Proceedings, Fifteenth International Conference on
Principles of Knowledge Representation and Reasoning (KR 2016)

A Higher-Order Semantics for Metaquerying in OWL2QL

Maurizio Lenzerini[1], **Lorenzo Lepore**[1], **Antonella Poggi**[2]

[1]Dipartimento di Ingegneria Informatica, Automatica e Gestionale "Antonio Ruberti"

[2]Dipartimento di Scienze Documentarie, Linguistico-Filologiche e Geografiche

Sapienza Università di Roma
lastname@dis.uniroma1.it

Abstract[1]

Inspired by recent work on higher-order Description Logics, we propose HOS, a new semantics for OWL 2 QL ontologies. We then consider SPARQL queries which are legal under the direct semantics entailment regime, we extend them with logical union, existential variables, and unrestricted use of variables so as to express meaningful meta-level queries. We show that both satisfiability checking and answering instance queries with metavariables have the same ABox complexity as under direct semantics.

Introduction

Recent papers point out the need of enriching ontology languages with metamodeling and metaquerying, i.e., features for specifying and reasoning about metaconcepts and metaproperties. Roughly speaking, a metaconcept is a concept whose instances can be themselves concepts, a metaproperty is a relationship between metaconcepts, and a metaquery is a query possibly using metaconcepts and metaproperties, and whose variables may be bound to predicates.

In this work, we focus on OWL 2 QL, one of the profiles of OWL 2 tailored to provide efficient query answering. The issue of metamodeling in OWL has been investigated in several papers. It is known that the semantics of metamodeling of OWL 2 Full leads to undecidability of basic inference problems (Motik 2007). A possible solution to this problem is to enable metamodeling in OWL 2 DL by axiomatizing higher-order features through first order assertions (Glimm, Rudolph, and Völker 2010), but the process involves the use of complex expressions that are not supported by the OWL 2 DL tractable profiles, including OWL 2 QL, and therefore seems hardly applicable in practice. Another possible solution is the stratification of class constructors and axioms to describe metalevels of classes and properties (Pan and Horrocks 2006), but such stratification poses challenges for the modeler, and rules out interesting ontology patterns. Although OWL 2 QL provides syntactic support for metamodeling through OWL 2 punning (by which the same name can be used to denote ontology elements of different categories, such as a class and an individual), we argue that the official semantics of OWL 2, the so-called *Direct Semantics* (DS), treats punning in a way that is not adequate for metamodeling. The reason is simply that proper metamodeling requires that the *same* element plays the role of both individual and class (or, class and relation), while DS sanctions that an individual and a class with the same name are different elements. This is confirmed by the fact that the *Direct Semantics Entailment Regime* (DSER), which is the logic-based semantics of SPARQL 1.1 (or simply SPARQL), when applied to OWL 2 QL, forces queries to obey the so-called *typing constraint*, which rules out the possibility of using the same variable in incompatible positions (for example, in individual and in class position). In this paper we present the logic Hi(OWL 2 QL), whose syntax is the one of OWL 2 QL, and whose semantics, inspired by the ones proposed in (Motik 2007; De Giacomo, Lenzerini, and Rosati 2011), allows every object o in an interpretation to be seen as an individual, a class and a relation. We describe the notion of chase for our logic, that forms the basis for the decidability and complexity results illustrated in the paper. Using the chase, we show that satisfiability and answering instance queries (i.e., queries with ABox atoms, possibly with metavariables) have the same complexity as in OWL 2 QL and in $Hi(DL\text{-}Lite)$ (De Giacomo, Lenzerini, and Rosati 2011).

The logic Hi(OWL 2 QL)

In this section we illustrate syntax and semantics of Hi(OWL 2 QL) and its associated query language.

