1 \cup X_2$, of $\mathcal{O}$ and $\mathcal{A}_A = \{r_A(a_A, b_A)\}$ such that $\Delta^{\mathcal{I}_A} \subseteq \{a_A^{\mathcal{I}_A}, b_A^{\mathcal{I}_A}\} \cup \mathsf{N_E}(\mathcal{O})$ and

- $s^{\mathcal{I}_A} = \emptyset$ for all roles s with $\mathcal{O} \not\models r \sqsubseteq s$ and all $A \in X_1 \cup X_2$;
- $r^{\mathcal{I}_A} = s^{\mathcal{I}_A} = \{(a_A^{\mathcal{I}_A}, b_A^{\mathcal{I}_A})\}$, for all s with $\mathcal{O} \models r \sqsubseteq s$ and all $A \in X_1 \cup X_2$;
- $a_A^{\mathcal{I}_A} \in A^{\mathcal{I}_A} \setminus (\bigcup_{C \in \mathsf{N_E}^C(\mathcal{O})} C^{\mathcal{I}_A} \cup \bigcup_{B \neq A, B \in X_2} B^{\mathcal{I}_A})$, for all $A \in X_2$.
- $a_A^{\mathcal{I}_A} \in A^{\mathcal{I}_A} \setminus \bigcup_{B \in X_2} B^{\mathcal{I}_A})$, for all $A \in X_1$;
- For all $C \in X_1$, $C^{\mathcal{I}_B} \cap \in A^{\mathcal{I}_B} = \emptyset$ for all $A, B \in X_2$.

It follows that we can take the union $\mathcal{I}$ of the models $\mathcal{I}_A$, $A \in X_1 \cup X_2$, and factorize through the equivalence relation $\sim$ defined by $d_1 \sim d_2$ if $d_1 = d_2$ or there exists $C \in \mathsf{N_E}^C(\mathcal{O})$ such that $d_1, d_2 \in C^{\mathcal{I}}$. The resulting model $\mathcal{I}/\sim$ is as required. This finishes the proof of Claim 1.

Using Claim 1 we prove NP-hardness similarly to the hardness proof in Theorem 9 and above. Again we do not attempt to work within the language of the given ontology $\mathcal{O}$. For simplicity, we consider the following case:

$(*)$ There are $C_1, C_2 \in \mathsf{N_E}^C(\mathcal{O}) \cup \mathsf{N_C}(\mathcal{O})$ such that $\mathcal{O} \models F_0 \sqsubseteq C_1 \sqcup C_2$ and for $\mathcal{A} = \{r(a_1, b_1), r(a_2, b_2)\}$, where $a_i, b_i \in \mathsf{N_I} \setminus \mathsf{N_E}$ there exists a model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{O}$ such that $b_i \in C_i^{\mathcal{I}} \setminus \bigcup_{i \neq j} C_j^{\mathcal{I}}$ for $i = 1, 2$.

The generalization to arbitrarily many disjuncts and datatype names as disjuncts is straightforward using the ideas from the proof of Theorem 9.

Assume $\varphi = c^0 \wedge \cdots \wedge c^n$ is a 2+2-formula in propositional letters $v_0, \ldots, v_m$ and let $c^i = u_0^i \vee u_1^i \vee \neg u_2^i \vee \neg u_3^i$ for $i \leq n$. Our aim is to define an data instance $\mathcal{A}_\varphi$ and a Boolean CQ q such that φ is unsatisfiable iff $\mathcal{O}, \mathcal{A}_\varphi \models q$. We represent the formula φ in the data instance $\mathcal{A}_\varphi$ as follows. We use only two enumeration individuals, a_0, a_1, all remaining individual names are from $\mathsf{N_I} \setminus \mathsf{N_E}$. In addition we use one fresh role name r. Now we take

- the individual names $v_0, \ldots, v_m$ represent variables and the individual names $0, 1$ represent truth constants;
- the individual names c_l^i and b_l^i are used to encode the four literals of each $2+2$ clause c^i, where $i \leq n$ and $l \leq 3$;
- for $i \leq n$ and $l \leq 3$, the assertions
$$r(c_l^i, b_l^i), r(b_l^i, u_l^i), r(c_l^i, u_l^i)$$
and
$$r(c_0^i, c_1^i), r(c_1^i, c_2^i), r(c_2^i, c_3^i)$$
to associate the literals c_l^i of a clause c^i to the variable/truth constant u_l^i.

