Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

On the Containment of SPARQL Queries under Entailment Regimes

Melisachew Wudage Chekol
Data and Web Science Group
University of Mannheim
Mannheim, Germany
mel@informatik.uni-mannheim.de

Abstract

Most description logics (DL) query languages allow instance retrieval from an ABox. However, SPARQL is a schema query language allowing access to the TBox (in addition to the ABox). Moreover, its entailment regimes enable to take into account knowledge inferred from knowledge bases in the query answering process. This provides a new perspective for the containment problem. In this paper, we study the containment of SPARQL queries over OWL EL axioms under entailment. OWL EL is the language used by many large scale ontologies and is based on $\mathcal{EL}^{++}$. The main contribution is a novel approach to rewriting queries using SPARQL property paths and the μ-calculus in order to reduce containment test under entailment into validity check in the μ-calculus.

Introduction

Recently, the study of SPARQL query containment has attracted a lot of attention. In (Kostylev et al. 2015), the authors explore the complexity of containment and evaluation problems for fragments of SPARQL 1.1 property paths. Interestingly, in their study they allow negated property paths. The study in (Pichler and Skritek 2014) provides complexity analysis for several fragments of SPARQL: the results ranging from NP-completeness for AND-UNION queries to undecidability for the full SPARQL. Additionally, in (Letelier et al. 2013) the containment and optimization of OPTIONAL queries is investigated while providing a Π_2^P-complete complexity for query subsumption (a solution ρ_1 subsumes another solution ρ_2 , if ρ_2 extends ρ_1 with more variables). However, these studies do not consider the containment problem under schema nor entailment regimes. On the contrary, the authors in (Chekol et al. 2012b) and (Chekol et al. 2012a) address containment under $\mathcal{SHI}$ schema axioms and RDFS entailment regime respectively and establish a double exponential upper bound complexity. For this study, we consider the OWL DL entailment.

Automata theoretic notions and a reduction into validity test in a logic have been widely used to address the problem of query answering and containment (Kostylev et al. 2015; Chekol et al. 2012b; Calvanese, Giacomo, and Lenzerini 2008; Genevès, Layaïda, and Schmitt 2007; Krötzsch and Rudolph 2007; Calvanese et al. 2000). Contrary to the automata techniques, the logic based approaches are fairly implementable. In this respect, recently in (Genevès and Schmitt 2015), the authors study logical combinators whose benefit is to provide an exponential gain in succinctness in terms of the size of a logical formula. This allows us to study containment for expressive query languages in exponential-time, even though their direct formulation into the underlying logic results in an exponential blow up of the formula size. Consequently, in this paper, we take advantage of this approach to study the containment of SPARQL queries over OWL EL ontologies under OWL DL entailment by a reduction into the μ-calculus. OWL 2 has three tractable profiles called OWL EL, OWL QL, and OWL RL. In particular, we are interested in OWL EL which is based on the description logic $\mathcal{EL}^{++}$ and is a maximal subset of OWL 2 DL in which reasoning problems can be decided in polynomial time. $\mathcal{EL}^{++}$ is very attractive and has been used widely as a language of choice for terminological reasoning in biomedical applications (Kazakov, Krötzsch, and Simancik 2014; Baader, Brandt, and Lutz 2005). Query answering over OWL EL ontologies under the OWL DL entailment regime can be done via ontology-based data access (OBDA – query answering in the presence of ontologies) which requires queries to be expanded using the terminological part of the ontology (Hansen et al. 2015; Bienvenu, Lutz, and Wolter 2012). Likewise, checking query containment under an entailment regime can be reduced to testing the containment of the rewritten or expanded queries. Indeed, for conjunctive queries query containment and query answering are equivalent problems. Thus, containment can be reduced to query answering as shown in (Calvanese, Giacomo, and Lenzerini 2008).

The fact that conjunctive query answering over unrestricted $\mathcal{EL}^{++}$ ontologies is undecidable was proved at the same time and independently by (Krötzsch and Rudolph 2007) and by (Rosati 2007). In particular combining role atoms in queries and complex role inclusion axioms can make reasoning more difficult. Thus, for this work, we consider OWL EL without role composition and nominals, precisely, the fragment based on the $\mathcal{ELH}_{\bot}^{dr}$ ($\mathcal{EL}$ extended with role hierarchies, domain and range restrictions, and a bottom concept) (Baader, Brandt, and Lutz 2008). With this restriction, we obtained a double exponential upper bound in the

presence of blank nodes for AND-UNION SPARQL queries and triple exponential for OPTIONAL pattern queries. However, the complexity bounds drop by exponentiation when blank nodes do not appear in the right-hand side query. The complexity bounds do not reflect the flexibility of our approach. On one hand, the size of the encodings can be reduced by upto exponentiation by using logical combinators (Genevès and Schmitt 2015). On the other hand, we have showed that our approach is easily extensible to all of the OWL profiles and other expressive description logics. In addition, by implementing a query rewriter and schema parser, we can take advantage of the implementation in http://sparql-qc-bench.inrialpes.fr/. Due to space limitations, proofs and details are omitted.

Preliminaries

An RDF graph contains a set of triples of the form $(\mathbf{I} \cup \mathbf{B}) \times \mathbf{I} \times (\mathbf{I} \cup \mathbf{B} \cup \mathbf{L})$ denoted by $\mathbf{IB} \times \mathbf{I} \times \mathbf{IBL}$, where $\mathbf{I}$, $\mathbf{B}$, and $\mathbf{L}$ are three disjoint infinite sets denoting the set of IRIs (identifying a resource), blank nodes (denoting an unidentified resource) and literals (a character string or some other type of data) respectively. We use an abstract syntax (a, b, c) to denote an RDF triple, where the *subject* $a \in \mathbf{IB}$, *predicate* $b \in \mathbf{I}$, and *object* $c \in \mathbf{IBL}$, to denote RDF triples. In addition, c can be a list denoted by $(c_1, \ldots, c_i, \ldots, c_n)$ and $c_i \in \mathbf{IBL}$. Since SPARQL's syntax is based on that of RDF's, next, we present an RDF representation of OWL EL (as done in (Bischof et al. 2014)).

OWL EL classes, properties, and individuals are represented by either $\mathbf{I}$ or $\mathbf{B}$ whereas complex classes and property expressions are represented by blank nodes $\mathbf{B}$. Further, we denote concepts and properties in OWL EL by $\mathbf{C}$, $\mathbf{D} \in \mathbf{IB}$ and $\mathbf{P}$, $\mathbf{Q} \in \mathbf{IB}$ respectively. For compact presentation, we provide shorthand names (in parenthesis) for RDF(S) and OWL vocabularies: `rdf:type` (`a`), `rdfs:subClassOf` (`sc`), `rdfs:domain` (`dom`), `rdfs:subPropertyOf` (`sp`), `rdfs:range` (`range`), `owl:members` (`members`), `owl:equivalentProperty` (`eqp`), `owl:equivalentClass` (`eqc`), `owl:onProperty` (`onp`), `owl:someValuesFrom` (`svf`), `owl:intersectionOf` (`int`), `owl:Thing` (`Thing`), `owl:topObjectProperty` (`topProp`), and `owl:allDisjointClasses` (`alldsjnt`).