Syntax. The syntax of Hi(OWL 2 QL) is the same as the one of OWL 2 QL. We express ontologies and queries in the extended functional style syntax (Glimm 2011). A Hi(OWL 2 QL) ontology (simply *ontology* in the following) is a finite set of OWL 2 QL *declaration axioms* and *logical axioms*. Logical axioms are classified into (*i*) *positive TBox axioms*, i.e., SubClassOf, SubObjectPropertyOf, SubDataPropertyOf, ReflexiveObjectProperty, and DataPropertyRange axioms, (*ii*) *negative TBox axioms*, i.e., DisjointClasses, DisjointObjectProperties, DisjointDataProperties, and

[1]Work supported by the EU under the FP7 project "Optique" – grant n. FP7-318338, and by MIUR under the SIR project "MODEUS" – grant n. RBSI14TQHQ.

IrreflexiveObjectPropertyaxioms, and (*iii*) *ABox axioms*, i.e., ClassAssertion, ObjectPropertyAssertion, and DataPropertyAssertionaxioms. Axioms not in the above list can be expressed by appropriate combinations of the ones listed, with the only exception of axioms of the form DifferentIndividuals, which are ruled out in Hi(OWL 2 QL) because their use in conjunctive queries lead to insurmountable computational obstacles (Gutierrez-Basulto et al. 2013).

By virtue of OWL 2 punning, the same entity name can in fact appear in positions of different types. The notion of *position* will be often used in the following. An entity is said to appear in *class position* if it appears in a SubClassOf or DisjointClasses axiom, or in a Declaration(Class) axiom, or in the first position of a ClassAssertion axiom. Similarly, we can define the notions of *object property*, *data property*, *data type*, *individual*, and *value position*.

The *vocabulary* $V_{\mathcal{O}}$ of an Hi(OWL 2 QL) ontology $\mathcal{O}$ is defined as $(V_N^{\mathcal{O}}, V_C^{\mathcal{O}}, V_{OP}^{\mathcal{O}}, V_{DP}^{\mathcal{O}}, V_{DT}^{\mathcal{O}}, \mathtt{L}_{\mathtt{QL}}^{\mathcal{O}})$, where (*i*) $V_N^{\mathcal{O}}$ is the set of IRIs occurring in $\mathcal{O}$ extended with the OWL 2 QL reserved vocabulary, (*ii*) $V_C^{\mathcal{O}}$ (resp., $V_{OP}^{\mathcal{O}}$, $V_{DP}^{\mathcal{O}}$), is the subset of $V_N^{\mathcal{O}}$ consisting of the IRIs that appear in class (resp., object property, data property) positions in $\mathcal{O}$, or are reserved IRIs denoting classes (resp., object properties, data properties) (*iii*) $V_{DT}^{\mathcal{O}}$ is a subset of the datatypes in OWL 2 QL (for each d of them, d^{DT} denotes the set of values of type d), and (*iv*) $\mathtt{L}_{\mathtt{QL}}^{\mathcal{O}}$ is the set of literals occurring in some logical axiom of $\mathcal{O}$ (for each lt of them, lt^{LS} denotes the corresponding data value).

Also, we denote with $Exp^{\mathcal{O}}$ the *finite* set of all *well-formed expressions* that can be built on the basis of $V_{\mathcal{O}}$, defined as $Exp^{\mathcal{O}} = V_N^{\mathcal{O}} \cup Exp_C^{\mathcal{O}} \cup Exp_{OP}^{\mathcal{O}} \cup V_{DP} \cup V_{DT}$,where $Exp_C^{\mathcal{O}} = V_C \cup \{\mathtt{ObjectSomeValuesFrom}(e_1 e_2) \mid e_1 \in Exp_{OP}^{\mathcal{O}}, e_2 \in V_C\} \cup \{\mathtt{DataSomeValuesFrom}(e_1 e_2) \mid e_1 \in V_{DP}, e_2 \in V_{DT}\}$ and $Exp_{OP}^{\mathcal{O}} = V_{OP} \cup \{\mathtt{ObjectInverseOf}(e_1) \mid e_1 \in V_{OP}\}$.

