Answering Metaqueries Over Hi(OWL 2 QL) Ontologies

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

Hi(OWL 2 QL) is a new ontology language with the OWL 2 QL syntax and a specific semantics designed to support metamodeling and metaquerying. In this paper we investigate the problem of answering metaqueries in Hi(OWL 2 QL), which are unions of conjunctive queries with both ABox and TBox atoms. We first focus on a specific class of ontologies, called TBox-complete, where there is no uncertainty about TBox axioms, and show that query answering in this case has the same complexity (both data and combined) as in OWL 2 QL. We then move to general ontologies and show that answering metaqueries is coNP-complete with respect to ontology complexity, Π_2^p-complete with respect to combined complexity, and remains AC^0 with respect to ABox complexity. Finally, we present an optimized query answering algorithm that can be used for TBox-complete ontologies.

1 Introduction

The interest in extending ontology languages with metamodeling (i.e., metaclasses and metaproperties) and metaquerying is growing considerably [de Carvalho *et al.*, 2015]. Roughly speaking, a metaclass is a class (unary predicate) whose instances can be themselves classes, a metaproperty is a relationship (binary predicate) between metaclasses, and a metaquery is a query where variables can appear in both predicate and object position. As pointed out, for example in [Guizzardi *et al.*, 2015], these features are important in many scenarios, such as modeling provenance of data, biological taxonomy (where the notion of species is crucial), product manufacturing (where product types have properties to be described and retrieved), or, in general, in ontology-based data access and integration, where data sources often contain not only objects, but also information about the classes which objects belong to, and such classes are to be modeled also as individuals in the ontology.

Unfortunately, the treatment of the above features in OWL 2, which is the de-facto ontology language, is unsatisfactory. As a simple example, suppose we want to retrieve all individual animals which live in Central Park Zoo (CPZ), and are instances of a class corresponding to an endangered species (ES). The natural formulation of the corresponding metaquery is $Q = \{ (y) \mid \exists x\ \mathtt{ES}(x) \wedge x(y) \wedge \mathtt{lives}(y, \mathtt{CPZ}) \}$. However, this query is not legal under the OWL 2 Direct Semantics (DS), which is the formal semantics adopted for interpreting the logical theories corresponding to OWL 2 ontologies. In particular, Q violates the so-called *typing constraint*, which, intuitively, rules out the possibility of using the same variable in incompatible positions (for example, in individual and in class position). The reason for such a limitation is that under DS no element can simultaneously be an individual and a class, which severely hampers the possibility of using metamodeling and metaquerying in OWL 2. To remedy this situation new semantics (e.g., [Motik, 2007; De Giacomo *et al.*, 2011; Lenzerini *et al.*, 2016]) have been proposed, mostly based on HiLog [Chen *et al.*, 1993], which is a logic with a higher order syntax based on a Henkin-style semantics. We refer, in particular, to the semantics presented in [Lenzerini *et al.*, 2016], specifically defined for Hi(OWL 2 QL) but generalizable to all Description Logics. In this semantics, each object in an interpretation can simultaneuosly be an individual object, a class, an object property (or, relationship), a data property (or, attribute), and a datatype, and therefore fully supports the kinds of metaqueries discussed above. Moreover, since classes and properties are first-order citizens, this semantics naturally pushes towards *full metaqueries*, i.e., queries where ABox atoms, i.e., instance assertions, and TBox atoms, such as ISA (e.g., every eagle is a bird, $\mathtt{Eagle} \sqsubseteq \mathtt{Bird}$) or disjointness (e.g., every eagle is not a condor) assertions coexist. An example of such queries is the one obtained from Q by requiring that the endangered species is a subclass of Bird: $\{ (y) \mid \exists x\ \mathtt{ES}(x) \wedge x(y) \wedge x \sqsubseteq \mathtt{Bird} \wedge \mathtt{lives}(y, \mathtt{CPZ}) \}$.

The goal of our work is to study algorithms for, and complexity of full metaquerying. We are especially interested in analyzing the case of OWL 2 QL, the tractable fragment of OWL 2, that is the most popular language used in the context of ontology-based data access, and checking whether the nice computational characteristics of the logic are kept when we move to full metaquering.