Definition 1. *An OWL EL ontology G uses the following triples to represent axioms:*

$(\mathbf{IB}, \mathbf{P}, \mathbf{IBL})$, $(\mathbf{IB}, \mathtt{a}, \mathbf{C})$, $(\mathbf{C}, \mathtt{sc}, \mathbf{D})$, $(\mathbf{C}, \mathtt{eqc}, \mathbf{D})$,
$(\mathbf{IB}, \mathtt{onp}, \mathbf{P})(\mathbf{IB}, \mathtt{svf}, \mathbf{D})$, $(\mathbf{P}, \mathtt{dom}, \mathbf{C})$, $(\mathbf{P}, \mathtt{range}, \mathbf{C})$,
$(\mathbf{C}, \mathtt{int}, (C_1, \ldots, C_n))$, *where* $C_1, \ldots, C_n \in \mathbf{IB}$,
$(\mathbf{P}, \mathtt{sp}, \mathbf{Q})$, $(\mathbf{P}, \mathtt{eqp}, \mathbf{Q})$, *and* $(\mathbf{B}, \mathtt{a}, \mathtt{alldsjnt})$,
$(\mathbf{B}, \mathtt{members}, (C_1, \ldots, C_n))$.

To provide the semantics of OWL EL ontologies, we extend the notion of universal model (a structure that realizes the entailments of an ontology) which has been used to provide semantics for OWL QL (Bischof et al. 2014). Entailment of OWL EL is defined model theoretically as usual. We first define RDF-based inference rules. A part of these rules is shown below, for a complete list we refer the reader to (Bischof et al. 2014; Krötzsch 2011). Applying these rules may lead to a universal model which can be infinite in size. The *closure* $cl(G)$ of an OWL EL ontology G is a possibly infinite RDF graph obtained from G by exhaustively applying the inference rules. For compact presentation, we do not display the **AND** operator between the triples.

$\Rightarrow (b, \mathtt{a}, \mathtt{Thing})$
$(x, \mathtt{a}, b)(b, \mathtt{int}, (C_1, \ldots, C_i, \ldots, C_n)) \Rightarrow (x, \mathtt{a}, C_i)$
$(x, \mathtt{a}, b)(b, \mathtt{onp}, P)(b, \mathtt{svf}, C) \Rightarrow (x, P, b_1)(b_1, \mathtt{a}, C)$
$(x, \mathtt{a}, C)(C, \mathtt{sc}, D) \Rightarrow (x, \mathtt{a}, D)$
$(x, \mathtt{a}, C)(C, \mathtt{eqc}, D) \Rightarrow (x, \mathtt{a}, D)$
$(x, \mathtt{a}, C)(D, \mathtt{eqc}, C) \Rightarrow (x, \mathtt{a}, D)$
$(x, P, y)(C, \mathtt{onp}, P)(C, \mathtt{svf}, \mathtt{Thing}) \Rightarrow (x, \mathtt{a}, C)$
$(x, P, y)(P, \mathtt{dom}, C) \Rightarrow (x, \mathtt{a}, C)$
$(x, P, y)(P, \mathtt{range}, C) \Rightarrow (y, \mathtt{a}, C)$
$(x, P, y)(P, \mathtt{sp}, Q) \Rightarrow (x, Q, y)$
$(x, P, y)(P, \mathtt{eqp}, Q) \Rightarrow (x, Q, y)$
$(x, P, y)(Q, \mathtt{eqp}, P) \Rightarrow (x, Q, y)$
$(x, \mathtt{a}, \mathtt{Thing})(y, \mathtt{a}, \mathtt{Thing}) \Rightarrow (x, \mathtt{topProp}, y)$

In the above rules, b, $b_1 \in \mathbf{B}$ and x, y, C, C_i, C_n, D, P, $Q \in \mathbf{V}$ are SPARQL variables as defined in the next section. An OWL ontology G is *inconsistent* if there is a mapping for the triple(s): $\{(x, \mathtt{a}, \mathtt{owl{:}Nothing})\}$ or $\{(x, \mathtt{a}, C_i), (x, \mathtt{a}, C_j), (b, \mathtt{a}, \mathtt{alldsjnt}), (b, \mathtt{members}, (C_1, \ldots, C_i, \ldots, C_j, \ldots, C_n))\}$. When so, every OWL axiom is a logical consequence, and there is no universal model. SPARQL queries are evaluated over OWL EL ontologies under OWL DL entailment where the OWL axioms are evaluated under OWL Direct Semantics (Glimm 2011; Glimm and Krötzsch 2010).

SPARQL queries are formed inductively from path patterns which in turn are defined inductively from *path expressions*, i.e., $\mathsf{PP} \in \mathbf{IBV} \times e\mathbf{V} \times \mathbf{IBLV}$. In PP, $\mathbf{V}$ is a set of variables disjoint with $\mathbf{IBL}$ and e is a property path[1] inductively defined as: $e ::= \mathbf{I} \mid e? \mid e^- \mid e \circ e' \mid e \circ\!\!- e' \mid e^+ \mid e^*$.

Definition 2. *A path pattern (simply query pattern) q is defined inductively: $q ::= \mathsf{PP} \mid q$ **AND** $q' \mid q$ **UNION** $q' \mid q$ **OPT** q'. We use the notation* cPP *for conjunctive,* uPP *for union, and* oPP *for optional path queries.*

Additionally, we define a basic graph pattern (BGP) as: $\mathsf{BGP} ::= \mathbf{IBV} \times \mathbf{IV} \times \mathbf{IBLV} \mid \mathsf{BGP}$ **AND** BGP'. Alternatively, we use a dot (.) to denote the operator **AND** . We consider SELECT DISTINCT queries which are of the form $q(W)$ such that W is a tuple of variables in $\mathbf{V}$ which are called *distinguished variables*, and q is a path pattern. The variables $\mathbf{V} \setminus W$ are called *non-distinguished variables*. However, non-distinguished variables are not a part of SPARQL. We consider a class of OPTIONAL (oPP)

[1] http://www.w3.org/TR/sparql11-query/

queries which are attractive both from theoretical complexity and optimization point of view. This class, known as *well-designed OPT patterns*, is first introduced in (Pérez, Arenas, and Gutierrez 2009) and well studied in (Kostylev et al. 2015).

Definition 3. *A query q is well-designed if for every subpattern $q' = (q_1 \ \mathbf{OPT}\ q_2)$ of q and every variable x occurring in q, it holds that: if x occurs inside q_2 and outside q', then x also occurs inside q_1.*

For example, the pattern (x, a, b) **OPT** $((y, c, d)$ **AND** $(x, e, z))$ is well-designed as the variable x occurs inside and outside the OPT operator. SPARQL has multiset (or bag) semantics, however, when dealing with containment, we consider set semantics. This is due to the undecidability of union of conjunctive queries under bag semantics (Ioannidis and Ramakrishnan 1995). We present a set semantics for SPARQL based on simple entailment (i.e., without taking reasoning into account).