In what follows, with a slight abuse of notation, an ontology $\mathcal{O}$ with TBox axioms $\mathcal{T}^{\mathcal{O}}$ and Abox $\mathcal{A}^{\mathcal{O}}$, will be denoted by $\langle Exp^{\mathcal{O}}, \mathcal{T}^{\mathcal{O}}, \mathcal{A}^{\mathcal{O}} \rangle$.

Semantics. The semantics of Hi(OWL 2 QL), called *HOS*, is based on the notion of interpretation structure, which plays the same role as the "interpretation domain" in classical first-order logic. Specifically, an *interpretation structure* is a tuple $\Sigma = \langle \Delta_o, \Delta_v, \cdot^I, \cdot^E, \cdot^R, \cdot^A, \cdot^T \rangle$ where:

- Δ_o, the *object domain*, and Δ_v, the *value domain* are two disjoint nonempty sets.
- $\cdot^E : \Delta_o \to \mathcal{P}(\Delta_o)$ is a partial function;
- $\cdot^R : \Delta_o \to \mathcal{P}(\Delta_o \times \Delta_o)$ is a partial function;
- $\cdot^A : \Delta_o \to \mathcal{P}(\Delta_o \times \Delta_v)$ is a partial function;
- $\cdot^T : \Delta_o \to \mathcal{P}(\Delta_v)$ is a partial function;
- $\cdot^I : \Delta_o \to \{\mathtt{T}, \mathtt{F}\}$ is a total function s.t. for each $d \in \Delta_o$, if $\cdot^E, \cdot^R, \cdot^A, \cdot^T$ are undefined for d, then $d^I = \mathtt{T}$.

Thus, the interpretation structure is not simply a set, but a mathematical representation of a world made up by elements which have complex inter-relationships, where such inter-relationships are represented by the various functions making up Σ. In particular, each element in the world represented by Σ is in the domain $\Delta = \Delta_o \cup \Delta_v$, and therefore is either an object or a value. Also, $\cdot^E$ is a function that, given a domain object o, either assigns to o its extension, i.e., a set of objects that are its instances, or is undefined for o, in which case o is not regarded as a class in the world represented by Σ. Functions $\cdot^R$, $\cdot^A$, and $\cdot^T$ behave analogously, reflecting whether and how objects are regarded in Σ as relations among objects, relations among objects and values, and sets of values, respectively. Finally, $\cdot^I$ is the boolean function specifying whether an object o is seen as an individual object (if o^I is true), or not (if o^I is false): $\cdot^I$ is crucial for making HOS compatible with DS, based on the idea that the the object domain of an interpretation under DS corresponds to Δ_i, where Δ_i is a nonempty set defined as $\{a \in \Delta_o \mid a^I = \mathtt{T}\}$. An *interpretation* $\mathcal{I}$ for $\mathcal{O}$ is a pair, $\langle \Sigma, \mathcal{I}_o \rangle$, where Σ is an interpretation structure and $\mathcal{I}_o$ is the *interpretation function* for $\mathcal{I}$, i.e., a function that maps every expression in $Exp^{\mathcal{O}}$ into an object in Δ_o, and every literal in $\mathtt{L}_{\mathtt{QL}}$ into a value in Δ_v, according to the set of conditions specified in the following table.