Several papers are relevant for our work. [Motik, 2007] studies the complexity of the inference problem in OWL Full, and suggests a HiLog-style semantics, further investigated in [De Giacomo *et al.*, 2011], which presents an algorithm based on the metagrounding technique for answering a specific class of queries, called guarded. [Glimm *et*

al., 2010] and [Pan and Horrocks, 2006] present techniques for metamodeling by axiomatizing higher-order knowledge into first order assertions, and introducing the stratification of class constructors and axioms, respectively. Other papers with focus on ontology languages with metamodeling capabilities are [Jekjantuk *et al.*, 2010; Homola *et al.*, 2013; 2014; Kubincová *et al.*, 2015], but none of them face the problem of query answering in such context. Relevant to our work are recent papers aiming at devising efficient techniques to answer SPARQL queries posed over OWL 2 QL ontologies [Arenas *et al.*, 2014; Kontchakov *et al.*, 2014; Kollia and Glimm, 2013]. However, such papers concentrate on DS, and therefore do not aim at the full power of metamodeling and metaquerying. The conclusion is that none of the above papers provide results for full metaquerying. Indeed, to our knowledge, it is unknown whether the problem of answering full metaqueries is decidable, even for the case of lighweight ontology languages such as OWL 2 QL.

In this paper we present several results about answering full metaqueries in Hi(OWL 2 QL).

- We identify a class of ontologies satisfying a certain completeness condition (TBox-completeness), and show that answering full metaqueries over ontologies within such a class has the same complexity as answering queries with ABox atoms only, over OWL 2 QL ontologies under DS.
- We show that query answering over general ontologies is decidable. In particular, we present a query answering algorithm that is in AC^0 w.r.t. ABox complexity, in coNP w.r.t. ontology complexity and in Π_2^p w.r.t. combined complexity. This is the first decidability (actually, tractability w.r.t. ABox complexity) result for full metaqueries over OWL 2.
- We provide complexity lower bounds for answering conjunctive metaqueries, showing that the problem is coNP-complete w.r.t. ontology complexity, and Π_2^p-complete w.r.t. combined complexity, thus sharpening the intractability results in [De Giacomo *et al.*, 2011] for queries in the presence of union.
- Motivated by the fact that, in practice, TBox-complete ontologies may be automatically obtained by applying specific ontology design methodologies, we focus on them and present an optimized algorithm for answering queries over TBox-complete ontologies. Experiments show that the optimized algorithm outperforms the one based on blind metagrounding.

2 Preliminaries

In this section we briefly recall the Hi(OWL 2 QL) ontology, the metaquery language, and the metaquery answering technique based on metagrounding. For details, please refer to [Lenzerini *et al.*, 2016].

Ontology language The syntax of Hi(OWL 2 QL) is the same as the one of OWL 2 QL. In this work, we express ontologies and queries in the extended functional style syntax [Glimm, 2011]. Axioms are classified into (*i*) *positive TBox axioms*, i.e., SubClassOf, SubObjectPropertyOf, SubDataPropertyOf, ReflexiveObjectProperty, and DataPropertyRange, (*ii*) *negative TBox axioms*, i.e., DisjointClasses, DisjointObjectProperties, DisjointDataProperties, and IrreflexiveObjectProperty, and (*iii*) *ABox axioms*, i.e., ClassAssertion, ObjectPropertyAssertion, and DataPropertyAssertion.

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*) $V_C^{\mathcal{O}}$ is the subset of $V_N^{\mathcal{O}}$ consisting of the IRIs appearing in class positions in $\mathcal{O}$ (i.e., in a SubClassOf or DisjointClasses axiom, or in a Declaration(Class) axiom, or in the first position of a ClassAssertion axiom), or are reserved IRIs denoting classes, (*iii*) $V_{OP}^{\mathcal{O}}$, $V_{DP}^{\mathcal{O}}$, $V_{DT}^{\mathcal{O}}$ are defined similarly for object properties, data properties, and datatypes, and (*iv*) $\mathtt{L}_{\mathtt{QL}}^{\mathcal{O}}$ is the set of literals occurring in some logical axiom of $\mathcal{O}$. Note that, by virtue of OWL 2 punning, the same entity name can in fact appear in positions of different types (e.g., as an argument of a SubClassOf axiom and as the argument of an ReflexiveObjectProperty axiom). Thus, $V_C^{\mathcal{O}}, V_{OP}^{\mathcal{O}}$, $V_{DP}^{\mathcal{O}}$ and $V_{DT}^{\mathcal{O}}$ may not be disjoint. Also, we denote with $Exp^{\mathcal{O}}$ the *finite* set of all *well-formed expressions* that denote elements of the ontology, and can be built on the basis of $V_{\mathcal{O}}$. For example, if e_1, e_2 are well-formed expressions, then $\mathtt{ObjectSomeValuesFrom}(e_1 e_2)$ is a well-formed expression too. Note that $Exp^{\mathcal{O}}$ is actually partitioned into class expressions ($Exp_C^{\mathcal{O}}$), object property expressions ($Exp_{OP}^{\mathcal{O}}$), $V_{DP}^{\mathcal{O}}$, and $V_{DT}^{\mathcal{O}}$. For example, $\mathtt{ObjectSomeValuesFrom}(e_1 e_2)$ is a class expression in $Exp_C^{\mathcal{O}}$.