We further extend $\mathcal{A}_\varphi$ to enforce a truth value for each variable v_i, $i \leq m$. Now $C_1(a)$ stands for true and $C_2(a)$ stands for false. We thus add to $\mathcal{A}_\varphi$ the assertions $r(f_i, a_i)$ for $i \leq m$ and

- to link variables v_i to a_i we add the assertions $r(v_i, a_i)$ for all $i \leq m$;
- to ensure that 0 and 1 have the expected truth values, add to $\mathcal{A}_\varphi$ the assertions $r(1, 1')$, $C_1(1')$ and $r(0, 0')$, $C_2(0')$.

Consider the Boolean UCQ (we omit existential quantifiers):

$$q_0 = \bigwedge_{0 \leq i \leq 2} r(x_i, x_{i+1}) \wedge \bigwedge_{0 \leq i \leq 3} \psi_i$$

where

- $\psi_i = r(x_i, z_i) \wedge r(z_i, y_i) \wedge r(x_i, y_i) \wedge \mathsf{ff}_i(y_i)$ for $i = 0, 1$ and
- $\psi_i = r(x_i, z_i) \wedge r(z_i, y_i) \wedge r(x_i, y_i) \wedge \mathsf{tt}_i(y_i)$ for $i = 2, 3$ and

where

$$\begin{aligned} \mathsf{tt}_i(y_i) &= r(y_i, w_i) \wedge C_1(w_i) \\ \mathsf{ff}_i(y_i) &= r(y_i, w_1) \wedge C_2(w_i) \end{aligned}$$

Then $\mathcal{O}, \mathcal{A}_\varphi \models q_0$ iff q_0 is not satisfiable.

Now assume that the conditions for non-tractability are not satisfied. Assume a UCQ $q = \bigvee_{i \in I} q_i$ is given and assume w.l.o.g. that q does not contain any individual names. We assume that quantified variables in q are all distinct. Let m be the number of variables in q and let X be the set of all pairs $(\mathcal{A},\pi)$ of data instances $\mathcal{A}$ with at most m individuals and mappings π from answer variables of q into $\mathsf{Ind}(\mathcal{A})$ such that $\mathcal{O},\mathcal{A}\models q(\pi(\vec{x}))$. We can regard every such $\mathcal{A}$ as a quantifier-free CQ $q_{\mathcal{A}}$.

We again use the following notation. Set ${\mathsf{N_E}}^C(\mathcal{O})=\{C_c \mid C_c\equiv\{c\}\in\mathcal{O}\}$ and set $D_0=\bigsqcup_{(D,\Delta^{\mathcal{D}})\in\mathsf{DT}} D$. Let for $c\in\mathsf{N_E}(\mathcal{O})$,

$$\begin{array}{rcl}\varphi_c(x) & = & (x=c)\vee\bigvee_{\mathcal{O}\models A\sqsubseteq C_c} A(x)\vee\\ & & \bigvee_{\mathcal{O}\models\mathsf{ran}(r)\sqsubseteq C_c\sqcup D_0}\exists y\, r(y,x)\\ & & \bigvee_{\mathcal{O}\models\mathsf{dom}(r)\sqsubseteq C_c}\exists y\, r(x,y)\end{array}$$

and

$$\varphi_\sim(x,x')=(x=x')\vee\bigvee_{c\in\mathsf{N_E}(\mathcal{O})}\varphi_c(x)\wedge\varphi_c(x')$$