for d in the reserved vocabulary: $(d^{\mathcal{I}_o})^I = \mathtt{F}$;
for $d \in Exp_C^{\mathcal{O}}$: $(d^{\mathcal{I}_o})^E$ is defined, and $(d^{\mathcal{I}_o})^T$ is undefined;
for $d \in Exp_{OP}^{\mathcal{O}}$: $(d^{\mathcal{I}_o})^R$ is defined, and $(d^{\mathcal{I}_o})^A$ is undefined;
for $d \in V_{\mathcal{O}_{DP}}$: $(d^{\mathcal{I}_o})^A$ is defined, and $(d^{\mathcal{I}_o})^R$ is undefined;
for $d \in V_{\mathcal{O}_{DT}}$: $(d^{\mathcal{I}_o})^E$ is undefined, and $(\mathrm{d}^{\mathcal{I}_o})^T = \mathrm{d}^{DT}$
for $lt = (l, d) \in \mathtt{L}_{\mathtt{QL}}$: $lt^{\mathcal{I}_o} = (l, d)^{LS}$;
$((\mathtt{ObjectInverseOf}(e))^{\mathcal{I}_o})^R = ((e^{\mathcal{I}_o})^R)^{-1}$;
$((\mathtt{ObjectSomeValuesFrom}(e_1\ e_2))^{\mathcal{I}_o})^E = \{d_1 \mid \langle d_1, d_2 \rangle \in (e_1^{\mathcal{I}_o})^R, d_2 \in (e_2^{\mathcal{I}_o})^E\}$;
$((\mathtt{DataSomeValuesFrom}(e_1\ e_2))^{\mathcal{I}_o})^E = \{d_1 \mid \langle d_1, d_2 \rangle \in (e_1^{\mathcal{I}_o})^A, d_2 \in (e_2^{\mathcal{I}_o})^T\}$;
$(\mathtt{owl{:}Thing}^{\mathcal{I}_o})^E = \Delta_i$; $(\mathtt{owl{:}Nothing}^{\mathcal{I}_o})^E = \emptyset$; $(\mathtt{rdfs{:}Literal}^{\mathcal{I}_o})^T = \Delta_v$;
$(\mathtt{owl{:}topObjectProperty}^{\mathcal{I}_o})^R = \Delta_i \times \Delta_i$;
$(\mathtt{owl{:}topDataProperty}^{\mathcal{I}_o})^A = \Delta_i \times \Delta_v$;
$(\mathtt{owl{:}bottomObjectProperty}^{\mathcal{I}_o})^R = (\mathtt{owl{:}bottomDataProperty}^{\mathcal{I}_o})^A = \emptyset$;

Note that the extension of e_1, when seen as a class (similarly for object and data properties), is given by the function $\cdot^E$ applied to $e_1^{\mathcal{I}_o}$, where $e_1^{\mathcal{I}_o}$ is the object in Δ_o that $\mathcal{I}$ associates to the expression e_1.

To define the semantics of logical axioms, we resort to the usual notion of satisfaction of an axiom with respect to an interpretation $\mathcal{I}$. Thus, for example, $\mathcal{I} \models \mathtt{SubClassOf}(e_1\ e_2)$ if $(e_1^{\mathcal{I}_o})^E$ and $(e_2^{\mathcal{I}_o})^E$ are defined, and $(e_1^{\mathcal{I}_o})^E \subseteq (e_2^{\mathcal{I}_o})^E$ where e_1, e_2 are expressions.

Two observations are in order: (*i*) the notions of *model* and *satisfiability* of an ontology are the usual ones; (*ii*) checking whether an axiom is logically implied by an ontology can be done in polynomial time using the same technique used for OWL 2 QL.

Query language. We concentrate on *unions of boolean instance conjunctive queries* (simply called *instance queries* in the following). Given a set of variables $\mathcal{V}$ disjoint from $V_{\mathcal{O}}$, a *boolean instance conjunctive query* q over an ontology $\mathcal{O}$ is an expression of the form **ask where** B, where B, called *query body*, is a conjunction of ABox atoms over $V_{\mathcal{O}} \cup \mathcal{V}$. In SPARQL jargon, a query body is a *basic graph pattern*(BGP), i.e., a conjunction of RDF triples involving variables. Since we use the FSS, in our case an atom has the same form of a logical axiom, but its arguments are *terms* which may contain variable symbols from $\mathcal{V}$. We call *metavariable* of an instance query a variable that occurs in class position, object property position, data property position or datatype position in the body, and we say that an atom

is *(meta)ground* if no (meta)variable occurs in it. A query is *ground* (resp. *metaground*) if all its atoms are ground (resp. metaground). Note that the presence of metavariables makes our instance queries suitable for metaquerying.