The semantics of Hi(OWL 2 QL), called Higher-Order Semantics (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:

- the *object domain* Δ_o and the *value domain* Δ_v are two disjoint non-empty 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 such that for each $d \in \Delta_o$, if $\cdot^E, \cdot^R, \cdot^A, \cdot^T$ are all undefined for d, then $d^I = \mathtt{T}$.

Note that domain objects are polymorphic: each of them can simultaneuosly be an individual object (this is the case where function $\cdot^I$ is T), a class ($\cdot^E$ is defined), an object property ($\cdot^R$ is defined), a data property ($\cdot^A$ is defined), and a datatype ($\cdot^T$ is defined). A crucial aspect of such interpretation structure, which distinguishes it from the one proposed in [De Giacomo *et al.*, 2011], is that the functions $\cdot^E$, $\cdot^R$, $\cdot^A$, and $\cdot^T$ are partial, so that, for example, an object can be both an individual and an object property without being a class. The importance of this feature for metaquerying will be clear later in the paper. An *interpretation* J for $\mathcal{O}$ is a pair, $\langle \Sigma^J, {\mathcal{I}_o}^J \rangle$, where $\Sigma^J = \langle \Delta_o^J, \Delta_v^J, \cdot^{I_J}, \cdot^{E_J}, \cdot^{R_J}, \cdot^{A_J}, \cdot^{T_J} \rangle$ is an interpretation structure and ${\mathcal{I}_o}^J$ is the *interpretation function* for J, i.e., a function that maps every expression in $Exp^{\mathcal{O}}$ into an object in Δ_o^J, and every literal in $\mathtt{L}_{\mathtt{QL}}$ into a value in Δ_v^J, according to a

set of suitable conditions. The semantics of logical axioms is based on the usual notion of *satisfaction of an axiom* with respect to an interpretation J. Also, the notions of *model* and *satisfiability* of an ontology are the usual ones. Finally, if α is an axiom, we say that α is *logically implied* by an ontology $\mathcal{O}$ if it is satisfied in every model of $\mathcal{O}$, while α is *violated* by $\mathcal{O}$ if $\mathcal{O} \cup \{\alpha\}$ is unsatisfiable.

Query language. As for the query language, we concentrate on unions of boolean conjunctive queries (simply called *queries* in the following). A boolean conjunctive query q over an ontology $\mathcal{O}$ is an expression of the form **ask where** B, where B, called *query body*[1], is a conjunction of atoms over $V_{\mathcal{O}}$. Thus, an atom has the same form of a logical axiom and involves, besides symbols in $V_{\mathcal{O}}$, variables in an alphabet $\mathcal{V}$ disjoint from $V_{\mathcal{O}}$. We call *metavariable* of a query a variable that occurs in predicate position (i.e., class position, object property position, dataproperty position or datatype position) in the query, 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), and is an *instance query* if it includes ABox atoms only.

The semantics of queries resorts to (α) interpreting the queried ontology according to HOS, (β) extending the class of legal queries for the SPARQL DS entailment regime, by relaxing the so-called *typing constraint*, and (γ) defining the answers to a query as the usual notion of certain answers, where both union, and existentially quantified variables are assigned the classical logical meaning.

Query answering through metagrounding. Given a query Q, an n-tuple $\vec{x}$ of variables of Q, and an n-tuple $\vec{t}$ of expressions and literals, the *$\vec{x}$-instantiation of Q with $\vec{t}$*, denoted $\text{INST}(Q, \vec{x} \leftarrow \vec{t})$, is the query obtained from Q by first substituting each x_i in $\vec{x}$ with t_i in $\vec{t}$, for $i \in \{1, \ldots, n\}$, and then replacing non Hi (OWL 2 QL) atoms, if any, with equivalent Hi (OWL 2 QL) atoms involving new variables[2].