Definition 4. *Let G be an RDF graph and q a SPARQL query pattern. The set $\mathsf{A}(q, G)$ of answers of q in G is defined inductively as:*

$$\begin{aligned}
\mathsf{A}((x, \mathbf{I}, y), G) &= \{\rho \mid (\rho(x), \rho(\mathbf{I}), \rho(y)) \in G\} \\
\mathsf{A}((x, e?, y), G) &= \{\rho \mid \rho(x) = \rho(y)\} \cup \mathsf{A}((x, e, y), G) \\
\mathsf{A}((x, e^-, y), G) &= \mathsf{A}((y, e, x), G) \\
\mathsf{A}((x, e \mid e', y), G) &= \mathsf{A}((x, e, y), G) \cup \mathsf{A}((x, e', y), G) \\
\mathsf{A}((x, e \circ e', y), G) &= \exists n : \mathsf{A}((x, e, n), G) \bowtie \mathsf{A}((n, e', y), G) \\
\mathsf{A}((x, e^+, y), G) &= \bigcup_{i \geq 1} \mathsf{A}((x, e^i, y), G) \\
\mathsf{A}((x, e^*, y), G) &= \{\rho \mid \rho(x) = \rho(y)\} \cup \mathsf{A}((x, e^+, y), G) \\
\mathsf{A}(q \ \mathbf{AND}\ q', G) &= \mathsf{A}(q, G) \bowtie \mathsf{A}(q', G) \\
\mathsf{A}(q \ \mathbf{UNION}\ q', G) &= \mathsf{A}(q, G) \cup \mathsf{A}(q', G) \\
\mathsf{A}(q \ \mathbf{OPT}\ q', G) &= \mathsf{A}(q, G) \bowtie \mathsf{A}(q', G) \cup \mathsf{A}(q, G) \setminus \mathsf{A}(q', G) \\
\mathsf{A}(q(W), G) &= \Pi_W \mathsf{A}(q, G)
\end{aligned}$$

where e^i is the composition of e i times, i.e., $e \circ \ldots \circ e$ and $u \in \mathbf{I}$ and $\rho(u) = u$, i.e., the mapping of a constant is itself. Two triple mappings ρ_1 and ρ_2 are said to be compatible if $\forall x \in dom(\rho_1) \cap dom(\rho_2), \rho_1(x) = \rho_2(x)$. If ρ_1 and ρ_2 are compatible, then $\rho_1 \cup \rho_2$ is also a mapping obtained by extending ρ_1 according to ρ_2 on all the variables in $dom(\rho_2) \setminus dom(\rho_1)$. Given two sets of mappings M_1 and M_2, the join of M_1 and M_2 is defined as: $M_1 \bowtie M_2 = \{\rho_1 \cup \rho_2 \mid \rho_1 \in M_1, \rho_2 \in M_2$ are compatible$\}$, the union $M_1 \cup M_2 = \{\rho \mid \rho \in M_1$ *or* $M_2\}$ *and* difference $M_1 \setminus M_2 = \{\rho_1 \in M_1 \mid \forall \rho_2 \in M_2, \rho_1$ *and* ρ_2 *are not compatible*$\}$. *And finally, the projection operator Π_W selects only those parts of the mappings relevant to variables in W.*

If the answers of a SPARQL query q over a graph G are non empty, i.e., $\mathsf{A}(q, G) \neq \emptyset$, we write q *is satisfiable in* G. The query evaluation problem for cPP and uPP queries is NP-complete. Besides, the evaluation problem is coNP-complete and $\sum_2^p$-complete for oPP without and with projection respectively (Kostylev et al. 2015). Furthermore, it is known that containment of 2RPQs (two-way regular path queries) and Conjunctive 2RPQs without projection is PSPACE-complete (Calvanese et al. 2003), and EXPSPACE-complete if projection is allowed. It has been proved that, these results carry over for SPARQL fragments cPP and oPP with projection. The PSPACE-complete bound is inherited if the right-hand side query is without projection. The problem of subsumption is EXPSPACE-complete for oPP queries (Kostylev et al. 2015). For query evaluation under OWL DL entailment, we obtain:

Theorem 1. *Let G be an OWL EL ontology, $cl(G)$ its closure and P a BGP. A variable mapping ρ is a solution for P over G under OWL DL entailment regime if and only if either ρ is a solution for P over $cl(G)$ under simple entailment, or $cl(G)$ is inconsistent*

From Theorem 1, we have a set of mappings for BGPs, if the query under evaluation contains UNION and/or OPTIONAL patterns, we can apply join, union and difference operations on these mappings to obtain a mapping for the patterns as in Definition 4.

Definition 5. *Given a set of OWL EL axioms $\mathcal{S}$ and SPARQL queries q and q', q is contained in q' under the OWL DL entailment regime, denoted $q \sqsubseteq^{\mathcal{S}}_{\mathrm{DL}} q'$, iff for any OWL EL ontology G which is a model of $\mathcal{S}$, $\mathsf{A}(q, G) \subseteq \mathsf{A}(q', G)$.*

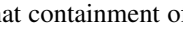
SPARQL queries can be encoded into the μ-calculus.

μ-calculus

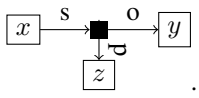
The μ-calculus is a logic obtained by adding fixpoint operators to ordinary modal logic (Kozen 1983). The syntax of the μ-calculus is composed of countable sets of *atomic propositions* AP, a set of *variables* Var, a set of *programs and their respective converses* $\mathrm{Prog} = \{s, p, o, \bar{s}, \bar{p}, \bar{o}\}$ for navigating in graphs. A μ-calculus formula, φ, can be defined inductively as follows:

$$\varphi \ ::= \top \mid q \mid X \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle a \rangle \varphi \mid [a]\varphi \mid \mu X \varphi \mid \nu X \varphi$$

where $q \in AP, X \in Var$ and $a \in Prog$ is a transition program or its converse $\bar{a}$. The greatest and least fixpoint operators (ν and μ) respectively introduce general and finite recursion in graphs. If a μ-calculus formula ψ appears under the scope of a least μ or greatest ν fixed point operator over all the programs $\{s, p, o, \bar{s}, \bar{p}, \bar{o}\}$ as, $\mu X.\psi \vee \langle s \rangle X \vee \langle p \rangle X \vee \cdots$ or $\nu X.\psi \wedge \langle s \rangle X \wedge \langle p \rangle X \wedge \cdots$, then, for legibility, we denote the formulae by $lfp(X, \psi)$ and $gfp(X, \psi)$, respectively. For instance, the μ-calculus encoding of the $q(x) = (x, \mathtt{sp}, \mathrm{transport})$ corresponding to the query that selects all modes of transportation is: $\mathrm{lfp}(X, \langle \bar{s} \rangle x \wedge \langle p \rangle \mathtt{sp} \wedge \langle o \rangle \mathrm{transport})$. μ-calculus formulas that are encodings of SPARQL queries are interpreted over RDF transition systems.

