SPARQL Query Containment under SHI Axioms

Melisachew Wudagae Chekol, Jérôme Euzenat, Pierre Genevès, Nabil Layaïda

SPARQL Query Containment under $\mathcal{SHI}$ Axioms

Melisachew Wudage Chekol, Jérôme Euzenat, Pierre Genevès, Nabil Layaïda

RESEARCH REPORT

N° 7943

April 2012

Project-Teams EXMO and WAM

ISSN 0249-6399   ISRN INRIA/RR--7943--FR+ENG

SPARQL Query Containment under $\mathcal{SHI}$ Axioms

Melisachew Wudage Chekol*, Jérôme Euzenat*, Pierre Genevès†, Nabil Layaïda*

Project-Teams EXMO and WAM

Research Report n° 7943 — April 2012 — 23 pages

Abstract: SPARQL query containment under schema axioms is the problem of determining whether, for any RDF graph satisfying a given set of schema axioms, the answers to a query are contained in the answers of another query. This problem has major applications for verification and optimization of queries. In order to solve it, we rely on the μ-calculus. Firstly, we provide a mapping from RDF graphs into transition systems. Secondly, SPARQL queries and RDFS and $\mathcal{SHI}$ axioms are encoded into μ-calculus formulas. This allows us to reduce query containment and equivalence to satisfiability in the μ-calculus. Finally, we prove a double exponential upper bound for containment under $\mathcal{SHI}$ schema axioms.

Key-words: Query containment, SPARQL, RDF, ontologies

* INRIA
† CNRS

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Inclusion de requêtes sur les axiom $\mathcal{SHI}$

Résumé :

Mots-clés : inclusion de requêtes, SPARQL, ontologies, RDF

1 Introduction

Access to semantic web data expressed in RDF (Resource Description Framework) may be achieved through querying. Currently, querying RDF graphs is done mainly with the SPARQL query language. It has been a source of research from various perspectives, in particular for extending the language and optimizing queries. Querying RDF graphs with SPARQL proceeds by matching graph patterns, i.e., triple patterns connected to form graphs by means of joins expressed using several occurrences of the same variable. Since queries in the semantic web are evaluated over huge RDF graphs, optimizations are necessary in order to find minimal queries to reduce the computational cost of query evaluation. Query containment has been a central point of discussion in many database and knowledge base applications, including data warehousing, data integration and query optimization (see [7] for details).

Query optimization aims at improving the runtime performance of query evaluation. Studies have contributed to query optimization using rewriting rules in particular in the relational algebra for databases [20, 9]. Similar approaches have also been applied to SPARQL [31]. These works, however, need at some point to prove the correctness of query optimization, i.e., the semantics of the optimized query is the same as the original one. In other words, the results of a given query are exactly the same as the optimized one regardless of the considered database. This can be reduced to query containment. Thus, query containment plays a vital role in optimization [20, 9]. It can be defined as determining if the result of one query is included in the result of another one for any RDF graph. In addition, query containment can be of independent interest for performing other optimizations. For example, if a query q is contained in q', then q can be evaluated on the materialized view of q' rather than on the whole data graph. To the best of our knowledge, the problem of SPARQL query containment (under a schema) has not been covered in the literature.

The aim of this paper is to address SPARQL query containment under a DL schema (the schema is formulated within the fragments of $\mathcal{SHIQ}$). We apply an approach which has already been successfully applied for XPath [16]. SPARQL is interpreted over graphs, hence we encode it in a graph logic, specifically the alternation-free fragment of the μ-calculus [24] with converse and nominals [32] interpreted over labeled transition systems. We show that this logic is powerful enough to deal with query containment for the fragment of SPARQL considered here in the presence of RDFS and $\mathcal{SHI}$ (schema) axioms. Furthermore, this logic admits exponential time decision procedures that is implemented in practice [32, 33, 16]. Hence, our approach opens a way to use this implementation.

We introduce a translation of RDF graphs into transition systems and SPARQL queries and schema axioms into μ-calculus formulae. Then, we show how query containment in SPARQL can be reduced to unsatisfiability in the μ-calculus. We prove a double exponential upper bound for the problem. An additional benefit of using a μ-calculus encoding is to take advantage of fixpoints and modalities for encoding recursion. They allow to deal with natural extensions of SPARQL such as path queries [2] or queries modulo RDF Schema.

Outline: after presenting RDF(S), $\mathcal{SHI}$ and SPARQL (§2), we show how to translate RDF graphs into transition systems (§3) and SPARQL queries and schema axioms into μ-calculus formulas (§4.1 and §4.2). Therefore, query containment in SPARQL under schema axioms can be reduced to unsatisfiability test in μ-calculus (§4.3). Finally, we present the complexity of the problem (§5) and the related works (§6) along with a summary of concluding remarks (§7).

2 Preliminaries

This section introduces the basics of RDF, RDF Schema, the description logic $\mathcal{SHI}$, and SPARQL.

2.1 RDF

is a language used to express structured information on the Web as graphs. We present a compact formalization of RDF [18]. Let U, B, and L be three disjoint infinite sets denoting the set of URIs (identifying a resource), blank nodes (denoting an unidentified resource) and literals (a character string or some other type of data) respectively. We abbreviate any union of these sets as for instance, UBL $= U \cup B \cup L$. A triple of the form $(s, p, o) \in$ UB $\times U \times$ UBL is called an *RDF triple*. s is the *subject*, p is the *predicate*, and o is the *object* of the triple. Each triple can be thought of as an edge between the subject and the object labelled by the predicate, hence a set of RDF triples is often referred to as an *RDF graph*. RDF has a model theoretic semantics [18].

Example 1 (RDF Graph)**.** *Consider 8 triples of an RDF graph about writers and their works (all identifiers correspond to URIs, _:b is a blank node):*

G	$=$	*{(Poe,wrote,thegoldbug),*	*(Baudelaire,translated,thegoldbug),*
		(Poe, wrote, theraven),	*(Mallarmé, translated, theraven),*
		(theraven, type, Poem),	*(Mallarmé, wrote, _:b),*
		(_:b, type, Poem),	*(thegoldbug, type, Novel) }*

RDFS (RDF Schema) may be considered as a simple ontology language expressing subsumption relations between classes or properties [18]. Technically, this is an RDF vocabulary used for expressing axioms constraining the interpretation of graphs. The RDFS vocabulary and its semantics are given in [18]. We consider a core fragment of RDFS called ρdf [26] which contains the minimal vocabulary, ρdf $= \{\texttt{sp,sc,type,dom,range}\}$, where `sp` denotes the *subproperty* relation, `sc` is *subclass*, and `dom` stands for *domain*. This fragment was proven to be minimal and well-behaved in [26]. Its semantics corresponds to that of full RDFS.

RDFS Axioms: from the RDFS inference rules [18], one can identify the following axioms:

- **Subclass**: constructs the subsumption relation between concepts, $<C, \texttt{sc}, D>$.
- **Subproperty**: constructs the subsumption relation between properties, $<p, \texttt{sp}, q>$.
- **Domain and Range**: restrict the values of properties (i.e., they assign type to properties). Hence, they are typing axioms, $<p, \texttt{dom}, C>$, and $<p, \texttt{range}, C>$.
- **Transitivity**: the properties `sc` and `sp` are transitive. We denote these axioms by $trans(\texttt{sc})$ and $trans(\texttt{sp})$.

RDFS is not an expressive schema language. For instance, it lacks negation and inverse roles. Hence, the next section introduces the description logic $\mathcal{SHI}$ which encompasses the DL fragment of RDFS [4, 14].

2.2 $\mathcal{SHI}$

Description logics are fragments of first-order logic that model a domain of interest in terms of concepts and roles [5]. For this study, we consider the description logic $\mathcal{SHI}$ which is a fragment of

$\mathcal{SHIQ}$ [19]. $\mathcal{ALC}$ ($\mathcal{A}$ttributive $\mathcal{L}$anguage with $\mathcal{C}$omplements) is a fragment of $\mathcal{SHIQ}$ without role hierarchies, transitive roles, inverse roles, and number restrictions. The satisfiability of $\mathcal{SHIQ}$ logic is proved to be ExpTime. We assume standard notation for the syntax and semantics of $\mathcal{SHI}$ knowledge bases.

Syntax In $\mathcal{SHI}$, concepts and roles are formed according to the following syntax:

$$\begin{aligned} C &::= \bot \mid \top \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists r.C \mid \forall r.C \\ r &::= p \mid p^- \end{aligned}$$

Additionally, the following holds:

$$\begin{aligned} C_1 \sqcup C_2 &= \neg(\neg C_1 \sqcap \neg C_2) \\ \top &= \neg(\bot) \\ \forall r.C &= \neg(\exists r.\neg C) \end{aligned}$$

p denotes an atomic role, $\bot$ represents an empty concept, A denotes an atomic concept, C denotes a complex concept, and r denotes a complex role which is an atomic role or its inverse.

$\mathcal{SHI}$ Axioms: The TBox is a finite set of axioms consisting of *concept inclusions* and *role inclusion axioms*:

$$C_1 \sqsubseteq C_2 \quad , \quad r_1 \sqsubseteq r_2 \quad \text{and} \quad trans(r)$$

A DL fragment of RDFS is a subset of $\mathcal{SHI}$. Hence RDFS axioms can be translated into $\mathcal{SHI}$ axioms as shown in Table 1.