A *metagrounding* of Q with respect to $\mathcal{O}$ is an $\vec{x}$-instantiation of Q with any n-tuple $\vec{t}$, where $x_1, x_2, \ldots, x_n$ are the metavariables in Q, and for every $i \in \{1, \ldots, n\}$, if x_i occurs in class (object property, data property, datatype) position in Q, then t_i is any expression in $Exp^{\mathcal{O}}_{C}$ (resp. $Exp^{\mathcal{O}}_{OP}$, $V^{\mathcal{O}}_{DP}$, $V^{\mathcal{O}}_{DT}$) that occurs in $\mathcal{O}$. We denote by $MG(Q, Exp^{\mathcal{O}})$ the metaground query that is the union of all metagroundings of Q w.r.t. $\mathcal{O}$. As reported in [Lenzerini *et al.*, 2016], it can be shown that answering a union of conjunctive instance queries Q over an Hi (OWL 2 QL) $\mathcal{O}$ can be reduced to answering $MG(Q, Exp^{\mathcal{O}})$, for which one can use the well-known rewriting approach, typical of the *DL-Lite* family (see, for instance, [Calvanese *et al.*, 2007]). In other words, instance metaqueries in Hi (OWL 2 QL) can be dealt with the so-called metagrounding technique. A similar observation holds for the problem of checking satisfiability of a Hi (OWL 2 QL) ontology.

In the next sections, we study the complexity of *query answering*: given a Hi (OWL 2 QL) satisfiable ontology $\mathcal{O}$, and a full metaquery Q over $\mathcal{O}$, is the certain answer of Q w.r.t. $\mathcal{O}$ true (i.e., $\mathcal{O} \models Q$)? In particular, we refer to different assumptions: (*i*) $\mathcal{T}, Exp^{\mathcal{O}}$ and Q are fixed (*ABox complexity*), (*ii*) Q is fixed (*ontology complexity*), (*iii*) $\mathcal{A}$ and Q are fixed (*TBox complexity*), and (*iv*) no component of the input is fixed (*combined complexity*).

We finally point out that, from now on, we implicitly assume to deal with a satisfiable Hi (OWL 2 QL) ontology $\mathcal{O}$ and a query Q over $\mathcal{O}$, and we assume that $\mathcal{O}$ does not contain any data property[3].

3 TBox-complete ontologies

We start with an example highlighting a source of complexity in answering metaqueries over general ontologies.

Example 1 Let $\mathcal{O}_1$ be the following ontology:

ClassAssertion(:B :F), ClassAssertion(:C :F),
DisjointClasses(:A :C),
ObjectPropertyAssertion(:R :E :E),
ObjectPropertyAssertion(:R :C :A),
ObjectPropertyAssertion(:R :B :C).

and let Q be the query: **ask** {DisjointClasses(:A $x). ClassAssertion(:B $y). ObjectPropertyAssertion(:R $x $z).ClassAssertion($z $y)} .

It is not hard to see that there exists no substitution μ of the metavariables of Q_1 for which $\mu(Q_1)$ is true in every model of $\mathcal{O}_1$. So, metagrounding does not suffice in this case. On the other hand, suppose to partition the sets of models of $\mathcal{O}_1$ into the set $\mathcal{M}_1$ of models in which :A and :B are interpreted as non-disjoint classes, and the set $\mathcal{M}_2$ of models in which :A and :B are interpreted as disjoint classes. Then, consider the metagroudings of Q_1 with the following substitutions: $\mu_1 : \{x \leftarrow$:C, $z \leftarrow$:A$\}$, and $\mu_2 : \{x \leftarrow$:B, $z \leftarrow$:C$\}$. Clearly, $\mu_1(Q_1)$ is satisfied in every model of $\mathcal{M}_1$, while $\mu_2(Q_1)$ is not. On the other hand, $\mu_2(Q_1)$ is satisfied in every model of $\mathcal{M}_2$, while $\mu_1(Q_1)$ is not. Therefore, for every every model there exists a substitution of variables that makes Q_1 evaluate to true, which proves that $\mathcal{O}_1 \models Q_1$. □

The example shows that, in the presence of TBox atoms in the query, query answering requires to reason by cases, originating from negative TBox axioms which are neither logically implied by $\mathcal{O}$ nor violated by $\mathcal{O}$. Motivated by the latter observation, in this section we investigate query answering for a class of ontologies, called TBox-complete, for which such a possibility does not exist.