RDF Transition Systems RDF transition systems were first introduced in (Chekol et al. 2012a). They are labelled transition system representations of RDF graphs where two sets of nodes are introduced: one set for each triple (called triple node) and the other set for each subject, predicate, and object of each triple. A triple node is connected to its subject, predicate, and object nodes. For instance, the RDF graph 

can be turned into an RDF transition system



using a function σ as: . Transition from one node to another is done by using a set of transition programs $\{s, p, o\}$ and their converses. An RDF transition system K is considered as a *model* of formula ψ if there exists a node s in the transition system where ψ holds, i.e., $K, s \models \psi$. If a formula has a model, then it is called *satisfiable*.

The μ-calculus with converse lacks functionality or number restrictions. Thus, one cannot impose that each triple node is connected to exactly one node for each subject, predicate, and object node. However, one can impose a lighter restriction to achieve this by taking advantage of the technique introduced in (Genevès and Layaïda 2006) and adopted in (Chekol 2012). Since it is not possible to ensure that there is only one successor, then we restrict all the successors to bear the same constraints. Thus, they become interchangeable. This can be done by rewriting the formulas using a function f such that all occurrences of $\langle a\rangle\varphi$ (existential formulas) are replaced by $\langle a\rangle\top \wedge [a]\varphi$. f is defined inductively on the structure of a μ-calculus formula. Thus, when checking for query containment, we assume that the formulas are rewritten using function f.

Lemma 1. *Let φ be a μ-calculus encoding of a SPARQL query, φ is satisfied by some RDF transition system if and only if $f(\varphi)$ is satisfied by some transition system.*

Containment under Entailment

To clarify the study of SPARQL query containment under entailment regimes, consider the following example.

Example 1. *Containment between q and q' does not hold under the simple entailment. However, containment holds under OWL DL entailment $q \sqsubseteq_{\text{DL}} q'$ because* `eqp` *can be expressed as a two-way* `sp` *relation.*

$$q(y) = (x, \texttt{eqp}, train).(y, x, Berlin)$$
$$q'(y) = (x, \texttt{sp}, train).(train, \texttt{sp}, x).(y, x, Berlin)$$

SPARQL query containment under OWL DL entailment can be determined by rewriting queries using ontology axioms and then reducing the encoding of the rewriting to the validity test in the μ-calculus.

Query Rewriting The rewriting problem for conjunctive queries over $\mathcal{EL}$ ontologies is shown to be ExpTime-hard (Hansen et al. 2015; Bienvenu, Lutz, and Wolter 2012). In addition, it has been shown that query evaluation in OWL QL under OWL DL entailment can be done by rewriting queries using SPARQL 1.1 property paths. Additionally, it is proven that this approach does not work for OWL EL if (1) intersection (`int`) of concepts appears in the subject position, and (2) there is unrestricted use of someValuesFrom (`svf`) in the subject and object positions of an RDF triple (Kazakov, Krötzsch, and Simancik 2014). However, this problem can be avoided by using a more expressive query language or an OBDA-based approach. Consequently, here, we use the μ-calculus in order to rewrite the queries. In fact, we perform the following to rewrite a query: if the query is not rewritable using SPARQL property paths, then we use the μ-calculus, otherwise, property paths are used.

We introduce $\mathbf{spp}$, $\mathbf{somepp}$, $\mathbf{intListp}$, and $\mathbf{scp}$ as abbreviations denoting the following path expressions:

$$\mathbf{spp} = (\texttt{sp} \mid \texttt{eqp} \mid \texttt{eqp}^-),$$
$$\mathbf{somepp} = (\texttt{onp} \circ \texttt{sp} \circ (\texttt{dom} \mid \texttt{range})),$$
$$\mathbf{intListp} = (\texttt{int} \circ \texttt{rdf:rest}^* \circ \texttt{rdf:first}),$$
$$\mathbf{scp} = (\texttt{sc} \mid \texttt{eqc} \mid \texttt{eqc}^- \mid \mathbf{somepp} \mid \mathbf{intListp}).$$

The subproperty relations can be queried by using either the subproperty or equivalent or inverted equivalent relations, this is denoted by $\mathbf{spp}$. $\mathbf{somepp}$ expresses `svf` through path composition using `onp`, `sp`, `dom`, and `range` relations. Furthermore, $\mathbf{scp}$ denotes several ways of querying subclass relations, for instance, using subclass itself, equivalent class, domain, range and so on. Finally, the intersection of concepts can be retrieved using the path expression $\mathbf{intListp}$ by navigating through the RDF collection elements using `rdf:last` and `rdf:first` relations. Beyond this, subclass relations can also be retrieved from the concepts that are defined with `onp` and `svf`. For instance, given the graph $G = \{(c, \texttt{sc}, x)\ (x, onp, r)\ (x, svf, d))\ (d, sc, d')\ (y, onp, r)\ (y, svf, d')\ (y, sc, e)\}$ and the query (z, sc, v), under OWL DL entailment the answer is $\{(d, d'), (c, e)\}$. However, as it is not possible to obtain this by rewriting the query using property paths, we use the μ-calculus formulae $mu((z, sc, v))$ and $mu'((z, sc, v))$ in Definition 6. In the definition, for legibility, we denote UNION by U .

Definition 6. *Given a SPARQL query pattern q, the rewriting of q is obtained inductively using a function τ as follows:*