	RDFS	$\mathcal{SHI}$
Subclass	$(C_1, \mathtt{sc}, C_2)$	$C_1 \sqsubseteq C_2$
Subproperty	$(r_1, \mathtt{sp}, r_2)$	$r_1 \sqsubseteq r_2$
Domain	$(r, \mathtt{dom}, C)$	$\exists r.\top \sqsubseteq C$
Range	$(r, \mathtt{range}, C)$	$\exists r^-.\top \sqsubseteq C$
Transitivity	$trans(\mathtt{sc})$	$trans(\mathtt{sc})$
Transitivity	$trans(\mathtt{sp})$	$trans(\mathtt{sp})$

Table 1: RDFS into $\mathcal{SHI}$

Semantics An interpretation, $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, consists of a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that assigns to each object name o an element $o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, to each concept A a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ of the domain, and to each role name r a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ over the domain. The role and concept constructs can be interpreted in $\mathcal{I}$ as follows:

$$\begin{aligned}
(\bot)^{\mathcal{I}} &= \emptyset \\
(A)^{\mathcal{I}} &\subseteq \Delta \\
(r)^{\mathcal{I}} &\subseteq \Delta \times \Delta \\
(\neg C)^{\mathcal{I}} &= \Delta \setminus C^{\mathcal{I}} \\
(C_1 \sqcap C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(\exists r.C)^{\mathcal{I}} &= \{x \mid \exists y.(x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \\
(\forall r.C)^{\mathcal{I}} &= \{x \mid \forall y.(x,y) \in r^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\} \\
(r^{-})^{\mathcal{I}} &=\{(x,y) \mid (y,x) \in r^{\mathcal{I}}\}
\end{aligned}$$

Next we define of the union of conjunctive queries (UCQs) in $\mathcal{SHI}$.

Definition 1 (Union of conjunctive queries (UCQs)). *An $\mathcal{SHI}$ conjunctive query is a finite set of atoms of the form $C(v)$ and $R(v, v')$ where v, v' are variables, and C and R denote a concept and a role (possibly inverse) respectively. A union of conjunctive query is a disjunction of conjunctive queries.*

Definition 2 (Answers to UCQs). *Let $\mathcal{A}$ be an ABox, $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ a model of $\mathcal{A}$, q a conjunctive query and $\tau : var(q) \to \Delta^{\mathcal{I}}$ such that:*

$$\begin{aligned}
\mathcal{I} \models^{\tau} C(v) &\quad \textit{if} \;\; \tau(v) \in C^{\mathcal{I}} \\
\mathcal{I} \models^{\tau} R(v,v') &\quad \textit{if} \;\; (\tau(v), \tau(v')) \in R^{\mathcal{I}}
\end{aligned}$$

If for all atoms at, $\mathcal{I} \models^{\tau}$ at, then $\mathcal{I} \models q$.

Definition 3 (Query Entailment). *Query entailment is the decision problem associated with query answering i.e., given a knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and a (union of) conjunctive query q, decide whether $\mathcal{K} \models q$.*

2.3 SPARQL

SPARQL is a W3C recommended query language for RDF [29]. It is based on the notion of query patterns defined inductively from triple patterns: a tuple $t \in \text{UBV} \times \text{UV} \times \text{UBLV}$, with V a set of variables disjoint from UBL, is called a triple pattern. Triple patterns grouped together using SPARQL operators AND and UNION form *query patterns* (or graph patterns)[1]. We use an abstract syntax that can be easily translated into the μ-calculus.

Definition 4 (Query Pattern). *A query pattern q is inductively defined as follows :*

$$q ::= \; t \in \text{UBV} \times \text{UV} \times \text{UBLV} \mid q_1 \textit{ AND } q_2 \mid q_1 \textit{ UNION } q_2$$

SPARQL has four query constructs, viz. SELECT, ASK, CONSTRUCT, and DESCRIBE. We focus on SELECT queries which are the core of SPARQL queries.

Definition 5. *A SPARQL SELECT query is a query of the form $q(\overrightarrow{w})$ where $\overrightarrow{w}$ is a tuple of variables in V which are called distinguished variables, and q is a query pattern.*

[1] We do not consider OPTIONAL and FILTER query patterns because containment over the full SPARQL is undecidable.

Example 2 (SPARQL queries). *Consider the following queries $q_1(?x)$ and $q_2(?x)$ on the graph of Example 1:*

```
SELECT ?x  WHERE  {
   { {?x ex:translated ?l} UNION {?x ex:wrote ?l} }
   ?l rdf:type ex:Poem .
}
```

```
 SELECT ?x  WHERE {
   {?x ex:translated ?l . ?l rdf:type ex:Poem .}
         UNION
   {?x ex:wrote ?l }
 }
```

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 [21].

The semantics of SPARQL queries is given by a partial mapping function ρ from V to UBL. The domain of ρ, $dom(\rho)$, is the subset of V on which ρ is defined. Two mappings ρ_1 and ρ_2 are said to be *compatible* if $\forall x \in dom(\rho_1) \cap dom(\rho_2)$, $\rho_1(x) = \rho_2(x)$. Further, if ρ_1 and ρ_2 are compatible, then $\rho_1 \cup \rho_2$ is also a mapping (we use $\uplus$ when $\rho_1 \cap \rho_2 = \emptyset$). This allows for defining the join, union, and difference operations between two sets of mappings M_1 and M_2 as shown below:

$$\begin{array}{lcl} M_1 \bowtie M_2 & = & \{\rho_1 \cup \rho_2 \mid \rho_1 \in M_1, \rho_2 \in M_2 \text{ are compatible mappings } \} \\ M_1 \cup M_2 & = & \{\rho \mid \rho \in M_1 \textsf{ or } \rho \in M_2\} \\ M_1 \setminus M_2 & = & \{\rho \in M_1 \mid \forall \rho_2 \in M_2,\ \rho \textsf{ and } \rho_2 \text{ are not compatible } \} \end{array}$$

The evaluation of query patterns over an RDF graph G is inductively defined as follows:

$$[\![\cdot]\!]_G : q \to 2^{V \times UBL}$$

$$[\![t]\!]_G = \{\rho \mid dom(\rho) = var(t) \text{ and } \rho(t) \in G\}$$

where $var(t)$ is the set of variables occurring in t.

$$\begin{aligned} [\![q_1 \textsf{ AND } q_2]\!]_G &= [\![q_1]\!]_G \bowtie [\![q_2]\!]_G \\ [\![q_1 \textsf{ UNION } q_2]\!]_G &= [\![q_1]\!]_G \cup [\![q_2]\!]_G \\ [\![q\{\overrightarrow{w}\}]\!]_G &= \pi_{\overrightarrow{w}}([\![q]\!]_G) \end{aligned}$$

Where the projection operator $\pi_{\overrightarrow{w}}$ selects only those part of the mappings relevant to variables in $\overrightarrow{w}$.

Example 3 (Answers to SPARQL queries). *The answers to query q_1 and q_2 of Example 2 on graph G of Example 1 are respectively $\{Poe, Mallarme\}$ and $\{Baudelaire, Poe, Mallarme\}$. Hence, $[\![q_1]\!]_G \subseteq [\![q_2]\!]_G$.*

Beyond this particular example, the goal of query containment is to determine whether this holds for any graph. Furthermore, a query under a set of schema axioms is a query whose answers are given with respect to graphs satisfying these axioms. Query containment under axioms can be defined as:

Definition 6 (Containment). *Given a set of axioms $\mathcal{C}$ and two queries q and q' with the same arity, q_1 is contained in q_2 with respect to $\mathcal{C}$, denoted $q \sqsubseteq_{\mathcal{C}} q'$, iff $[\![q]\!]_G \subseteq [\![q']\!]_G$ for every graph G satisfying $\mathcal{C}$.*

Definition 7 (Equivalence). *Two queries q and q' under a set of axioms $\mathcal{C}$ are equivalent, i.e., $q \equiv_{\mathcal{C}} q'$, iff $q \sqsubseteq_{\mathcal{C}} q'$ and $q' \sqsubseteq_{\mathcal{C}} q$.*

Complexity The evaluation of SPARQL queries is proved to be PSPACE-complete. However, the evaluation problem is NP-complete for the fragment containing only AND, and UNION query patterns [27].

In the next section, we introduce the modal logic μ-calculus and an encoding of RDF graphs as transition systems.

3 RDF Graphs as Transition Systems

Before presenting the encoding of RDF graphs as transition systems over which the μ-calculus is interpreted, we introduce the syntax and semantics of the μ-calculus.

3.1 μ-calculus

The modal μ-calculus [24] is an expressive logic which adds recursive features to modal logic using fixpoint operators. The syntax of the μ-calculus is composed of countable sets of *atomic propositions AP*, a set of *nominals Nom*, a set of *variables* Var, and a set of *programs* Prog for navigating in graphs. A μ-calculus formula, φ, can be defined inductively as follows:

$$\begin{aligned} \varphi \quad ::= \quad & \top \mid \bot \mid p \mid X \mid \neg\varphi \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \langle a \rangle \varphi \mid [a]\varphi \mid \\ & \mu X \varphi \mid \nu X \varphi \end{aligned}$$

where $p \in AP, X \in Var$ and $a \in Prog$ is either an atomic program or its converse $\bar{a}$. The greatest and least fixpoint operators (ν and μ) respectively introduce general and finite recursion in graphs [24].

The semantics of the μ-calculus is given over a transition system, $K = (S, R, L)$ where S is a non-empty set of nodes, $R : Prog \rightarrow 2^{S \times S}$ is the transition function, and $L : AP \rightarrow 2^S$ assigns a set of nodes to each atomic proposition or nominal where it holds, such that $L(p)$ is a *singleton* for each nominal p. For converse programs, R can be extended as $R(\bar{a}) = \{(s', s) \mid (s, s') \in R(a)\}$. In addition, a valuation function $V : \text{Var} \rightarrow 2^S$ is used to assign a set of nodes to each variable. For a valuation V, variable X, and a set of nodes $S' \subseteq S$, $V[X/S']$ is the valuation that is obtained from V by assigning S' to X. The semantics of a formula, in terms of a transition system K (a.k.a. Kripke structure) and a valuation function, is represented by $[\![\varphi]\!]^K_V$. The semantics of basic μ-calculus formulae is defined as follows:

$$\begin{aligned}
[\![\top]\!]^K_V &= S \qquad [\![\bot]\!]^K_V \ = \ \emptyset \\
[\![p]\!]^K_V &= L(p), p \in AP \cup Nom, \\
& \quad L(p) \text{ is singleton for } p \in Nom \\
[\![X]\!]^K_V &= V(X), X \in \text{Var} \quad [\![\neg\varphi]\!]^K_V \ = \ S \backslash [\![\varphi]\!]^K_V \\
[\![\varphi \wedge \psi]\!]^K_V &= [\![\varphi]\!]^K_V \cap [\![\psi]\!]^K_V, \quad [\![\varphi \vee \psi]\!]^K_V \ = \ [\![\varphi]\!]^K_V \cup [\![\psi]\!]^K_V \\
[\![\langle a \rangle \varphi]\!]^K_V &= \{s \in S \mid \exists s' \in S.(s, s') \in R(a) \ \wedge s' \in [\![\varphi]\!]^K_V\} \\
[\![[a]\varphi]\!]^K_V &= \{s \in S \mid \forall s' \in S.(s, s') \in R(a) \Rightarrow s' \in [\![\varphi]\!]^K_V\} \\
[\![\mu X \varphi]\!]^K_V &= \bigcap \{S' \subseteq S \mid [\![\varphi]\!]^K_{V[X/S']} \subseteq S'\} \\
[\![\nu X \varphi]\!]^K_V &= \bigcup \{S' \subseteq S \mid S' \subseteq [\![\varphi]\!]^K_{V[X/S']}\}
\end{aligned}$$

The next sections introduce a representation of RDF graphs as transition systems and queries as μ-calculus formulas.