We provide the semantics of instance queries by defining a new SPARQL entailment regime, called HOS entailment regime, specifying (α) which is the semantics used to interpret the queried ontology, (β) which queries are legal, and (γ) which is the notion of answer to queries (called solution in SPARQL jargon) that we adopt. As for (α), we simply adopt HOS, described above. As for (β), we base our proposal on the class of queries which are legal for the SPARQL DS entailment regime, but extend it by relaxing the typing constraint. As for (γ) we rely on the usual definition of certain answers, where we assign the classical logical meaning to both the existentially quantified variables (meaning that we do not require that they are bound, in every model, to a known ontology element, as in SPARQL), and the union (meaning that the certain answers to a union of conjunctive queries are obtained as the intersection of the answers to the union over all models of the ontology, instead of the union of the certain answers to each conjunctive query in the union, as in SPARQL).

We observe that the semantics of conjunctive instance queries is captured by a suitable extension of the notion of query homomorphism (Chandra and Merlin 1977).

Satisfiability and instance queries

In this section, we study the complexity of the following problems, given an ontology $\mathcal{O} = \langle Exp^{\mathcal{O}}, \mathcal{T}, \mathcal{A} \rangle$. *Satisfiability*: is $\mathcal{O}$ satisfiable? *Query answering*: given an instance query Q, is the certain answer of Q wrt $\mathcal{O}$ true (i.e., $\mathcal{O} \models Q$)? We start by defining the notion of chase in our setting, and then we exploit it to devise a technique for the two above mentioned problems. In what follows, we implicitly refer to an ontology $\mathcal{O}$.

The chase for our logic is similar to the one used in *DL-Lite* (Calvanese et al. 2007): we build a (possibly infinite) structure, starting from an initial structure $Chase^0(\mathcal{O})$, and repeatedly computing $Chase^{j+1}(\mathcal{O})$ from $Chase^j(\mathcal{O})$ by applying suitable rules.

- We make use of two infinite alphabets $\mathcal{S}_o$ and $\mathcal{S}_v$ of variables, both disjoint from $Exp^{\mathcal{O}}$ and $\mathtt{L}_{\mathtt{QL}}^{\mathcal{O}}$, for introducing new unknown individuals and new data values, when needed.
- $Chase^0(\mathcal{O})$ is the set of axioms obtained from $\mathcal{O}$ by adding, for every DataPropertyAssertion axiom stating that a pair (e, lt) belongs to a data property, an axiom DatatypeAssertion(D lt), for every datatype D containing the data value denoted by lt according to $DM_{\mathtt{QL}}$. Formally, such an axiom is not in the syntax of OWL 2 QL, but is needed to make explicit in the chase that lt belongs to the datatype D.
- To compute $Chase^{j+1}(\mathcal{O})$ from $Chase^j(\mathcal{O})$ we apply one of the chase rules, where each rule can be applied only if suitable conditions hold, and are used to enforce the satisfaction of one of the axioms in $\mathcal{O}$. For the lack of space we do not list here all such rules and conditions, but rather present the following example: if ClassAssertion(c_1 e)$\in$ $Chase^j(\mathcal{O})$, SubClassOf(c_1 ObjectSomeValuesFrom(p c)) $\in \mathcal{O}$, and $\forall e'$ such that ObjectPropertyAssertion(p e e') $\in Chase^j(\mathcal{O})$, we have that ClassAssertion(c e')$\notin$ $Chase^j(\mathcal{O})$, then we set $Chase^{j+1}(\mathcal{O})$=$Chase^j(\mathcal{O}) \cup \{$ClassAssertion($c$ s),ObjectPropertyAssertion(p e s)$\}$, where $s \in \mathcal{S}_o$ does not appear in $Chase^j(\mathcal{O})$.
- Finally, we set $Chase(\mathcal{O}) = \bigcup_{i \in \mathbb{N}} Chase^i(\mathcal{O})$.