$$\begin{aligned}
\tau((s, \texttt{sp}, o)) &= (s, \mathbf{spp}^+, o)\\
\tau((s, p, o)) &= (s, x, o)\ \textbf{AND}\ (x, \mathbf{spp}^*, p)\\
\tau((s, \texttt{type}, o)) &= (s, \texttt{type} \circ \mathbf{scp}^*, o)\ \textbf{U}\ ((s, x, y)\ .\\
&\quad (x, \mathbf{spp}^* \circ \texttt{dom} \circ \mathbf{scp}^*, o))\ \textbf{U}\\
&\quad ((y, x, s)\ .\ (x, \mathbf{spp}^* \circ \texttt{range} \circ \mathbf{scp}^*, o))\\
\tau((p, \texttt{dom}, o)) &= (p, \mathbf{spp}^* \circ \texttt{dom} \circ \mathbf{scp}^*, o)\\
\tau((p, \texttt{range}, o)) &= (p, \mathbf{spp}^* \circ \texttt{range} \circ \mathbf{spc}^*, o)\\
\tau((s, x, o)) &= (s, x, o)\ \textit{when } x \textit{ is a variable}\\
\tau((p, \texttt{eqp}, q)) &= (p, \texttt{eqp}^+, q)\ \textbf{U}\ ((p, \texttt{sp}, q).p, \texttt{sp}^-, q))\\
\tau((s, \texttt{eqc}, o)) &= (s, \texttt{eqc}^+, o)\ \textbf{U}\ ((p, \texttt{sc}, q).p, \texttt{sc}^-, q))\\
\tau(X) &= \mathit{lfp}(X, \langle p\rangle\texttt{type} \wedge \langle o\rangle\texttt{alldsjnt}) \wedge\\
&\quad \langle s\rangle(\langle p\rangle\texttt{members} \wedge \langle o\rangle\big(\langle p\rangle\texttt{rdf:rest} \wedge\\
&\quad \langle\bar{s}\rangle\langle\bar{o}\rangle\langle p\rangle\texttt{rdf:first}\big) \wedge \langle\bar{s}\rangle\langle\bar{o}\rangle X\big)\\
\textit{where } X &= (s, \texttt{a}, \texttt{alldsjnt})\\
\tau(s, \texttt{sc}, o) &= (s, \mathbf{scp}^+, o)\ \textbf{U}\ mu[(s, \texttt{sc}, o)]\\
&\quad \textbf{U}\ mu'[(s, \texttt{sc}, o)]\\
mu[(s, \texttt{sc}, o)] &= \langle\bar{s}\rangle s \wedge \mu X.\langle s\rangle(\langle p\rangle\texttt{sc} \wedge \langle o\rangle o) \vee \langle p\rangle(\texttt{sc}\\
&\quad \wedge \langle o\rangle\langle s\rangle([\langle p\rangle(\texttt{onp} \wedge \langle\bar{p}\rangle\langle\bar{s}\rangle X) \wedge\\
&\quad \langle o\rangle\langle\bar{o}\rangle\langle\bar{s}\rangle X] \wedge [\langle p\rangle(\texttt{svf} \wedge \langle\bar{p}\rangle\langle\bar{s}\rangle X) \wedge\\
&\quad \langle o\rangle\langle s\rangle(\langle p\rangle\texttt{sc} \wedge \langle o\rangle\langle\bar{o}\rangle\langle\bar{s}\rangle X)]))\\
mu'[(s, \texttt{sc}, o)] &= \langle\bar{s}\rangle s \wedge \mu X.\langle s\rangle(\langle p\rangle\texttt{sc} \wedge \langle o\rangle o) \vee \langle p\rangle(\texttt{sc}\\
&\quad \wedge \langle o\rangle\langle s\rangle([\langle p\rangle(\texttt{svf} \wedge \langle\bar{p}\rangle\langle\bar{s}\rangle X) \wedge\\
&\quad \langle o\rangle\langle\bar{o}\rangle\langle\bar{s}\rangle X] \wedge [\langle p\rangle(\texttt{onp} \wedge \langle\bar{p}\rangle\langle\bar{s}\rangle X) \wedge\\
&\quad \langle o\rangle\langle s\rangle(\langle p\rangle\texttt{sp} \wedge \langle o\rangle\langle\bar{o}\rangle\langle\bar{s}\rangle X)]))\\
\tau(q\ \textbf{AND}\ q') &= \tau(q)\ \textbf{AND}\ \tau(q')\\
\tau(q\ \textbf{U}\ q') &= \tau(q)\ \textbf{U}\ \tau(q')\\
\tau(q\ \textbf{OPT}\ q') &= \tau(q)\ \textbf{OPT}\ \tau(q')
\end{aligned}$$

For AND-UNION query patterns, it is proved that the rewriting without the μ-calculus is correct and the data complexity

of query answering is NLOGSPACE (Bischof et al. 2014). In addition, the μ-calculus rewriting is in PTIME by a reduction into model checking in the μ-calculus (Kozen 1983). It is possible to extend these proofs to the more complex OPTIONAL patterns. Next, we show how these query rewritings can be encoded into the μ-calculus. If the rewriting is already a μ-calculus formula, then its encoding is itself. Otherwise, we proceed as discussed below.

Encoding SPARQL queries into the μ-calculus The principle of the translation is that each triple is associated with a sub-formula ($\langle\bar{s}\rangle subject \wedge \langle p\rangle predicate \wedge \langle o\rangle object$) stating the existence of the triple somewhere in a transition system. Hence, it is quantified by μ (least fixed point) so as to propagate the sub-formula to the entire transition system. μ encodes a reflexive transitive closure over all the programs and is denoted by $lfp(X, \text{sub-formula})$ (Chekol et al. 2012a). Given a containment problem $q \sqsubseteq q'$, the encoding is as follows: the variables, constants, and blank nodes of the left-hand side query q are encoded using atomic propositions of the μ-calculus. Basically, the variables, constants and blank nodes are frozen, i.e., equivalent to obtaining a canonical instance of the query. Whereas the blank nodes on the right-hand side query q' are treated as existential variables and are encoded by instantiating them in all possible ways with the elements of q (i.e., constants, blank nodes and variables) that appear on the subject and object positions since blank nodes do not appear on the predicate positions. The variables and constants of q' are encoded into atomic propositions in the μ-calculus. Afterwards, a recursive function E is used to inductively construct a formula.

Definition 7. *The μ-calculus encoding of the containment test between SPARQL queries q and q' can be obtained as:*

$$\Phi_{q\sqsubseteq q'} = \mathsf{E}(q) \wedge \neg\Big(\bigvee_{i}^{|el(q)|}\bigvee_{j}^{|bn(q')|} \mathsf{E}_{b_j\mapsto l_i}(q')\Big)$$

$$\begin{array}{ll}
\mathsf{E}((x,y,z)) & = lfp\big(X, \langle\bar{s}\rangle x \wedge \langle p\rangle y \wedge \langle o\rangle z\big)\\
\mathsf{E}((x,e,z)) & = lfp\big(X, \langle\bar{s}\rangle x \wedge \mathsf{RE}(e,z)\big)\\
\mathsf{E}((x,e^-,y)) & = \mathsf{E}((y,e,x))\\
\mathsf{E}((x,e^- \circ e_1,y)) & = \mathsf{E}((x,e^-,y)) \vee \mathsf{E}((x,e_1,y))\\
\mathsf{E}((x,e \circ e_1^-,y)) & = \mathsf{E}((x,e_1,y) \vee \mathsf{E}((x,e_1^-,y))\\
\mathsf{E}((x,e^- \circ e_1^-,y)) & = \mathsf{E}((x,e^-,y)) \vee \mathsf{E}((x,e_1^-,y))\\
\mathsf{E}((x,e^- \circ e_1,y)) & = \langle o\rangle x \wedge \mathsf{RE}(e^- \circ e_1,y)\\
\mathsf{E}((x,e \circ e_1^-,y)) & = \langle\bar{s}\rangle x \wedge \mathsf{RE}(e,\langle\bar{o}\rangle\mathsf{RE}(e_1^-,y))\\
\mathsf{E}((x,e^- \circ e_1^-,y)) & = \langle o\rangle x \wedge \mathsf{RE}(e^-,\langle\bar{o}\rangle\mathsf{RE}(e_1^-,y))\\
\mathsf{E}((x,(e^-)^+,y)) & = \langle o\rangle x \wedge \mathsf{RE}((e^-)^+,y)\\
\mathsf{E}((x,(e^-)^*,y)) & = \langle o\rangle x \wedge \mathsf{RE}((e^-)^*,y)\\
\mathsf{E}((x,(e^-)?,y)) & = \langle o\rangle x \wedge \mathsf{RE}((e^-)?,y)\\
\mathsf{E}(q \textbf{ AND } q') & = \mathsf{E}(q) \wedge \mathsf{E}(q')\\
\mathsf{E}(q \textbf{ UNION } q') & = \mathsf{E}(q) \vee \mathsf{E}(q')
\end{array}$$