3.2 Encoding of RDF graphs

An RDF graph is encoded as a transition system in which nodes correspond to RDF entities and RDF triples. Edges relate entities to the triples they occur in. Different edges are used for distinguishing the functions (subject, object, predicate). Expressing predicates as nodes, instead of atomic programs, makes it possible to deal with full RDF expressiveness in which a predicate may also be the subject or object of a statement.

Definition 8 (Transition system associated to an RDF graph)**.** *Given an RDF graph, $G \subseteq UB \times U \times UBL$, the transition system associated to G, $\sigma(G) = (S, R, L)$ over $AP = \mathrm{UBL} \cup \{s', s''\}$, is such that:*

- *$S = S' \cup S''$ with S' and S'' the smallest sets such that $\forall u \in U_G, \exists n^u \in S'$, $\forall b \in B_G, \exists n^b \in S'$, and $\forall l \in L_G, \exists n^l \in S''$,*
- *$\forall t = (s, p, o) \in G$, $\langle n^s, n^t \rangle \in R(s)$, $\langle n^t, n^p \rangle \in R(p)$, and $\langle n^t, n^o \rangle \in R(o)$,*
- *$L : \mathrm{AP} \rightarrow 2^S$; $\forall u \in U_G, L(u) = \{n^u\}$, $\forall b \in B_G, L(b) = S'$, $L(s') = S'$, $\forall l \in L_G, L(l) = \{n^l\}$ and $L(s'') = S''$,*
- *$\forall n^t, n^{t'} \in S''$, $\langle n^t, n^{t'} \rangle \in R(d)$.*

The program d is introduced to render each triple accessible to the others and thus facilitate the encoding of queries. The function σ associates what we call a *restricted transition system* to any RDF graph. Formally, we say that a transition system K is a *restricted transition system* iff there exists an RDF graph G such that $K = \sigma(G)$.

A restricted transition system is thus a bipartite graph composed of two sets of nodes: S', those corresponding to RDF entities, and S'', those corresponding to RDF triples. For example, Figure 1 shows the restricted transition system associated with the graph of Example 1.

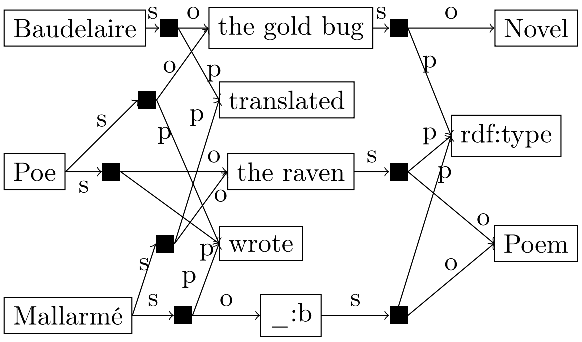


Figure 1: Transition system encoding the RDF graph of Example 1. Nodes in S'' are black anonymous nodes; nodes in S' are the other nodes (d-transitions are not displayed).

When checking for query containment, we consider the following constraints:

- The set of programs is fixed: $Prog = \{s, p, o, d, \bar{s}, \bar{p}, \bar{o}, \bar{d}\}$.
- A model must be a restricted transition system.

This last constraint can be expressed in the μ-calculus as follows:

Proposition 1 (RDF restriction on transition systems [11])**.** *A formula φ is satisfied by some restricted transition system if and only if $\varphi \wedge \varphi_r$ is satisfiable by some transition system, i.e. $\exists K_r [\![\varphi]\!]^{K_r} \neq \emptyset \iff \exists K [\![\varphi \wedge \varphi_r]\!]^{K} \neq \emptyset$, where:*

$$\varphi_r = \quad \nu X.\theta \wedge \kappa \ \wedge (\neg\langle d\rangle\top \vee \langle d\rangle X)$$

in which $\theta = \langle\bar{s}\rangle s' \wedge \langle p\rangle s' \wedge \langle o\rangle s' \wedge \neg\langle s\rangle\top \wedge \neg\langle\bar{p}\rangle\top \wedge \neg\langle\bar{o}\rangle\top$ and $\kappa = [\bar{s}]\xi \wedge [p]\xi \wedge [o]\xi$ with

$$\begin{aligned}\xi = (&\neg\langle\bar{s}\rangle\top \wedge \neg\langle o\rangle\top \wedge \neg\langle p\rangle\top \wedge \neg\langle d\rangle\top \wedge \neg\langle\bar{d}\rangle\top \\ &\wedge \neg\langle s\rangle s' \wedge \neg\langle\bar{o}\rangle s' \wedge \neg\langle\bar{p}\rangle s).\end{aligned}$$

The formula φ_r ensures that θ and κ hold in every node reachable by a d edge, i.e. in every triple node. The formula θ forces each s'' node to have one and only one subject, predicate and object. The formula κ navigates from a s'' node to every reachable s' node, and forces the latter not to be directly connected to other subject, predicate or object nodes.

If a μ-calculus formula ψ appears under the scope of a least μ or greatest ν fixed point operator over all the programs $\{s, p, o, d, \bar{s}, \bar{p}, \bar{o}, \bar{d}\}$ 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 the sake of legibility, we denote the recursion components of the respective formulae as $mu(X)$ for the μ recursion part and $nu(X)$ for the ν recursion part. Thus, the formulae become $\mu X.\psi \vee mu(X)$ and $\nu X.\psi \wedge nu(X)$.

4 SPARQL Query Containment

In this section, we encode queries and schema axioms as μ-calculus formulas. Then, we reduce query containment under schemas to μ-calculus unsatisfiability and prove the correctness of this reduction.

4.1 Encoding Queries as μ-calculus Formulae

In this section, we discuss the encoding of the containment problem $q_1(\overrightarrow{w}) \sqsubseteq q_2(\overrightarrow{w})$. For any query $q(\overrightarrow{w})$, we call the variables in $\overrightarrow{w}$ *distinguished* or answer variables. Furthermore, we denote the *non-distinguished* or existential variables in q by $ndvar(q)$, and the URIs/constants by $uris(q)$. When encoding $q_1 \sqsubseteq q_2$, we call q_1 the left-hand side query and q_2 the right-hand side query.

Queries are translated into μ-calculus formulas. The principle of the translation is that each triple pattern is associated with a sub-formula stating the existence of the triple somewhere in the graph. Hence, they are quantified by μ so as to put them out of the context of a state. In this translation, variables are replaced by nominals or some formula that are satisfied when they are at the corresponding position in such triple relations. A function called $\mathcal{A}$ is used to encode queries inductively on the structure of query patterns. AND and UNION are translated into boolean connectives $\wedge$ and $\vee$ respectively.

Encoding left-hand side query:

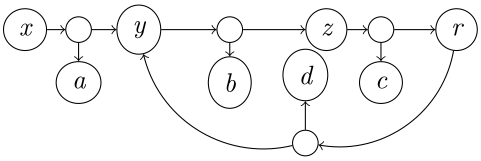
q_1 is frozen, that is, every term in q_1 becomes a nominal in μ-calculus. Here we introduce two sets of nominals, one set for denoting constants and the other for the distinguished variables. In a state of the transition system if a nominal encoding a constant is true, then no other nominal is true in this state i.e., $n_{c_1} \wedge n_{c_2}$ does not hold for two distinct constants c_1 and c_2. These can be incorporated in a μ-calculus satisfiability solver as it is done in [16]. Further, function $\mathcal{A}$ is used to recursively compute a μ-calculus formula corresponding to q_1. The encoding of the SPARQL query is $\mathcal{A}(q)$ such that:

$$\begin{aligned}\mathcal{A}((x,y,z)) &= \mu X.\big(\langle\bar{s}\rangle x \wedge \langle p\rangle y \wedge \langle o\rangle z\big) \vee mu(X)\\ \mathcal{A}(q_1 \textsf{ AND } q_2) &= \mathcal{A}(q_1) \wedge \mathcal{A}(q_2)\\ \mathcal{A}(q_1 \textsf{ UNION } q_2) &= \mathcal{A}(q_1) \vee \mathcal{A}(q_2)\end{aligned}$$

In order to encode the right-hand side query, we need the notion of cyclic queries.