Note that $Chase(\mathcal{O})$ is a (possibly infinite) set of OWL 2 QL ABox axioms, except for the possible presence of variables in individual and data value positions, and of axioms of the form DatatypeAssertion(e lt) discussed above. Based on $Chase(\mathcal{O})$ we define the so-called *canonical pseudo-interpretation* $Can(\mathcal{O}) = \langle \Sigma^{Can(\mathcal{O})}, {\mathcal{I}_o}^{Can(\mathcal{O})} \rangle$ for $\mathcal{O}$ as follows:

- $\Sigma^{Can(\mathcal{O})} = \langle \Delta_o^{Can(\mathcal{O})}, \Delta_v^{Can(\mathcal{O})}, \cdot^{I_{Can(\mathcal{O})}}, \cdot^{E_{Can(\mathcal{O})}}, \cdot^{R_{Can(\mathcal{O})}}, \cdot^{A_{Can(\mathcal{O})}}, \cdot^{T_{Can(\mathcal{O})}} \rangle$ is an interpretation structure such that (i) $\Delta_o^{Can(\mathcal{O})} = (Exp^{\mathcal{O}} \cup \mathcal{S}_o)$, (ii) $\Delta_v^{Can(\mathcal{O})} = (\mathtt{rdfs:Literal}^{DT} \cup \mathcal{S}_v)$, (iii) if e occurs in individual position in $Chase(\mathcal{O})$, then $e^{I_{Can(\mathcal{O})}} =$ T otherwise $e^{I_{Can(\mathcal{O})}} =$ F, and (iv) the various functions $\cdot^{I_{Can(\mathcal{O})}}, \cdot^{E_{Can(\mathcal{O})}}, \cdot^{R_{Can(\mathcal{O})}}, \cdot^{A_{Can(\mathcal{O})}}, \cdot^{T_{Can(\mathcal{O})}}$ are derived from $Chase(\mathcal{O})$; e.g., if $e \notin Exp_C^{\mathcal{O}}$ then $\cdot^{E_{Can(\mathcal{O})}}$ is undefined for e, otherwise if $e \in V_C^{\mathcal{O}}$, then $e^{E_{Can(\mathcal{O})}} = \{e_1 \mid$ ClassAssertion(e e_1) $\in Chase(\mathcal{O})\}$, and similarly for the other cases.
- ${\mathcal{I}_o}^{Can(\mathcal{O})}$ is such that (*i*) it maps every expression into itself, i.e., for every $e \in Exp^{\mathcal{O}}$, ${e^{\mathcal{I}_o}}^{Can(\mathcal{O})} = e$, and (*ii*) it maps every literal according to $DM_{\mathtt{QL}}$, i.e. for every $lt \in \mathtt{L}_{\mathtt{QL}}^{\mathcal{O}}$, ${lt^{\mathcal{I}_o}}^{Can(\mathcal{O})} = lt^{LS}$.

Note that $Can(\mathcal{O})$ is called a pseudo-interpretation because it may not conform to the definition of interpretation for $\mathcal{O}$. In particular, $d^{T_{Can(\mathcal{O})}}$ may include variables in $\mathcal{S}_v$ (thus not in d^{DT}), and both owl:Nothing$^{E_{Can(\mathcal{O})}}$ and owl:bottomObjectProperty may be nonempty. Also, we exploit the property of Hi(OWL 2 QL) of allowing the functions $\cdot^{E_{Can(\mathcal{O})}}, \cdot^{R_{Can(\mathcal{O})}}, \cdot^{A_{Can(\mathcal{O})}}, \cdot^{T_{Can(\mathcal{O})}}$ to be undefined; in particular, we let these functions be undefined for each newly introduced element from $\mathcal{S}_o$, which therefore represent individuals in $Can(\mathcal{O})$. This reflects the intuition that there is no need to regard the elements introduced in the chase as classes or properties in order to make $Can(\mathcal{O})$ representing the minimal knowledge satisfying $\mathcal{O}$. We observe that this fact is crucial for extending our query answering technique to the case of queries with TBox atoms.