where $el(q)$ denotes a set containing all constants, blank nodes, and variables that appear in the subject and object positions of q, $bn(q')$ denotes a set containing all the blank nodes in q', and $b_j \mapsto l_i$ denotes the jth blank node $b_j \in bn(q')$ is replaced by $l_i \in el(q)$. Further, in lfp, x and z are atomic propositions and property paths that appear in the query are encoded into atomic propositions using the function RE*. This function takes two arguments (the predicate which is a path expression and the object of a triple).*

$$\begin{array}{ll}
\mathsf{RE}(I,y) & = \langle p\rangle I \wedge \langle o\rangle y\\
\mathsf{RE}(e \circ e',y) & = (\mathsf{RE}(e,y) \vee \mathsf{RE}(e',y))\\
\mathsf{RE}(e \circ e',y) & = \mathsf{RE}(e,\langle s\rangle\mathsf{RE}(e',y))\\
\mathsf{RE}(e?,y) & = \mathsf{RE}(e,y) \vee \langle\bar{s}\rangle y\\
\mathsf{RE}(e^+,y) & = \mu X.\mathsf{RE}(e,y) \vee \mathsf{RE}(e,\langle s\rangle X)\\
\mathsf{RE}(e^*,y) & = \mathsf{RE}(e^+,y) \vee \langle\bar{s}\rangle y\\
\mathsf{RE}(I^-,y) & = \langle p\rangle I \wedge \langle\bar{s}\rangle y\\
\mathsf{RE}((e^-)^+,y) & = \mu X.\mathsf{RE}(e^-,y) \vee \mathsf{RE}(e^-,\langle\bar{o}\rangle X)\\
\mathsf{RE}((e^-)^*,y) & = \mathsf{RE}((e^-)^+,y) \vee \langle o\rangle y\\
\mathsf{RE}((e^-)?,y) & = \mathsf{RE}(e^-,y) \vee \langle o\rangle y\\
\mathsf{RE}(e^- \circ e_1,y) & = \mathsf{RE}(e^-,\langle s\rangle\mathsf{RE}(e,y))\\
\mathsf{RE}(e \circ e_1^-,y) & = \mathsf{RE}(e,\langle\bar{o}\rangle\mathsf{RE}(e_1^-,y))\\
\mathsf{RE}(e^- \circ e_1^-,y) & = \mathsf{RE}(e^-,\langle\bar{o}\rangle\mathsf{RE}(e_1^-,y))
\end{array}$$

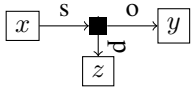
Lemma 2. *Let $q \in \{\mathsf{cPP}, \mathsf{uPP}\}$, for every RDF transition system K whose associated OWL EL ontology is G, we have that q is satisfiable in G iff $\mathsf{E}(q)$ is satisfiable in K.*

Encoding OPTIONAL Pattern (oPP) Queries We assume that the query patterns are in OPT *normal form*, i.e., an OPT query pattern containing only the operators AND and OPT is in OPT normal form if the OPT operator never occurs in the scope of an AND operator. It was shown that every well-designed graph pattern can be transformed into OPT normal form in polynomial time (Letelier et al. 2013). To translate OPTIONAL pattern queries into the μ-calculus, we extend the function E as shown below.

Definition 8. *Given two queries $q, q' \in \{\mathsf{oPP}\}$, the μ-calculus encoding of the containment test is obtained by extending E as: $\mathsf{E}(q \textbf{ OPT } q') = \mathsf{E}(q) \vee \mathsf{E}(q \textbf{ AND } q')$.*

Lemma 3. *Let $q \in \mathsf{oPP}$, for every RDF transition system K whose associated OWL EL ontology is G, we have that q is satisfiable in G iff $\mathsf{E}(q)$ is satisfiable in K.*

Proof. (sketch) ($\Rightarrow$) assume that q is satisfiable in G and K is an RDF transition system associated to G. To show that, $\mathsf{E}(q)$ is satisfiable in K, we proceed inductively on the construction of the formula. The base case is when $q(x) = (x,z,y) \textbf{ OPT } (x,r,s)$ is an OPT pattern query. Consider that q is satisfiable in its two canonical instance graph $G = \{(x,z,y)\}$ and $G' = \{(x,z,y),(x,r,s)\}$ and $\mathsf{E}(q)$ is satisfiable on their associated RDF transition graphs



$K =$ and K' respectively. The encoding of q is $\mathsf{E}(q) = \mathsf{E}((x,y,z)) \vee \mathsf{E}((x,y,z) \textbf{ AND } (x,r,s))$. $\mathsf{E}(q) = lfp\big(X, \langle\bar{s}\rangle x \wedge \langle p\rangle y \wedge \langle o\rangle z\big) \vee (lfp\big(Y, \langle\bar{s}\rangle x \wedge \langle p\rangle y \wedge \langle o\rangle z\big) \wedge lfp\big(Z, \langle\bar{s}\rangle x \wedge \langle p\rangle r \wedge \langle o\rangle s\big))$, in this encoding the variables represent atomic propositions. These formulae are satisfied in both of the transition systems, i.e., $K \models \mathsf{E}(q)$ and $K' \models \mathsf{E}(q)$. In fact, we can encode each variable with $\top$ so that it can be instantiated in every node of the transition system. The formula obtained in this way is satisfiable in K and in every transition system that is associated with every satisfying graph of q. The semantics of the OPT operator is captured by the fact that $\mathsf{E}(q)$ is satisfied in K. This is due to the first disjunct in $\mathsf{E}(q)$ being true in K whereas the

second disjunct does not hold. For the inductive case, one can continue similarly by taking care of the query pattern connectives AND and OPT.

($\Leftarrow$) this direction can be shown by working inductively on the construction of the formula and the structure of its models which are RDF transition systems due to restrictions imposed on the formula (cf. preliminaries section). □

With Lemma 3 we can check the containment of OPT queries. Next, we present the encoding of OWL EL axioms in the μ-calculus.