Definition 9 (Cyclic Query)**.** *A SPARQL query is referred to as cyclic iff a transition graph induced from the query patterns is cyclic. The transition graph* [2] *is constructed in the same way as done in Definition 8.*

Example 4. *q is cyclic, as shown graphically,* $q(x) = (x,a,y),(y,b,z),(z,c,r),(r,d,y)$



Encoding right-hand side query:

the encoding of the right-hand side query q' is different from that of the left due to the non-distinguished variables that appear in cycles in the query. The distinguished variables and constants are encoded as nominals whereas the non-distinguished variables $ndvar(q')$ are encoded as follows:

- First, for each triple $t_i \in q'$, introduce a fresh nominal n_i, i.e., $t(t_i) = n_i$. This nominal is satisfied in a triple node S'' in a restricted transition system.
- Second, we use a function that assigns a formula for each $x \in ndvar(q')$ as follows:
  - If x occurs only once in q', x is encoded as $\top$.
  - If x appears multiple times in q' and
    - x is a subject of triple $t_i \in q'$, then it is encoded as $\langle s\rangle t(t_i)$.
    - x is a predicate of triple $t_i \in q'$, then it is encoded as $\langle\bar{p}\rangle t(t_i)$.
    - x is an object of triple $t_i \in q'$, then it is encoded as $\langle\bar{o}\rangle t(t_i)$.

  $$m_i = \{x \mapsto \varphi \mid x \in t_i \wedge \begin{cases}\varphi = \langle s\rangle t(t_i) \text{ if } subject(x) \text{ or}\\ \varphi = \langle\bar{p}\rangle t(t_i) \text{ if } predicate(x) \text{ or}\\ \varphi = \langle\bar{o}\rangle t(t_i) \text{ if } object(x)\end{cases} \}$$

  Note that there is an exponential number of m_i's in terms of the number of non-distinguished variables. More precisely, there are at most $O(n^k)$ mappings, where n is the number of triples where non-distinguished variables appear, and k is the number of non-distinguished variables.

[2] The transition graph is similar to the tuple-graph used in [7] to detect the dependency among variables.

- Finally, the function $\mathcal{A}$ uses t and m to encode the query inductively:

$$\begin{aligned}
\mathcal{A}(q,m) &= \bigvee_{i=1}^{|m|} \mathcal{A}(q,m_i)\\
\mathcal{A}((x,y,z),m) &= \mu X.\big(t((x,y,z)) \wedge \langle\bar{s}\rangle d(m,x)\\
&\quad \wedge \langle p\rangle d(m,y) \wedge \langle o\rangle d(m,z)\big) \vee mu(X)\\
\mathcal{A}(q_1 \text{ AND } q_2,m) &= \mathcal{A}(q_1,m) \wedge \mathcal{A}(q_2,m)\\
\mathcal{A}(q_1 \text{ UNION } q_2,m) &= \mathcal{A}(q_1,m) \vee \mathcal{A}(q_2,m)\\
d(m,x) &= \begin{cases} \varphi & \text{if } (x \mapsto \varphi) \in m\\ \top & \text{if } unique(x)\\ x & \text{otherwise} \end{cases}
\end{aligned}$$

The disjuncts in the encoding guarantee that possible set of substitutions m capture the intended semantics of a cyclic query.

Example 5 (SPARQL query encoding)**.** *Consider the encoding of $q_1 \sqsubseteq q_2$ of Example 2. To encode q_1, freeze the variables and constants and proceed with $\mathcal{A}$ such that $\mathcal{A}(q_1) =$*

$$\begin{aligned}
&\big((\mu X.(\langle\bar{s}\rangle x \wedge \langle p\rangle \texttt{translated} \wedge \langle o\rangle l) \vee mu(X))\\
&\quad \vee (\mu X.(\langle\bar{s}\rangle x \wedge \langle p\rangle \texttt{wrote} \wedge \langle o\rangle l) \vee mu(X))\big)\wedge\\
&(\mu X.(\langle\bar{s}\rangle l \wedge \langle p\rangle \texttt{type} \wedge \langle o\rangle \texttt{Poem}) \vee mu(X))
\end{aligned}$$

To encode q_2, one first computes t and m. Hence, $t((x, translated, l)) = n_1$, $t((l, type, Poem)) = n_2$, $t((x, wrote, l)) = n_3$ and $m = \{m_1, m_2, m_3\}$ where $m_1 = \{y \mapsto \langle\bar{o}\rangle n_1\}$, $m_2 = \{y \mapsto \langle s\rangle n_2\}$, and $m_3 = \{y \mapsto \langle\bar{o}\rangle n_3\}$. Finally, $\mathcal{A}(q_2,m) = \bigvee_{i=1}^{|m|} \mathcal{A}(q,m_i) =$

$$\begin{aligned}
&\big((\mu X.(n_1 \wedge \langle\bar{s}\rangle x \wedge \langle p\rangle \texttt{translated} \wedge \langle o\rangle\langle\bar{o}\rangle n_1) \vee mu(X)\\
&\quad \wedge \mu X.(n_2 \wedge \langle\bar{s}\rangle\langle\bar{o}\rangle n_1 \wedge \langle p\rangle \texttt{type} \wedge \langle o\rangle \texttt{Poem}) \vee mu(X))\\
&\quad \vee \mu X.(n_3 \wedge \langle\bar{s}\rangle x \wedge \langle p\rangle \texttt{wrote} \wedge \langle o\rangle\langle\bar{o}\rangle n_1) \vee mu(X)\big)\vee\\
&\big((\mu X.(n_1 \wedge \langle\bar{s}\rangle x \wedge \langle p\rangle \texttt{translated} \wedge \langle o\rangle\langle s\rangle n_2) \vee mu(X)\\
&\quad \wedge \big(\mu X.(n_2 \wedge \langle\bar{s}\rangle\langle s\rangle n_2 \wedge \langle p\rangle \texttt{type} \wedge \langle o\rangle \texttt{Poem}) \vee mu(X))\\
&\quad \vee \mu X.(n_3 \wedge \langle\bar{s}\rangle x \wedge \langle p\rangle \texttt{wrote} \wedge \langle o\rangle\langle s\rangle n_2) \vee mu(X)\big)\vee\\
&\big((\mu X.(n_1 \wedge \langle\bar{s}\rangle x \wedge \langle p\rangle \texttt{translated} \wedge \langle o\rangle\langle\bar{o}\rangle n_3) \vee mu(X)\\
&\quad \wedge \mu X.(n_2 \wedge \langle\bar{s}\rangle\langle\bar{o}\rangle n_3 \wedge \langle p\rangle \texttt{type} \wedge \langle o\rangle \texttt{Poem}) \vee mu(X))\\
&\quad \vee \mu X.(n_3 \wedge \langle\bar{s}\rangle x \wedge \langle p\rangle \texttt{wrote} \wedge \langle o\rangle\langle\bar{o}\rangle n_3) \vee mu(X)\big)
\end{aligned}$$

4.2 Encoding Axioms

In this section, we provide the encoding of $\mathcal{SHI}$ axioms that are used together with query encodings to determine if any two queries are contained in each other.

Definition 10 (μ-calculus encoding of a Schema)**.** *Given a set of axioms $c_1, c_2, ..., c_n$ of a schema $\mathcal{C}$, the μ-calculus encoding of $\mathcal{C}$ is:*

$$\eta(\mathcal{C}) = \eta(c_1) \wedge \eta(c_2) \wedge ... \wedge \eta(c_n).$$

Where η translates each axiom into an equivalent formula using ω which in turn recursively encodes concepts and roles:

- *Concept Inclusion*

$$\begin{aligned}
\eta(C_1 \sqsubseteq C_2) &= \nu X.\ \big(\omega(C_1) \Rightarrow \omega(C_2)\big) \wedge nu(X) \\
\omega(A) &= A \qquad \omega(\neg C) = \ \neg\omega(C) \qquad \omega(\bot) = \ \bot \\
\omega(C_1 \sqcap C_2) &= \omega(C_1) \wedge \omega(C_2) \\
\omega(\exists r.C) &= \langle s\rangle\big(\langle p\rangle R \wedge \langle o\rangle(\langle s\rangle\langle o\rangle\omega(C))\big) \\
\omega(\forall r.C) &= [s]\big([p]R \Rightarrow [o]([s][o]\omega(C))\big) \\
\omega(\exists r^{-}.C) &= \langle \bar{o}\rangle\big(\langle p\rangle r \wedge \langle \bar{s}\rangle(\langle s\rangle\langle o\rangle\omega(C))\big) \\
\omega(\forall r^{-}.C) &= [\bar{o}]\big([p]r \Rightarrow [\bar{s}]([s][o]\omega(C))\big)
\end{aligned}$$

- *Role Inclusion*

$$\begin{aligned}
\eta(r_1 \sqsubseteq r_2) &= \nu X.\ \big(\omega(r_1) \Rightarrow \omega(r_2)\big) \wedge nu(X) \\
\omega(r) &= r
\end{aligned}$$

- *Transitivity*

$$\eta(trans(r)) = \nu X.\langle s\rangle\big(\langle p\rangle r \wedge \langle o\rangle(y \wedge \langle s\rangle(\langle p\rangle r \wedge \langle o\rangle z)) \Rightarrow (\langle p\rangle r \wedge \langle o\rangle z)\big) \wedge nu(X)$$

  Note that y and z are fresh atomic propositions.

So far we proposed various functions to produce formulas corresponding to the encodings of queries and schema axioms. Hence, the problem of containment under a schema can be reduced to formula unsatisfiability in μ-calculus as:

$$q \sqsubseteq_{\mathcal{C}} q' \Leftrightarrow \eta(\mathcal{C}) \wedge \mathcal{A}(q) \wedge \neg\mathcal{A}(q', m) \wedge \varphi_r \text{ is unsatisfiable.}$$

For the sake of legibility in writing, we use $\Phi(\mathcal{C}, q, q')$ to denote $\eta(\mathcal{C}) \wedge \mathcal{A}(q) \wedge \neg\mathcal{A}(q', m) \wedge \varphi_r$.

4.3 Reducing Containment to Unsatisfiability

We prove the correctness of reducing query containment to unsatisfiability test.