Obtain from $q_{\mathcal{A}}$ the query $q'_{\mathcal{A}}$ by replacing

- every atom $A(y)$ by $\exists y'(\varphi_\sim(y,y')\wedge A(y'))$ and
- every $r(y_1,y_2)$ by
  $$\exists y_1'\exists y_2'(\varphi_\sim(y_1,y_1')\wedge\varphi_\sim(y_2,y_2')\wedge r(y_1',y_2'))$$

Now let Q be the disjunction over all

$$Q_{\mathcal{A},\pi}=\bigwedge_{\pi(x_i)=y}(x_i=y)\wedge q_{\mathcal{A}'}$$

with $(\mathcal{A},\pi)\in X$. It is readily checked that Q is a rewriting of q. ❑

G Proof of Theorem 18

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.

Proof. We consider $\mathcal{O}$ without datatypes. The extension required to include datatypes is straightforward. Let Σ be a finite signature of concept names, role names, and individual names. A Σ-*interpretation* $\mathcal{I}$ is an interpretation in which $X^{\mathcal{I}}=\emptyset$ for all concept and role names not in Σ and in which exactly the individuals a in Σ are interpreted as $a^{\mathcal{I}}\in\Delta^{\mathcal{I}}$. Given Σ-interpretations $\mathcal{I}$ and $\mathcal{J}$ we say that a mapping h from $\Delta^{\mathcal{I}}$ to $\Delta^{\mathcal{J}}$ is a homomorphism if

- $h(a^{\mathcal{I}})=a^{\mathcal{J}}$ for all $a\in\Sigma$;
- $d\in A^{\mathcal{I}}$ implies $h(d)\in A^{\mathcal{J}}$ for all $A\in\Sigma$;
- $(d,d')\in r^{\mathcal{I}}$ implies $(h(a),h(b))\in r^{\mathcal{J}}$, for all $r\in\Sigma$.

We write $\mathcal{I}\rightarrow\mathcal{J}$ iff there exists a homomorphism from $\mathcal{I}$ to $\mathcal{J}$.

Now assume $\mathcal{O}$ and $q(\vec{x},\vec{b})$ are given, where $\vec{x}=x_1,\ldots,x_k$. Let $\vec{a}=a_1\cdots a_k$. Let Σ be the set of all concept and role names in $\mathcal{O}$ and q together with the individuals in $\vec{a}$, $\vec{b}$ and all $\{c\}$ for $c\in\mathsf{N_E}(\mathcal{O})$. In what follows we assume w.l.o.g. that data instance $\mathcal{A}$ contain the individuals in Σ.

Since q is qvar-acyclic we can construct in polynomial time a concept C_q in the description logic $\mathcal{ALCIUO}$ which extends $\mathcal{ALC}$ with inverse roles, the universal role u and nominals $\{a_1\},\ldots,\{a_k\}$, $\{b_1\},\ldots,b_k\}$, and $\{c\}$ for $c\in\mathsf{N_E}(\mathcal{O})$ such that for every data instance $\mathcal{A}$ for $\mathcal{O}$ (and with $\vec{a}$, $\vec{b}$, and c in $\mathsf{Ind}(\mathcal{A})$ for $c\in\mathsf{N_E}(\mathcal{O})$), $\mathcal{O},\mathcal{A}\models q(\vec{a},\vec{b})$ iff $C^{\mathcal{I}}\neq\emptyset$ for all models $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{A}$. In what follows we regard $\mathsf{ran}(r)$ and $\mathsf{dom}(r)$ as the $\mathcal{ALCI}$ concepts $\exists r.\top$ and $\exists r^-.\top$, respectively. We are going to construct a set Γ of templates $\mathcal{B}$ as follows: denote by $\mathsf{sub}(\mathcal{O},q)$ the closure under single negation of the set of (subconcepts of) concepts that occur in $\mathcal{O}$ or C_q. A $\mathcal{O}$-*type* t is a subset of $\mathsf{sub}(\mathcal{O},q)$ such that there exists a model $\mathcal{I}$ of $\mathcal{O}$ (in particular, $c^{\mathcal{I}}=c$ for all $c\in\mathsf{N_E}(\mathcal{O})$) and $d\in\Delta^{\mathcal{I}}$ such that

$$t=\mathsf{tp}_{\mathcal{I}}(d):=\{D\in\mathsf{sub}(\mathcal{O},q)\mid d\in D^{\mathcal{I}}\}$$