Encoding OWL EL Schema Given a set of OWL EL axioms $\mathcal{S} = \{s_1, \ldots, s_n\}$, the μ-calculus encoding of $\mathcal{S}$ is obtained using the function Θ as follows:

$$\begin{aligned}
\Theta(\mathbf{I}) &= \mathbf{I}\\
\Theta(\mathcal{S}) &= \Theta(s_1) \wedge \cdots \wedge \Theta(s_n)\\
\Theta((\mathbf{C}, \mathtt{sc}, \mathbf{D})) &= \mathit{gfp}\big(X, \Theta(\mathbf{C}) \Rightarrow \Theta(\mathbf{D})\big)\\
\Theta((\mathbf{C}, \mathtt{eqc}, \mathbf{D})) &= \mathit{gfp}\big(X, (\Theta(\mathbf{C}) \Rightarrow \Theta(\mathbf{D})) \wedge\\
&\qquad (\Theta(\mathbf{D}) \Rightarrow \Theta(\mathbf{C}))\big)\\
\Theta((b, \mathtt{int}, (C_1,&\\
\ldots, C_n)) &= \langle s\rangle(\langle p\rangle \mathtt{int} \wedge\\
&\qquad \langle o\rangle(\Theta(C_1) \wedge \cdots \wedge \Theta(C_n))\\
\Theta((b, \mathtt{onp}, \mathbf{P}),&\\
(b, \mathtt{svf}, \mathbf{C})) &= \langle s\rangle\big(\langle p\rangle \mathbf{P} \wedge \langle o\rangle(\langle s\rangle\langle o\rangle\Theta(\mathbf{C}))\big)\\
\Theta((\mathbf{P}, \mathtt{sp}, \mathbf{Q})) &= \mathit{gfp}\big(X, \mathbf{P} \Rightarrow \mathbf{Q}\big)\\
\Theta((\mathbf{P}, \mathtt{eqp}, \mathbf{Q})) &= \mathit{gfp}\big(X, (\mathbf{P} \Rightarrow \mathbf{Q}) \wedge (\mathbf{Q} \Rightarrow \mathbf{R})\big)\\
\Theta((\mathbf{P}, \mathtt{domain}, \mathbf{C})) &= \langle s\rangle\big(\langle p\rangle \mathbf{P} \Rightarrow \langle p\rangle \mathtt{a} \wedge \langle o\rangle \mathbf{C}\big)\\
\Theta((\mathbf{P}, \mathtt{range}, \mathbf{C})) &= \langle \bar{o}\rangle\langle p\rangle \mathbf{P} \Rightarrow \langle s\rangle(\langle p\rangle \mathtt{a} \wedge \langle o\rangle \mathbf{C})
\end{aligned}$$

where $X = (b, \mathtt{a}, \mathtt{alldsjnt}), (b, \mathtt{members}, (C_1, \ldots, C_n))$ and in $\Theta(\mathbf{C})$ and $\Theta(\mathbf{D})$, if $\mathbf{C}$ and $\mathbf{D}$ are blank nodes $b \in \mathbf{B}$, then we check if b is contained in the triples $\{(b, \mathtt{int}, (C_1, \ldots, C_n)\}$ or $\{(b, \mathtt{onp}, \mathbf{P}), (b, \mathtt{svf}, \mathbf{C})\}$ and construct the formula accordingly. Blank nodes and IRIs, in Θ, are encoded as atomic propositions in the μ-calculus.

Lemma 4. *Given a set of OWL EL axioms $\mathcal{S}$, $\mathcal{S}$ has a model iff $\Theta(\mathcal{S})$ is satisfiable.*

Proof. (Sketch) The proof directly follows from that of Lemma 1 of (Chekol 2012). □

So far, we have presented the rewriting of SPARQL queries using property paths, the encoding of queries and OWL EL axioms in the μ-calculus. Thus, we can reduce query containment under OWL DL entailment regime into validity test in the μ-calculus as: $q \sqsubseteq^{\mathcal{S}}_{\mathrm{DL}} q' \Leftrightarrow \tau(q) \sqsubseteq^{\mathcal{S}} \tau(q') \Leftrightarrow \Phi_{\tau(q)\sqsubseteq^{\mathcal{S}}\tau(q')} \wedge \Theta(\mathcal{S})$. We abbreviate $\Phi_{\tau(q)\sqsubseteq^{\mathcal{S}}\tau(q')} \wedge \Theta(\mathcal{S})$ by $\Phi(S, q, q')$.

Theorem 2. *Given two SPARQL queries q, $q' \in \{\mathsf{cPP}, \mathsf{uPP}\}$, and a set of OWL EL axioms $\mathcal{S}$, $q \sqsubseteq^{\mathcal{S}}_{\mathrm{DL}} q$ iff $\Phi(S, q, q')$ is unsatisfiable.*

Proof. ($\Rightarrow$) We show the contrapositive: if $q \not\sqsubseteq^{\mathcal{S}}_{\mathrm{DL}} q'$, then $\Phi(\mathcal{S}, q, q')$ is satisfiable. One can verify that every model G of $\mathcal{S}$ in which there is at least one tuple satisfying q but not q' can be turned into an RDF transition system model for $\Phi(\mathcal{S}, q, q')$. To do so, consider a graph G which is a model of $\mathcal{S}$. Assume also that there is a tuple $\overrightarrow{a}$ in the answers of q over G but not in the answers of q'. We can construct an RDF transition system K from G (as done in (Chekol 2012)). From Lemma 4, we obtain that $\Theta(\mathcal{S})$ is satisfiable in K. At this point, it remains to verify that while $\mathsf{E}(q)$ is satisfiable in K, $\phi = \left(\bigvee_i^{|el(q)|} \bigvee_j^{|bn(q')|} \mathsf{E}_{b_j \mapsto l_i}(q')\right)$ is not. To do so, we build the formulas $\mathsf{E}(q)$ and ϕ by first skolemizing the distinguished variables using the answer tuple $\overrightarrow{a}$. Consequently, from Lemma 2 one obtains $\mathsf{E}(q)$ is satisfiable. However, ϕ is unsatisfiable, this is because the atomic propositions in the formula corresponding to the constants, blank nodes and variables that do not appear in the SELECT clause are not satisfied in K. This is justified by the fact that if a formula φ is satisfiable in an RDF transition system, then its negation $\neg\varphi$ is unsatisfiable. So far we have: $\Theta(\mathcal{S})$, $\mathsf{E}(q)$, and $\neg\phi$ are satisfiable in K. Thus, $\Phi(\mathcal{S}, q, q')$ is satisfiable in K. Without loss of generality, we get that $\Phi(\mathcal{S}, q, q')$ is satisfiable.

($\Leftarrow$) $\Phi(\mathcal{S}, q, q')$ implies that there exists a transition system where the formula $\Phi(\mathcal{S}, q, q')$ holds. Consequently, K is an RDF transition system due to the restriction imposed in Lemma 1. From K it is possible to construct a model G so that we can utilize Lemma 4 to verify that indeed G is a model of $\mathcal{S}$. Thus, it remains to show that the answers of q are not included in the answers of q' over G. From our assumption, we have that $\mathsf{E}(q) \wedge \neg\phi$ is satisfiable in K. From this, we obtain that $\mathsf{E}(q)$ is satisfiable while ϕ is not. It is possible to build an OWL EL ontology from the model $\mathcal{I}$ using a function that uses assertions to form triples. Thus, we have that the answers of q over G are not empty but q' is empty because G contains all those triples that satisfy q and not q'. Therefore, we get that the answers of q are not contained in that of q'. □

The size of the encoding for checking the containment of $\{\mathsf{cPP}, \mathsf{uPP}\}$ queries in the μ-calculus is exponential and the satisfiability test of a μ-calculus formula can be done in an exponential amount of time. Therefore, we get a double exponential upper bound as shown below.