Lemma 1. *Given a set of schema axioms $\mathcal{C} = \{c_1, \cdots, c_n\}$, $\mathcal{C}$ has a model iff $\eta(\mathcal{C})$ is satisfiable.*

Proof. ($\Rightarrow$) assume that there exists a model $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ of $\mathcal{C}$ such that $\mathcal{I} \models \mathcal{C}$. We build a restricted transition system $K = (S, R, L)$ from $\mathcal{I}$ using the following:

- for each element of the domain $e \in \Delta^{\mathcal{I}}$, we create a node $n^e \in S'$,
- for each atomic concept A, if $a \in A^{\mathcal{I}}$, then $(n^a, t) \in R(s)$, $(t, n^{type}) \in R(p)$, $(t, n^A) \in R(o)$, $L(type), = n^{type}$, $L(A) = n^A$ and $L(a) = n^a$ where $t \in S''$,
- for each atomic role R, if $(x, y) \in R^{\mathcal{I}}$, then $(n^x, t) \in R(s)$, $(t, n^R) \in R(p)$, and $(t, n^y) \in R(o)$ such that $n^x, n^y, n^R \in S'$, $t \in S''$, and $L(x) = n^x$, $L(R) = n^R$, $L(y) = n^y$,
- $S = S' \cup S''$

To show that $\eta(\mathcal{C})$ is satisfiable in K. We proceed inductively on the construction of the formula. Since the axioms $c_1, \cdots, c_n$ are made of role or concept inclusions or transitivity, we consider the following cases:

- when $\eta(c_i) = \nu X.\big(\omega(C_1) \Rightarrow \omega(C_2)\big) \wedge nurec(X)$. Since $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$, we get that $[\![\omega(C_1)]\!]^K \subseteq [\![\omega(C_2)]\!]^K$. And hence, $\omega(C_1) \Rightarrow \omega(C_2)$ is satisfiable in K. Besides, the general recursion ν guarantees that the constraint is satisfied in each state of the transition system. Therefore, $\eta(c_i)$ is satisfiable.
- when $\eta(c_i) = \nu X.\big(\omega(r_1) \Rightarrow \omega(r_2)\big) \wedge nurec(X)$. From $r_1^{\mathcal{I}} \subseteq r_2^{\mathcal{I}}$ we have that $\exists n^{r_1} \in L(r_1)$ implies $\exists n^{r_2} \in L(r_2)$ in K. Thus, $\exists s \in [\![\omega(r_1) \Rightarrow \omega(r_2)]\!]^K$. As K is a construction of $\mathcal{I}$, $\eta(c_i)$ is satisfiable in K.
- when $\eta(c_i) = \eta(trans(r))$. Starting from the assumption, we have that $\mathcal{I} \models trans(r)$, thus $(x, y) \in r^{\mathcal{I}} \wedge (y, z) \in r^{\mathcal{I}} \wedge (x, z) \in r^{\mathcal{I}}$. Henceforth, K contains states n^x, n^y, n^z, and n^r where x, y, z, and r are true. This implies that $\eta(trans(r))$ is true in K and as a result satisfiable.

Since K is a model of each $\eta(c_i)$, then $\eta(\mathcal{C})$ is satisfiable.
($\Leftarrow$) consider a transition system model K for $\eta(\mathcal{C})$. From K, we construct an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ and show that it is a model of $\mathcal{C}$.

- $\Delta^{\mathcal{I}} = S$, $A^{\mathcal{I}} = [\![A]\!]^K$ for each atomic concept A,
- $\top^{\mathcal{I}} = [\![\top]\!]^K$, for a top concept,
- $r^{\mathcal{I}} = \{(s, s') \mid \forall t \in [\![r]\!]^K \wedge t' \in S \wedge (s, t') \in R(s) \wedge (t', t) \in R(p) \wedge (t', s') \in R(o)\}$ for each atomic role r,

Consequently, formulas such as $\nu X.\big(\omega(r_1) \Rightarrow \omega(r_2)\big) \wedge nurec(X)$ and $\nu X.\big(\omega(C_1) \Rightarrow \omega(C_2)\big) \wedge nurec(X)$ are true in $\mathcal{I}$. The first formula expresses that there is no node in the transition system where $\omega(r_1)$ holds and $\omega(r_2)$ does not hold. This is equivalent to $\omega(r_1) \Rightarrow \omega(r_2)$ and $[\![r_1]\!]^K \subseteq [\![r_2]\!]^K$ since r_1 and r_2 are basic roles. Thus, we obtain $r_1^{\mathcal{I}} \subseteq r_2^{\mathcal{I}}$ and $\mathcal{I} \models r_1 \sqsubseteq r_2$.

On the other hand, for the latter formula from above, one can exploit its construction. Note however that, similar justifications as above can be worked out to arrive at $\mathcal{I} \models C_1 \sqsubseteq C_2$ if C_1 and C_2 are basic concepts. Nonetheless, if they are complex concepts, we proceed as below. Consider the case when $C_1 = A \sqcap B$ and $C_2 = \exists R.C$, $[\![\omega(C_1) \Rightarrow \omega(C_2)]\!]^K$

$$\begin{aligned}
&\Leftrightarrow [\![\omega(A \sqcap B)]\!]^K \subseteq [\![\omega(\exists R.C)]\!]^K\\
&\Leftrightarrow [\![A \wedge B]\!]^K \subseteq [\![\langle s\rangle\big(\langle p\rangle R \wedge \langle o\rangle(\langle s\rangle\langle o\rangle C)\big)]\!]^K\\
&\Leftrightarrow [\![A]\!]^K \wedge [\![B]\!]^K \subseteq \{s \mid \exists s'.s \in [\![\langle s\rangle\langle p\rangle R]\!]^K \wedge s' \in [\![\langle s\rangle\langle o\rangle C]\!]^K\}\\
&\Leftrightarrow A^{\mathcal{I}} \cap B^{\mathcal{I}} \subseteq \{s \mid \exists s'.(s, s') \in R^{\mathcal{I}} \wedge s' \in C^{\mathcal{I}}\}\\
&\Leftrightarrow (A \sqcap B)^{\mathcal{I}} \subseteq (\exists R.C)^{\mathcal{I}}\\
&\Leftrightarrow \mathcal{I} \models C_1 \sqsubseteq C_2
\end{aligned}$$

Accordingly, from $\mathcal{I} \models c_1 \wedge \cdots \wedge \mathcal{I} \models c_n$, it follows that $\mathcal{I} \models \mathcal{C}$. □

Theorem 1. *Given a query $q(\overrightarrow{w})$, there exists an RDF graph G such that $[\![q(\overrightarrow{w})]\!]_G \neq \emptyset$.*

Proof. (*Sketch*) From any query it is possible to build an homomorphic graph by collecting all triples connected by AND and only those at the left of UNION (replacing variables by blanks). This graph is consistent as all RDF graphs [18]. It is thus a graph satisfying the query. □

Lemma 2. *For any SPARQL query q, q is satisfiable iff $\mathcal{A}(q) \wedge \varphi_r$ and $\mathcal{A}(q,m) \wedge \varphi_r$ are satisfiable.*

Proof. We prove for $\mathcal{A}(q,m) \wedge \varphi_r$, the proof for $\mathcal{A}(q) \wedge \varphi_r$ follows immediately.
($\Rightarrow$) q is satisfiable implies there exists at least a canonical instance of q as a graph G such that $[\![q]\!]_G \neq \emptyset$ (cf. Theorem 1). Now, we construct a restricted transition system $\sigma(G) = (S, R, L)$ in the same way as it is done in Definition 8. To prove that $\sigma(G)$ is a model of $\mathcal{A}(q,m)$, we consider two cases:

(i) when q is cyclic, and

(ii) when q is cycle-free

First, (i) consider when q is cyclic, in this case, its encoding is $\bigvee_{i=1}^{|m|} \mathcal{A}(q, m_i)$. From this encoding it can be seen that nominals are introduced and set to be true in S'' nodes. With these nominals, one can successfully create a formula that can encode multiply occurring non-distinguished variables. Henceforth, creating a formula that is satisfiable in cyclic models.

It can be verified that $\sigma(G)$ is a model for one of the disjuncts $\mathcal{A}(q, m_i)$, this is because nominals encoding the constants and distinguished variables are true in $\sigma(G)$ as they exist already in G. In addition, the subformula $\langle s \rangle n_i \mid \langle \bar{o} \rangle n_i \mid \langle \bar{p} \rangle n_i$ depending on where the non-distinguished variable appears in $t_i \in q$ guarantees that n_i is true in the triple node S'' and any labelling matches the node denoting the non-distinguished variable. Therefore, $\mathcal{A}(q,m)$ is satisfiable in $\sigma(G)$. To elaborate, if $l \in (x,y,z) \in q$

- for l either a distinguished variable or constant, l is satisfiable in $\sigma(G)$ since $[\![l]\!]^{\sigma(G)} \in L(l)$,
- for l a uniquely appearing non-distinguished variable, l is true in $\sigma(G)$ since its encoding $\top$ is true everywhere in the transition system,
- for l a multiply occurring non-distinguished variable is true in $\sigma(G)$ since $\exists t \in S''.\ t \in L(n) \wedge t \in [\![n \wedge \langle o \rangle \top]\!]^{\sigma(G)}$. Where n is a nominal denoting the reified triple (x,y,z).

If (ii) q is cycle-free, then encoding the non-distinguished variable with $\top$ suffices to justify that $\sigma(G)$ is a model of its encoding.
($\Leftarrow$) Assume that $\mathcal{A}(q,m) \wedge \varphi_r$ is satisfiable. This implies that there exists a restricted transition system $K = (S, R, L)$ such that $[\![\mathcal{A}(q,m) \wedge \varphi_r]\!]^K \neq \emptyset$. We build an RDF graph G from K as follows:

- if $\forall s_1, s_2, s_3 \in S' \wedge t \in S''.(s_1, t) \in R(s) \wedge (t, s_2) \in R(p) \wedge (t, s_3) \in R(o)$ and for each triple $t_i = (x_i, y_i, z_i) \in q$ if $s_1 \in L(x_i) \wedge s_2 \in L(y_i) \wedge s_3 \in L(z_i) \wedge z_i \in [\![\top]\!]^K$, then $(x_i, y_i, z_i) \in G$. This case holds if x_i, y_i and z_i are either distinguished variables or constants. Note here that if x_i or y_i or z_i appear in another triple $t_j = (x_j, y_j, z_j) \in q$, then the equivalent item in t_j is replaced with the value of the corresponding entry in t_i.
- if $\forall s_1, s_2, s_3 \in S' \wedge t \in S''.(s_1, t) \in R(s) \wedge (t, s_2) \in R(p) \wedge (t, s_3) \in R(o)$ and for each triple $t_i = (x_i, y_i, z_i) \in q$ if $s_1 \in L(x_i) \wedge s_2 \in L(y_i)$, then $(x_i, y_i, c_i) \in G$ where c_i is a fresh constant. This case holds if z_i is a non-distinguished variable. Similarly, the case when x_i or y_i or both are variables can be worked out.
- if $\forall s_1, s_2, s_3 \in S' \wedge t \in S''.(s_1, t) \in R(s) \wedge (t, s_2) \in R(p) \wedge (t, s_3) \in R(o)$ and for each triple $t_i = (x_i, y_i, z_i) \in q$ and x_i is a non-distinguished variable that appears in a cycle and if $s_1 \in [\![\top]\!]^K \wedge s_2 \in L(y_i) \wedge s_3 \in L(z_i)$, then $(c_i, y_i, z_i) \in G$. Where c_i is a fresh constant and all such occurrences of the variable x_i in other triples is replaced by c_i.