Let us now show that $Can(\mathcal{O})$ plays a crucial role in representing the set of all models of $\mathcal{O}$. If M is an interpretation for $\mathcal{O}$, we say that a function Ψ from $\Delta_o^{Can(\mathcal{O})}$ to Δ_o^M and from $\Delta_v^{Can(\mathcal{O})}$ to Δ_v^M is an *instance-based homomorphism* from $Can(\mathcal{O})$ to M if for every $e \in Exp^{\mathcal{O}}$, $\Psi(e) = {e^{\mathcal{I}_o}}^M$, for every $v \in \mathtt{rdfs:Literal}^{DT}$, $\Psi(v) = v$, and all the *instance-based properties* satisfied by $Can(\mathcal{O})$ are preserved in M under Ψ, e.g., for every $e_1, e_2 \in \Delta_o^{Can(\mathcal{O})}$ such that $e_2 \in e_1^{E_{Can(\mathcal{O})}}$, $\Psi(e_2) \in \Psi(e_1)^{E_M}$.

Proposition 1 *For every model M of $\mathcal{O}$, there exists an*

instance-based homomorphism from $Can(\mathcal{O})$ to M.

Since all extension functions are undefined for the elements of $\mathcal{S}_o$ in $Can(\mathcal{O})$, any query homomorphism from Q to $Can(\mathcal{O})$ will associate to every metavariable of Q an object o such that $o = e^{\mathcal{I}_o{}^{Can(\mathcal{O})}}$ for an $e \in Exp^{\mathcal{O}}$. To capture this fact, let us introduce the notion of metagrounding of an instance query Q: given an n-tuple $\vec{x} = (x_1, x_2, \ldots, x_n)$ of metavariables of Q, and an n-tuple $\vec{t} = (x_1, x_2, \ldots, x_n)$ of expressions in $Exp^{\mathcal{O}}$, a *metagrounding of* Q is obtained from Q by substituting each x_i in $\vec{x}$ with t_i in $\vec{t}$, where, for $i \in \{1, \ldots, n\}$, if x_i occurs in class (object property, data property, datatype) position in Q, then t_i belongs to $Exp^{\mathcal{O}}_C$ (resp. $Exp^{\mathcal{O}}_{OP}$, $V^{\mathcal{O}}_{DP}$, $V^{\mathcal{O}}_{DT}$). We denote by $MG(Q, Exp^{\mathcal{O}})$ the metaground instance query that is the union of all metagroundings of Q w.r.t. $\mathcal{O}$.

Proposition 2 *If Q is an instance query over $\mathcal{O}$, then $Can(\mathcal{O}) \models Q$ if and only if $Can(\mathcal{O}) \models MG(Q, Exp^{\mathcal{O}})$.*

We are now ready to exploit all the above properties to design suitable algorithms for satisfiability and answering instance queries. If $\mathcal{O}_n$ is the set of negative axioms of $\mathcal{O}$, then the *unsatisfiability query* for $\mathcal{O}$ is the metaground instance query of the form

$$Q^u_{\mathcal{O}} = (\bigcup_{\alpha \in \mathcal{O}_n} q_\alpha) \cup (\bigcup_{d_1, d_2 \in V^{\mathcal{O}}_{DT},\ d_1^{DT} \cap d_2^{DT} = \emptyset} q_{d_1, d_2})$$

where q_α has the form:

- **ask** $\{$`ClassAssertion`$(e_1\ \$x)$. `ClassAssertion`$(e_2\ \$x)\}$, if $\alpha =$ `DisjointClasses`$(e_1\ e_2)$, and has an anologous form if $\alpha =$ `DisjointObjectProperties`$(e_1\ e_2)$, or $\alpha =$ `DisjointDataProperties`$(e_1\ e_2)$,
- **ask** $\{$`ObjectPropertyAssertion`$(e_1\ \$x\ \$x)\}$, if $\alpha =$ `IrreflexiveObjectProperty`(e_1)

and q_{d_1,d_2} has the form **ask** $\{$`DatatypeAssertion`$(d_1\ \$x)$. `DatatypeAssertion`$(d_2\ \$x)\}$.