Proposition 1. *Given two SPARQL queries q, $q' \in \{\mathsf{cPP}, \mathsf{uPP}\}$ and a set of OWL EL axioms $\mathcal{S}$, containment under OWL DL entailment can be solved in double exponential amount of time in the size of the encoding.*

We prove the soundness of the containment of oPP queries.

Theorem 3. *Given two SPARQL queries $q, q' \in \mathsf{oPP}$ and a set of OWL EL axioms $\mathcal{S}$, q is contained in q' under OWL DL entailment, $q \sqsubseteq^{\mathcal{S}}_{\mathrm{DL}} q'$ iff $\Phi(S, q, q')$ is unsatisfiable.*

Proposition 2. *Given two SPARQL queries $q, q' \in \mathsf{oPP}$, and a set of OWL EL axioms $\mathcal{S}$, containment under OWL DL entailment can be solved in triple exponential amount of time in the size of the encoding.*

This result follows from the fact that the size of the encoding of oPP queries is double exponential if q' does not contain blank nodes.

Conclusion

In this work, we have shown that query containment over a set of OWL EL axioms under the OWL DL entailment can be reduced to validity test in the μ-calculus by rewriting queries using property paths and the μ-calculus. We have provided a double exponential upper bound for containment of cPP and uPP queries. Whereas there is a further jump in complexity if the queries are oPP. In both cases, if the right hand side query does not have blank nodes, there is an exponential drop in complexity. Note that SPARQL query containment in the presence of blank nodes coincides with that of having non-distinguished variables (i.e., union of conjunctive query containment). We plan to extend the implementation in (Chekol et al. 2013) by writing a parser that performs query rewriting, and designing a benchmark. Even though, there are no other systems that we can compare it too, we will carry out experiments to evaluate its performance and to see how well it copes with the size of the formulas obtained from query rewritings. We will apply the technique introduced in (Genevès and Schmitt 2015) in order to reduce the size of the formula by upto exponential.

Furthermore, our approach is very flexible, it can directly be applied to OWL QL without inverse roles. It can also easily be extended to deal with OWL RL without cardinality restrictions. Besides, without additional modification we can use our encoding to determine containment of queries of the form BGP MINUS BGP$'$. It is possible to prove an exponential upper bound for this problem when the right-hand side query has no blank nodes.

References

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the EL envelope. In *IJCAI*, volume 5, 364–369.

Baader, F.; Brandt, S.; and Lutz, C. 2008. Pushing the EL envelope further. In *Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*.

Bienvenu, M.; Lutz, C.; and Wolter, F. 2012. Deciding FO-rewritability in EL. *Description Logics* 70–80.

Bischof, S.; Krötzsch, M.; Polleres, A.; and Rudolph, S. 2014. Schema-agnostic query rewriting in SPARQL 1.1. In *ISWC*, 584–600.

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Vardi, M. Y. 2000. Containment of Conjunctive Regular Path Queries with Inverse. In *Proc. 7th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2000)*, 176–185.

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Vardi, M. Y. 2003. Reasoning on regular path queries. *SIGMOD Record* 32(4):83–92.

Calvanese, D.; Giacomo, G. D.; and Lenzerini, M. 2008. Conjunctive query containment and answering under description logic constraints. *ACM Transactions on Computational Logic (TOCL)* 9(3):22.

Chekol, M. W.; Euzenat, J.; Genevès, P.; and Layaïda, N. 2012a. SPARQL query containment under RDFS entailment regime. In *Proc. IJCAR*, 134–148.

Chekol, M. W.; Euzenat, J.; Genevès, P.; and Layaïda, N. 2012b. SPARQL query containment under SHI axioms. In *Proc. AAAI*, 10–16.

Chekol, M. W.; Euzenat, J.; Genevès, P.; and Layaïda, N. 2013. Evaluating and Benchmarking SPARQL Query Containment Solvers. In *International Semantic Web Conference (2)*, 408–423.

Chekol, M. W. 2012. *Static Analysis of Semantic Web Queries*. Ph.D. Dissertation, Université de Grenoble.

Genevès, P., and Layaïda, N. 2006. A system for the static analysis of XPath. *ACM Trans. Inf. Syst.* 24(4):475–502.

Genevès, P., and Schmitt, A. 2015. Expressive logical combinators for free. In *IJCAI*, 311–317.

Genevès, P.; Layaïda, N.; and Schmitt, A. 2007. Efficient static analysis of XML paths and types. In *Proc. PLDI*, 342–351. ACM.

Glimm, B., and Krötzsch, M. 2010. SPARQL beyond subgraph matching. *The Semantic Web–ISWC 2010* 241–256.

Glimm, B. 2011. Using SPARQL with RDFS and OWL entailment. *Reasoning Web. Semantic Technologies for the Web of Data* 137–201.

Hansen, P.; Lutz, C.; Seylan, I.; and Wolter, F. 2015. Efficient query rewriting in the description logic EL and beyond. In *IJCAI*, 3034–3040.

Ioannidis, Y. E., and Ramakrishnan, R. 1995. Containment of conjunctive queries: Beyond relations as sets. *ACM Transactions on Database Systems (TODS)* 20(3):288–324.

Kazakov, Y.; Krötzsch, M.; and Simancik, F. 2014. The incredible ELK - from polynomial procedures to efficient reasoning with EL ontologies. *J. Autom. Reasoning* 53(1):1–61.

Kostylev, E. V.; Reutter, J. L.; Romero, M.; ; and Vrgoč, D. 2015. Sparql with property paths. In *International Semantic Web Conference*.

Kozen, D. 1983. Results on the propositional μ-calculus. *Theoretical computer science* 27(3):333–354.

Krötzsch, M., and Rudolph, S. 2007. Conjunctive queries for EL with composition of roles. In *Proceedings of the 2007 International Workshop on Description Logics (DL2007), Brixen-Bressanone, Italy, 8-10 June, 2007*.

Krötzsch, M. 2011. Efficient rule-based inferencing for owl el. In *IJCAI*, volume 11, 2668–2673.

Letelier, A.; Pérez, J.; Pichler, R.; and Skritek, S. 2013. Static analysis and optimization of semantic web queries. *ACM Transactions on Database Systems (TODS)* 38(4):25.

Pérez, J.; Arenas, M.; and Gutierrez, C. 2009. Semantics and complexity of sparql. *ACM Transactions on Database Systems (TODS)* 34(3):16.

Pichler, R., and Skritek, S. 2014. Containment and Equivalence of Well-designed SPARQL. 39–50.

Rosati, R. 2007. On conjunctive query answering in el. In *20th International Workshop on Description Logics DL07*. Citeseer.