Since G is a technical construction obtained from an instance of q, then it holds that $[\![q]\!]_G \neq \emptyset$. Thus, q is satisfiable. □

Theorem 2 (Soundness)**.** *Given two SPARQL queries $q_1(\overrightarrow{w})$ and $q_2(\overrightarrow{w})$, and a set of axioms $\mathcal{C}$, if $\eta(\mathcal{C}) \wedge \mathcal{A}(q_1) \wedge \neg\mathcal{A}(q_2, m) \wedge \varphi_r$ is unsatisfiable, then $q_1(\overrightarrow{w}) \sqsubseteq_{\mathcal{C}} q_2(\overrightarrow{w})$.*

Proof. We show the contrapositive. If $q_1 \not\sqsubseteq_{\mathcal{C}} q_2$, then $\Phi(\mathcal{C}, q_1, q_2)$ is satisfiable. One can verify that every model G of $\mathcal{C}$ in which there is at least one tuple satisfying q_1 but not q_2 can be turned into a transition system model for $\Phi(\mathcal{C}, q_1, q_2)$. To do so, consider a graph G that satisfies schema axioms $\mathcal{C}$. Assume also that there is a tuple $\overrightarrow{a} \in [\![q_1]\!]_G$ and $\overrightarrow{a} \notin [\![q_2]\!]_G$. Let us construct a transition system K from G. From Lemma 1, we obtain that $[\![\eta(\mathcal{C})]\!]^K \neq \emptyset$. Further, since K is a restricted transition system (cf. Definition 8), $[\![\varphi_r]\!]^K \neq \emptyset$. At this point, it remains to verify that $[\![\mathcal{A}(q_1)]\!]^K \neq \emptyset$ and $[\![\mathcal{A}(q_2)]\!]^K = \emptyset$.

Let us construct the formulas $\mathcal{A}(q_1)$ and $\mathcal{A}(q_2, m)$ by first skolemizing the distinguished variables using the answer tuple $\overrightarrow{a}$. Consequently, from Lemma 2 one obtains, $[\![\mathcal{A}(q_1)]\!]^K \neq \emptyset$. However, $[\![\mathcal{A}(q_2, m)]\!]^K = \emptyset$, this is because the nominals in the formula corresponding to the constants and non-distinguished variables are not satisfied in K. This implies that $[\![\neg\mathcal{A}(q_2, m)]\!]^K \neq \emptyset$. This is justified by the fact that if a formula φ is a satisfiable in a restricted transition system, then $[\![\varphi]\!]^K = S$ thus $[\![\neg\varphi]\!]^K = \emptyset$. So far we have: $[\![\eta(\mathcal{C})]\!]^K \neq \emptyset$ and $[\![\varphi_r]\!]^K \neq \emptyset$ and $[\![\mathcal{A}(q_1)]\!]^K \neq \emptyset$ and $[\![\neg\mathcal{A}(q_2, m)]\!]^K \neq \emptyset$. Without loss of generality, $[\![\Phi(\mathcal{C}, q_1, q_2)]\!]^K \neq \emptyset$. Therefore, $\Phi(\mathcal{C}, q_1, q_2)$ is satisfiable. □

Theorem 3 (Completeness)**.** *Given queries $q_1(\overrightarrow{w})$ and $q_2(\overrightarrow{w})$, and a set of axioms $\mathcal{C}$, if $\eta(\mathcal{C}) \wedge \mathcal{A}(q_1) \wedge \neg\mathcal{A}(q_2, t, m) \wedge \varphi_r$ is satisfiable, then $q_1(\overrightarrow{w}) \not\sqsubseteq_{\mathcal{C}} q_2(\overrightarrow{w})$.*

Proof. $\Phi(\mathcal{C}, q, q')$ is satisfiable $\Rightarrow \exists K.[\![\Phi(\mathcal{C}, q, q')]\!]^K \neq \emptyset$. Consequently, K is a restricted transition system due to $[\![\varphi_r]\!]^K \neq \emptyset$ (cf. Proposition 1). Hence, K admits a reified triple structure. Using $K = (S' \cup S'', R, L)$ we construct a model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{C}$ such that $q \not\sqsubseteq q'$ holds:

- $\Delta^{\mathcal{I}} = S'$, $A^{\mathcal{I}} = [\![A]\!]^K$ for each atomic concept A,
- $\top^{\mathcal{I}} = [\![\top]\!]^K$, for a top concept,
- $r^{\mathcal{I}} = \{(s, s') \mid \forall t \in [\![r]\!]^K \wedge t' \in S'' \wedge (s, t') \in R(s) \wedge (t', t) \in R(p) \wedge (t', s') \in R(o)\}$ for each atomic role r,
- for each constant c in q and q', $c^{\mathcal{I}} = [\![c]\!]^K$,
- for each distinguished and non-distinguished variable v in q, $v^{\mathcal{I}} = [\![v]\!]^K$, and
- for each distinguished variable v in q', $v^{\mathcal{I}} = [\![v]\!]^K$.

One can utilize Lemma 1, to verify that indeed $\mathcal{I}$ is a model of $\mathcal{C}$. Thus, it remains to show that $[\![q]\!]_{\mathcal{I}} \not\subseteq [\![q']\!]_{\mathcal{I}}$. From our assumption, one anticipates the following:

$$\begin{aligned}
[\![\mathcal{A}(q) \wedge \neg\mathcal{A}(q')]\!]^K \neq \emptyset &\Rightarrow [\![\mathcal{A}(q)]\!]^K \neq \emptyset \text{ and } [\![\neg\mathcal{A}(q', m)]\!]^K \neq \emptyset \\
&\Rightarrow [\![\mathcal{A}(q)]\!]^K \neq \emptyset \text{ and } [\![\mathcal{A}(q', m)]\!]^K = \emptyset
\end{aligned}$$

Note here that, if a formula φ is satisfiable in a restricted transition system K_r, then $[\![\varphi]\!]^{K_r} = S$. We use a function f to construct an RDF graph G from the interpretation $\mathcal{I}$. f uses assertions in $\mathcal{I}$ to form triples:

$$\begin{aligned} f(a \in A^{\mathcal{I}}) &= (a, \texttt{type}, A) \in G \\ f((a,b) \in r^{\mathcal{I}}) &= (a, r, b) \in G \\ f((a,b) \in (r^{-})^{\mathcal{I}}) &= (b, r, a) \in G \\ f((x,y,z)) &= (x,y,z) \in G, \ \forall (x,y,z) \in q \end{aligned}$$

As a consequence, $[\![q]\!]_G \neq \emptyset$ and $[\![q']\!]_G = \emptyset$ because G contains all those triples that satisfy q and not q'. Therefore, we get $[\![q]\!]_G \not\subseteq [\![q']\!]_G$. Fundamentally, there are two issues to be addressed (i) when q' contains a cycle and (ii) when q' is not cyclic. (i) can be dealt with nominals, i.e., since cycles can be expressed by a formula in a μ-calculus extended with nominals and inverse, cyclic queries can be encoded by such a formula. Hence, the constraints expressed by $\neg\mathcal{A}(q', m)$ are satisfied in a transition system containing cycles. On the other hand, (ii) if there are no cycles in q', then replacing non-distinguished variables with $\top$ suffices (cf. the proof of Lemma 2). □

5 Complexity

In the following, we establish the complexity of the containment problem under schema axioms. The schema axioms can be formed using the fragments of $\mathcal{SHIQ}$. More specifically, the fragments without number restrictions. The expressiveness of the schema language is limited as such due to the expressive power of the logic used for the encoding: μ-calculus with nominals and converse becomes undecidable when extended with graded modalities [6].

Proposition 2 (Query satisfiability)**.** *Give a schema $\mathcal{C}$ and a query q, the complexity of satisfiability of q with respect to $\mathcal{C}$ is $2^{\mathcal{O}(|\mathcal{C}|+|q|)}$.*

Proposition 3. *SPARQL query containment under the fragments of $\mathcal{SHIQ}$ schema axioms can be determined in a time of $2^{\mathcal{O}(n^2 log\ n)}$ where $n = \mathcal{O}(|\eta(\mathcal{C})| + |\mathcal{A}(q_1)| + |\mathcal{A}(q_2)|)$ is the size of the formula, and $\eta(\mathcal{C})$, $\mathcal{A}(q_1)$ and $\mathcal{A}(q_2)$ denote the encodings of schema axioms $\mathcal{C}$, and queries q_1 and q_2.*

Note that due to duplication in the encoding of q_2, the size of $|\mathcal{A}(q_2)|$ is exponential in terms of the non-distinguished variables that appear in cycles in the query. Hence, we obtain a 2ExpTime upper bound for containment. As pointed out in [7], the problem is solvable in ExpTime if there is no cycle on the right hand side query. This complexity is a lower bound due to the complexity of satisfiability in μ-calculus which is $2^{\mathcal{O}(n^2 log\ n)}$ [30, 32].