Theorem 3 (i) *$\mathcal{O}$ is unsatisfiable if and only if $Can(\mathcal{O}) \models Q^u_{\mathcal{O}}$,* (ii) *If Q is an instance query over $\mathcal{O}$, then $\mathcal{O} \models Q$ if and only if $Can(\mathcal{O}) \models Q^u_{\mathcal{O}}$ or $Can(\mathcal{O}) \models MG(Q, Exp^{\mathcal{O}})$.*

Proof(sketch). For item (i), the crucial aspect is to show that, if $Can(\mathcal{O}) \not\models Q^u_{\mathcal{O}}$, then it is possible to build a model M^c for $\mathcal{O}$. In a nutshell, M^c is obtained by replacing every v in $\mathcal{S}_v$ with a distinct value $\mu(v)$ such that: *(i)* $\mu_v(v)$ belongs to the value space of every datatype d such that $v \in d^{Can(\mathcal{O})}$, and does not belong to the value space of any other datatype different from `rdfs:Literal`, *(ii)* $\mu_v(v)$ is distinct from all values represented by some literal in $\mathcal{O}$, and *(iii)* $\forall v' \in \mathcal{S}_v, v' \neq v$, $\mu_v(v') \neq \mu_v(v)$. This is possible thanks to the following properties of `OWL 2 QL` datatypes (see also (Savkovic and Calvanese 2012)): *(i)* for every pair of different datatypes D_1 and D_2, $D_1^{DT} \setminus D_2^{DT}$ is an infinite set, and *(ii)* $\Delta_v \setminus U$ is an infinite set, where $U = \bigcup_{d \in \mathtt{D}'} d^{DT}$. For item (ii), one direction exploits M^c as a counterexample, and the other direction easily follows from Proposition 2. □

It remains to describe how we check whether $Can(\mathcal{O}) \models Q$, where Q is ground. Inspired by the technique described in (De Giacomo, Lenzerini, and Rosati 2011), we use the well-known rewriting approach, typical of the *DL-Lite* family (Calvanese et al. 2007), thus opening up the possibility to use off-the-shelf `OWL 2 QL` reasoners, such as Mastro (Calvanese et al. 2011) or Ontop (Rodriguez-Muro and Calvanese 2012). As a consequence, satisfiability and answering instance queries are in AC^0 w.r.t. ABox complexity and in PTIME w.r.t. ontology complexity, and answering instance queries is also NP-complete w.r.t. combined complexity.

Conclusion

We plan to continue our work along several directions. The most important aspect is studying query answering under HOS for queries that possibly contain TBox atoms too. Also, we aim at devising extensions of both the ontology and the query languages, that increase metamodeling and metaquerying capabilities.

References

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

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; Poggi, A.; Rodriguez-Muro, M.; Rosati, R.; Ruzzi, M.; and Savo, D. F. 2011. The Mastro system for ontology-based data access. *Semantic Web J.* 2(1).

Chandra, A. K., and Merlin, P. M. 1977. Optimal implementation of conjunctive queries in relational data bases. In *Proc. of STOC 1977*, 77–90.

De Giacomo, G.; Lenzerini, M.; and Rosati, R. 2011. Higher-order description logics for domain metamodeling. In *Proc. of AAAI 2011*.

Glimm, B.; Rudolph, S.; and Völker, J. 2010. Integrated metamodeling and diagnosis in OWL 2. In *Proc. of ISWC 2010*, volume 6496 of *LNCS*.

Glimm, B. 2011. Using SPARQL with RDFS and OWL entailment. In *RW-11*. 137–201.

Motik, B. 2007. On the properties of metamodeling in OWL. *J. of Logic and Computation* 17(4):617–637.

Pan, J. Z., and Horrocks, I. 2006. OWL FA: a metamodeling extension of OWL DL. In *Proc. of WWW 2006*.

Rodriguez-Muro, M., and Calvanese, D. 2012. Quest, an OWL 2 QL reasoner for ontology-based data access. In *Proc. of OWLED 2012*, volume 849 of *CEUR*.

Savkovic, O., and Calvanese, D. 2012. Introducing datatypes in *DL-Lite*. In *Proc. of ECAI 2012*.