We call $\mathsf{tp}_{\mathcal{I}}(d)$ the $\mathcal{O}$-type of d. For $\mathcal{O}$-types t_1,t_2 and role r we set $t_1\rightsquigarrow_r t_2$ iff there exists a model $\mathcal{I}$ of $\mathcal{O}$ such that $\mathsf{tp}_{\mathcal{I}}(d_1)=t_1$, $\mathsf{tp}_{\mathcal{I}}(d_2)=t_2$, and $(d_1,d_2)\in r^{\mathcal{I}}$. A set T of $\mathcal{O}$-types is *complete* if there exists a model $\mathcal{I}$ of $\mathcal{O}$ such that

$$T=\{\mathsf{tp}_{\mathcal{I}}(d)\mid d\in\Delta^{\mathcal{I}}\}.$$

From each complete set T of $\mathcal{O}$-types T we construct a Σ-template $\mathfrak{B}_T$ as follows: let $\Delta^{\mathcal{B}_T}=T$ and

- $a^{\mathcal{B}_T}=t$ for the unique t with $a\in t$, for all $a\in\Sigma$;
- $t\in A^{\mathcal{B}_T}$ if $A\in t$, for all $A\in\Sigma$;
- $(t,t')\in r^{\mathcal{B}_T}$ if $t\rightsquigarrow_r t'$, for all $r\in\Sigma$.

Now let Γ be the set of all $\mathcal{B}_T$ with T a *maximal* complete T such that $C_q^{\mathcal{B}_T}=\emptyset$. Γ contains at most exponentially many distinct templates of at most exponential size and can be constructed in exponential time. Thus, it remains to prove that $(\mathcal{O},q(\vec{x}))$ and the complement of CSP(Γ) are mutually FO-reducible. For a data instance $\mathcal{A}$ with $\vec{a}$, $\vec{b}$, and c in $\mathsf{Ind}(\mathcal{A})$ for all $c\in\mathsf{N_E}(\mathcal{O})$ we denote by $\mathcal{J}_{\mathcal{A}}$ the Σ-interpretation with

- $\Delta^{\mathcal{J}_{\mathcal{A}}}=\mathsf{Ind}(\mathcal{A})$;
- $a^{\mathcal{J}_{\mathcal{A}}}=a$, for all $a\in\Sigma$;
- $A^{\mathcal{J}_{\mathcal{A}}}=\{a\in\mathsf{Ind}(\mathcal{A})\mid A(a)\in\mathcal{A}\}$, for $A\in\Sigma$;
- $r^{\mathcal{J}_{\mathcal{A}}}=\{(a,b)\in\mathsf{Ind}(\mathcal{A})^2\mid r(a,b)\in\mathcal{A}\}$, for $r\in\Sigma$.

Now one can show the following.

Claim 1. For any data instance $\mathcal{A}$ for $\mathcal{O}$: $\mathcal{O},\mathcal{A}\models q(\vec{a})$ iff $\mathcal{J}_{\mathcal{A}}\not\rightarrow\mathcal{B}$ for any $\mathcal{B}\in\Gamma$.

For a Σ-interpretation $\mathcal{I}$, we denote by $\mathcal{A}_{\mathcal{I}}$ the ABox corresponding to $\mathcal{I}$. One can show the following:

Claim 2. For any Σ-interpretation $\mathcal{I}$, $\mathcal{I}\not\rightarrow\mathcal{B}$ for any $\mathcal{B}\in\Gamma$ iff $\mathcal{O},\mathcal{A}_{\mathcal{I}}\models q(\vec{a})$. ❑