The double exponential upper bound matches its lower bound depending on the expressiveness of the schema language. For instance, if $\mathcal{ALC}$ or $\mathcal{ALCH}$ are used instead of $\mathcal{SHI}$, then 2ExpTime is not a lower bound. In fact, if the right-hand side query is not cyclic, then ExpTime is a lower bound for DL's $\mathcal{ALC}$ and $\mathcal{ALCH}$. Here, we prove that the 2ExpTime complexity result in Proposition 3 is a lower bound for the schema language $\mathcal{SHI}$. To do so, we reduce query entailment in $\mathcal{SHI}$ to query containment in SPARQL. Conjunctive query entailment in $\mathcal{SHI}$ is already known to be 2ExpTime-hard [25].

Lemma 3. *Query containment under a TBox, $q \sqsubseteq_{\mathcal{T}} q'$, can be polynomially reduced to query entailment with respect to a knowledge base that has a frozen q as an ABox, $\mathcal{K} = \langle \mathcal{T}, f(q) \rangle \models q'$ [7].*

Theorem 4. *Query containment under $\mathcal{SHI}$ axioms is 2ExpTime-hard.*

Proof. To prove this, we reduce the problem of (union of) conjunctive query containment [25] in $\mathcal{SHI}$ to SPARQL query containment under $\mathcal{SHI}$ axioms. In doing so, we use a function π that translates union of conjunctive queries in $\mathcal{SHI}$ into SPARQL queries.

$$\begin{aligned}
\pi(q_1 \vee \cdots \vee q_n) &= \pi(q_1) \text{ UNION } \cdots \text{ UNION } \pi(q_n)\\
\pi(C(x)) &= (x, \mathtt{type}, C)\\
\pi(R(x,y)) &= (x, R, y)\\
\pi(inv(R)(x,y)) &= (y, R, x)\\
\pi(C(x), R(x,y)) &= (x, \mathtt{type}, C) \text{ AND } (x, R, y)\\
\pi(q_1, \ldots, q_n) &= \pi(q_1) \text{ AND } \cdots \text{ AND } \pi(q_n)
\end{aligned}$$

It remains to show that, given a TBox $\mathcal{T}$ and (union of) conjunctive queries q and q',

$$\pi(q) \sqsubseteq_{\mathcal{T}} \pi(q') \Leftrightarrow \mathcal{K} = \langle \mathcal{T}, f(q) \rangle \models q'$$

($\Rightarrow$) this direction is immediate from [7]. There, it has been implicitly shown that query containment under TBox axioms can be reduced to query entailment w.r.t. a knowledge base (cf. Lemma 3).
($\Leftarrow$) $\langle \mathcal{T}, f(q) \rangle \models q'$

$$\begin{aligned}
&\Rightarrow \forall \mathcal{I}.\big(\mathcal{I} \models \mathcal{T} \text{ and } \mathcal{I} \models f(q) \Rightarrow \mathcal{I} \models q'\big)\\
&\Rightarrow \forall \mathcal{I}.\big(\mathcal{I} \models \mathcal{T} \text{ and } [\![\pi(q)]\!]_G \neq \emptyset \Rightarrow [\![\pi(q')]\!]_G \neq \emptyset\big)^*\\
&\Rightarrow \forall \mathcal{I}.\big(\mathcal{I} \models \mathcal{T} \text{ and } [\![\pi(q)]\!]_G \subseteq [\![\pi(q')]\!]_G\big)\\
&\Rightarrow \pi(q) \sqsubseteq_{\mathcal{T}} \pi(q')
\end{aligned}$$

* G can be constructed from $\mathcal{I}$ using σ', by translating each assertion of the form $a \in A^{\mathcal{I}}$ into $(a, \mathtt{type}, A) \in G$ and $(a, b) \in R^{\mathcal{I}}$ into $(a, R, b) \in G$. Furthermore, the knowledge base $f(q)$ can be unfrozen to q. Now, let us verify that, if there is an interpretation for a knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{O} \rangle$, then the following holds:

$$\forall \mathcal{I}.\big(\mathcal{I} \models q \Leftrightarrow [\![\pi(q)]\!]_{G=\sigma'(I)} \neq \emptyset\big)$$

this can be proved by induction on the structure of the query.
(Base case) when $q(v) = C(v)$.
($\Rightarrow$) $\mathcal{I} \models C(v) \Rightarrow \tau(v) \in C^{\mathcal{I}}$. Hence, we get $G = \sigma'(I) = \{(\tau(v), \mathtt{type}, C)\}$. Clearly, $[\![\pi(C(v))]\!]_G = [\![(v, \mathtt{type}, C)]\!]_G \neq \emptyset$.
($\Leftarrow$) Assume $[\![\pi(C(v))]\!]_G \neq \emptyset$

$$\begin{aligned}
&\Rightarrow \exists \rho.\rho \in [\![\pi(C(v))]\!]_G\\
&\Rightarrow \exists \rho.\rho \in [\![(v, \mathtt{type}, C)]\!]_G\\
&\Rightarrow \exists \rho.(\rho(v), \rho(\mathtt{type}), \rho(C)) \in G\\
&\Rightarrow \exists \rho.(\rho(v), \mathtt{type}, C) \in G
\end{aligned}$$

From G, one can generate a DL interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ such that $C^{\mathcal{I}} = \{\rho(v)\}$. Thus, $\mathcal{I} \models C(v)$ since $\tau(v) = \rho(v) \in C^{\mathcal{I}}$.
When $q(v, v') = R(v, v')$.
($\Rightarrow$) $\mathcal{I} \models R(v, v') \Rightarrow (\tau(v), \tau(v')) \in R^{\mathcal{I}}$. Let us construct a graph G using $\mathcal{I}$, G at least contains

$\{(\tau(v), R, \tau(v'))\}$. Obvious, $[\![\pi(R(v, v'))]\!]_G = [\![(v, R, v')]\!]_G \neq \emptyset$.
($\Leftarrow$) Assume $[\![\pi(R(v, v'))]\!]_G$

$$\begin{aligned}
&\Rightarrow \exists \rho . \rho \in [\![(v, R, v')]\!]_G \\
&\Rightarrow \exists \rho . (\rho(v), \rho(R), \rho(v')) \in G \\
&\Rightarrow \exists \rho . (\rho(v), R, \rho(v')) \in G
\end{aligned}$$

Using G, let us build a DL interpretation $\mathcal{I}$ such that $R^{\mathcal{I}} = \{(\rho(v), \rho(v'))\}$. As a consequence, $\mathcal{I} \models^{\tau} R(v, v')$ since $(\tau(v), \tau(v')) = (\rho(v), \rho(v')) \in R^{\mathcal{I}}$. This concludes the proof of the base case. (Inductive case) when $q(\overrightarrow{v}) = q_1, \ldots, q_n$. Assume that $\mathcal{I} \models q_1, \ldots, q_n$

$$\begin{aligned}
&\Leftrightarrow \mathcal{I} \models^{\tau} q_1 \text{ and } \ldots \text{ and } \mathcal{I} \models^{\tau} q_n \\
&\Leftrightarrow \exists \rho . \rho \in [\![\pi(q_1)]\!]_{\sigma'(\mathcal{I})} \text{ and } \ldots \text{ and } \rho \in [\![\pi(q_n)]\!]_{\sigma'(\mathcal{I})} \text{ by induction hypothesis} \\
&\Leftrightarrow \exists \rho . \rho \in [\![\pi(q_1)]\!]_{\sigma'(\mathcal{I})} \bowtie \cdots \bowtie [\![\pi(q_n)]\!]_{\sigma'(\mathcal{I})} \\
&\Leftrightarrow \exists \rho . \rho \in [\![\pi(q_1) \text{ AND } \cdots \text{ AND } \pi(q_2)]\!]_{\sigma'(\mathcal{I})} \\
&\Leftrightarrow [\![\pi(q_1) \text{ AND } \cdots \text{ AND } \pi(q_2)]\!]_{\sigma'(\mathcal{I})} \neq \emptyset
\end{aligned}$$

When $q(\overrightarrow{v}) = q_1 \vee \cdots \vee q_n$. Starting from $\mathcal{I} \models q_1 \vee \cdots \vee q_n$

$$\begin{aligned}
&\Leftrightarrow \mathcal{I} \models^{\tau} q_1 \text{ or } \ldots \text{ or } \mathcal{I} \models^{\tau} q_n \\
&\Leftrightarrow \exists \rho . \rho \in [\![\pi(q_1)]\!]_{\sigma'(\mathcal{I})} \text{ or } \ldots \text{ or } \rho \in [\![\pi(q_n)]\!]_{\sigma'(\mathcal{I})} \text{from induction hypothesis.} \\
&\Leftrightarrow \exists \rho . \rho \in [\![\pi(q_1)]\!]_{\sigma'(\mathcal{I})} \cup \cdots \cup \rho \in [\![\pi(q_n)]\!]_{\sigma'(\mathcal{I})} \\
&\Leftrightarrow \exists \rho . \rho \in \big([\![\pi(q_1)]\!]_{\sigma'(\mathcal{I})} \cup \cdots \cup [\![\pi(q_n)]\!]_{\sigma'(\mathcal{I})}\big) \\
&\Leftrightarrow \exists \rho . \rho \in \big([\![\pi(q_1) \text{ UNION } \cdots \text{ UNION } \pi(q_n)]\!]_{\sigma'(\mathcal{I})}\big) \\
&\Leftrightarrow [\![\pi(q_1) \text{ UNION } \cdots \text{ UNION } \pi(q_n)]\!]_{\sigma'(\mathcal{I})} \neq \emptyset
\end{aligned}$$

This concludes the induction step and the overall proof. □

6 Related Works

In the following we briefly review works that previously established closely related results for related query languages. We took a similar approach as [16] that established the optimal complexity for XPath query containment and provided an effective implementation.

Studies on the translation of SPARQL into relational algebra and SQL [13, 10] indicate a close connection between SPARQL and relational algebra in terms of expressiveness. In [28], a translation of SPARQL queries into a datalog fragment (non-recursive datalog with negation) that is known to be equally expressive as relational algebra (RA) was presented. This translation makes the close connection between SPARQL and rule-based languages explicit and shows that RA is at least as expressive as SPARQL. Tackling the opposite direction, it was recently shown in [3] that SPARQL is relationally complete, by providing a translation of the above-mentioned datalog fragment into SPARQL. As argued in [3], the results from [28] and [3] taken together imply that SPARQL has the same expressive power as relational algebra. From early results on query containment in relational algebra and first-order logic, one can infer that containment in relational algebra is undecidable (contrary to the results in [12]). Therefore, containment of SPARQL queries is also undecidable. Hence, in this paper, we considered a fragment of SPARQL containing only conjunction and disjunction for this study.

Query containment has also been studied under different kinds of constraints. Results in this setting include, decidability of conjunctive query containment under functional and inclusion dependencies is studied in [22], also [1] proved decidability of this problem under functional and multi-valued dependencies. Further, decidability and undecidability results are proved in [7] for non-recursive datalog queries under expressive description logic constraints. Moreover, the undecidability is proved in [8] for recursive queries under inclusion dependencies.

The most closely related work is [7] in which query containment under description logic constraints is studied based on an encoding in propositional dynamic logic with converse (CPDL). They establish 2ExpTime upper bound complexity for containment of queries consisting of union of conjunctive queries under $\mathcal{DLR}$ schema axioms. Our work is similar in spirit, in the sense that the μ-calculus is a logic that subsumes CPDL, and may open the way for extensions of the query languages and ontologies (for instance OWL-DL). Besides, the two languages are different since SPARQL allows for predicates to be used as subject or object of other triple patterns and can be in the scope of a variable. This is not directly allowed in $\mathcal{DLR}$ (union) of conjunctive queries. Our encoding of RDF graphs and SPARQL queries preserves this capability.

Other related results come from the study of query entailment and query answering. Query entailment (and hence containment) in DLs ranging from $\mathcal{ALCI}$ to $\mathcal{SHIQ}$ is shown to be 2ExpTime-hard in [25, 17, 15]. In relation, we have introduced a novel approach of determining SPARQL query containment under a TBox using μ-calculus formula satisfiability while maintaining the same complexity bound.

In this paper we do not deal with the same query language than the one dealt with in [17]. In fact, the supported SPARQL fragment is strictly larger than the one studied in [17]. Specifically, UCQs in [17] are made of $C(x), R(x, y)$ for an atom C, a role R, and variables x and y, whereas we do also support queries capable of querying concept and role names at the same time, such as $q(x) = (x, y, z)$. Further, the purpose of reducing the problem to μ-calculus is exactly about extending query containment to even more features (such as SPARQL 1.1 paths with recursion, entailment regimes, and negation). For instance, it is known that recursive paths can be easily supported in mu-calculus (using fixpoints) whereas it is known that extending previous approaches with this feature is notoriously difficult. Beyond this, the novelty of the study is the reduction of the SPARQL containment problem to μ-calculus satisfiability, and the advantages of using such a logic: great expressivity, good computational properties, extensibility. The main focus of the contribution is not the complexity bound by itself but rather a new approach with a broader logic, paving the way for future extensions as it was never done before.

Here, we would like to emphasize that, in addition to the complexity bound we provide, no implementation has been reported in previous works, whereas in our case our work opens the way to use an implementation like the one in [32] or [16].

Finally, the evaluation of SPARQL query under schema constraints is considered by W3C under the entailment regime principle in which SPARQL queries are evaluated by taking into account the semantics of a schema language [23]. It is possible to define query containment under such entailment regimes. However, because the schema is not made explicit in entailment regimes, this would not allow to consider containment under a particular schema as we did here. And this could be very useful particularly because (1) schema are very often separated from the data and (2) this allows for compiling the schema.

7 Conclusion

We have introduced a mapping from RDF graphs into transition systems and the encodings of queries and schema axioms in the μ-calculus. We proved that this encoding is correct and can be used for checking query containment. We have provided implementable algorithms, as a consequence, this work opens a way to use available implementations of μ-calculus satisfiability solvers from [32] and [16]. Beyond this, we have established a double exponential upper bound for containment test under $\mathcal{SHI}$ axioms. This bound is further strengthened to be 2ExpTime-hard by a reduction from the query entailment problem.

As a future work, we plan to extend the schema language with nominals, such as $\mathcal{SHOI}$, and analyse the optimality of the complexity. Because nominals are part of the logic, the complexity of containment under $\mathcal{SHOI}$ axioms has already a 2ExpTime upper bound. Furthermore, it would be interesting to identify the fragments of SPARQL queries and DLs that can be encoded in versions of the μ-calculus with nominals and converse, graded modalities and converse, and nominals and graded modalities [6, 32].

References

[1] Alfred V. Aho, Yehoshua Sagiv, and Jeffrey D. Ullman. Equivalences Among Relational Expressions. *SIAM J. Comput.*, 8(2):218–246, 1979.

[2] Faisal Alkhateeb, Jean-François Baget, and Jérôme Euzenat. Extending SPARQL with regular expression patterns (for querying RDF). *J. Web Semantics*, 7(2):57–73, 2009.

[3] Renzo Angles and Claudio Gutierrez. The Expressive Power of SPARQL. *The Semantic Web-ISWC 2008*, pages 114–129, 2008.

[4] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The DL-Lite Family and Relations. *J. of Artificial Intelligence Research*, 36:1–69, 2009.

[5] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2007. ISBN 9780511717383.

[6] Piero A. Bonatti, Carsten Lutz, Aniello Murano, and Moshe Y. Vardi. The Complexity of Enriched μ-calculi. *Automata, Languages and Programming*, pages 540–551, 2006.

[7] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Conjunctive Query Containment and Answering under Description Logics Constraints. *ACM Trans. on Computational Logic*, 9(3):22.1–22.31, 2008.

[8] Diego Calvanese and Riccardo Rosati. Answering Recursive Queries under Keys and Foreign Keys is Undecidable. In *Proc. of the 10th Int. Workshop on Knowledge Representation meets Databases (KRDB 2003)*, volume 79, pages 3–14, 2003.

[9] Ashok K. Chandra and Philip M. Merlin. Optimal Implementation of Conjunctive Queries in Relational Data Bases. In *Proceedings of the ninth annual ACM symposium on Theory of computing*, pages 77–90. ACM, 1977.

[10] A. Chebotko, S. Lu, H.M. Jamil, and F. Fotouhi. Semantics preserving sparql-to-sql query translation for optional graph patterns. Technical report, Technical Report TR-DB-052006-CLJF, 2006.

[11] Melisachew Wudage Chekol, Jérôme Euzenat, Pierre Genevès, and Nabil Layaïda. PSPARQL query containment. Research report 7641, June 2011. `http://hal.inria.fr/inria-00598819/PDF/RR-7641.pdf`.

[12] Melisachew Wudage Chekol, Jérôme Euzenat, Pierre Genevès, and Nabil Layaïda. PSPARQL query containment. In *DBPL'11*, August 2011.

[13] R. Cyganiak. A relational algebra for SPARQL. *Digital Media Systems Laboratory HP Laboratories Bristol. HPL-2005-170*, 2005.

[14] Jos de Bruijn and Stijn Heymans. Logical foundations of RDF (S) with datatypes. *J. of Artificial Intelligence Research*, 38(1):535–568, 2010.

[15] T. Eiter, C. Lutz, M. Ortiz, and M. Šimkus. Query answering in description logics with transitive roles. In *Proc. of IJCAI*, pages 759–764, 2009.

[16] Pierre Genevès, Nabil Layaïda, and Alan Schmitt. Efficient Static Analysis of XML Paths and Types. In *PLDI '07*, pages 342–351, New York, NY, USA, 2007. ACM.

[17] B. Glimm, I. Horrocks, C. Lutz, and U. Sattler. Conjunctive query answering for the description logic *shiq*. *J Artif Intell Res*, 31:157–204, 2008.

[18] Patrick Hayes. RDF Semantics. W3C Recommendation, 2004.

[19] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In *Logic for Programming and Automated Reasoning*, pages 161–180. Springer, 1999.

[20] Yannis E. Ioannidis. Query Optimization. *ACM Comput. Surv.*, 28(1):121–123, 1996.

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

[22] David S. Johnson and Anthony C. Klug. Testing Containment of Conjunctive Queries under Functional and Inclusion Dependencies. *J. Comput. Syst. Sci.*, 28(1):167–189, 1984.

[23] Ilianna Kollia, Birte Glimm, and Ian Horrocks. SPARQL Query Answering over OWL Ontologies. In *Proc. 8th ESWC, Heraklion (GR)*, volume 6643 of *LNCS*, pages 382–396, 2011.

[24] Dexter Kozen. Results on the propositional μ-calculus. *Theor. Comp. Sci.*, 27:333–354, 1983.

[25] C. Lutz. The complexity of conjunctive query answering in expressive description logics. *Automated Reasoning*, pages 179–193, 2008.

[26] Sergio Muñoz, Jorge Pérez, and Claudio Gutierrez. Minimal Deductive Systems for RDF. volume 4519 of *LNCS*, pages 53–67, 2007.

[27] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of SPARQL. *ACM Transactions on Database Systems (TODS)*, 34(3):16, 2009.

[28] Axel Polleres. From SPARQL to rules (and back). In *WWW '07*, pages 787–796, 2007.

[29] Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF. W3C Rec., 2008.

[30] Ulrike Sattler and Moshe Y. Vardi. The Hybrid μ-Calculus. In *IJCAR*, pages 76–91, 2001.

[31] Michael Schmidt, Michael Meier, and Georg Lausen. Foundations of SPARQL Query Optimization. In *ICDT '10*, pages 4–33, New York, NY, USA, 2010. ACM.

[32] Yoshinori Tanabe, Koichi Takahashi, and Masami Hagiya. A Decision Procedure for Alternation-Free Modal μ-calculi. In *Advances in Modal Logic*, pages 341–362, 2008.

[33] Yoshinori Tanabe, Koichi Takahashi, Mitsuharu Yamamoto, Akihiko Tozawa, and Masami Hagiya. A Decision Procedure for the Alternation-Free Two-Way Modal μ-calculus. In *TABLEAUX*, pages 277–291, 2005.

Contents

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399