Let $\mathcal{U}^{\mathcal{O}}$ denote the set of negative axioms that can be constructed over $V^{\mathcal{O}}_{N}$. If α is an axiom in $\mathcal{U}^{\mathcal{O}}$, α is said to be *certain* if either $\mathcal{O} \models \alpha$, or α is violated by $\mathcal{O}$; otherwise, we say that α is *uncertain*. Thus, if every negative axiom in $\mathcal{U}^{\mathcal{O}}$ is certain, then we say that $\mathcal{O}$ is *TBox-complete*. Note that every unsatisfiable ontology is trivially TBox-complete.

Example 2 The ontology $\mathcal{O}_1$ presented in Example 1 is not TBox-complete. Indeed, both the axioms DisjointClasses(:A :B) and DisjointClasses(:B :A) are neither logically implied nor violated by $\mathcal{O}_1$. □

[1] In SPARQL jargon, a query body is a basic graph pattern.

[2] E.g., ClassAssertion(ObjectSomeValuesFrom($e_1 e_2$)e_3) is replaced by the conjunction of ObjectPropertyAssertion($e_1 e_3 w$) and ClassAssertion($e_2 w$), where w is a new variable.

[3] The whole approach can be extended to capture data properties too.

The conceptual tool we use for addressing query answering in a TBox-complete ontology $\mathcal{O}$ is a variant of the chase used for *DL-Lite* [Calvanese *et al.*, 2007]. We build a (possibly infinite) structure, starting from an initial structure $Chase^0(\mathcal{O})$, and then repeatedly computing $Chase^{j+1}(\mathcal{O})$ from $Chase^j(\mathcal{O})$ by applying suitable rules. In doing so, we make use of an infinite alphabet $\mathcal{S}_o$ of variables, disjoint from $Exp^{\mathcal{O}}$ and $\mathtt{L}^{\mathcal{O}}_{\mathtt{QL}}$, for introducing new unknown individuals, when needed. The initial structure $Chase^0(\mathcal{O})$ is defined as the set of axioms obtained from $\mathcal{O}$ by adding, for every class C that is nonempty in $\mathcal{O}$ (i.e., such that the TBox does not logically imply $\mathtt{DisjointClasses}(C\ C)$), an axiom asserting that C contains a new unknown individual in $\mathcal{S}_o$, that is specific for C, and doing a similar addition for non-empty object properties. Intuitively, the new individual for C will be an instance of only those classes that are superclasses of C in $\mathcal{O}$ (similarly, for object properties), and therefore will be used in the chase to falsify all subset and disjointness relations that are not implied by $\mathcal{O}$. Then, 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. For example, if $\mathtt{ClassAssertion}(c_1\ e) \in Chase^j(\mathcal{O})$, $\mathtt{SubClassOf}(c_1\ \mathtt{ObjectSomeValuesFrom}(p\ c)) \in \mathcal{O}$, and $\forall e'$ such that $\mathtt{ObjectPropertyAssertion}(p\ e\ e') \in Chase^j(\mathcal{O})$, we have that $\mathtt{ClassAssertion}(c\ e') \notin Chase^j(\mathcal{O})$, then we set $Chase^{j+1}(\mathcal{O}) = Chase^j(\mathcal{O}) \cup \{\mathtt{ClassAssertion}(c\ s), \mathtt{ObjectPropertyAssertion}(p\ e\ s)\}$, 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 ABox axioms.

Based on $Chase(\mathcal{O})$, we define the model (called *canonical*) $Can(\mathcal{O})$ of $\mathcal{O}$ as follows:

- ${\Sigma^{Can(\mathcal{O})}}$is an interpretation structure such that (*i*) $\Delta_o^{Can(\mathcal{O})} = (Exp^{\mathcal{O}} \cup \mathcal{S}_o)$, (*ii*) $\Delta_v^{Can(\mathcal{O})} = \mathtt{D}$, where $\mathtt{D}$ is the set data values admitted in $\mathtt{OWL\,2\,QL}$, (*iii*) if e occurs in individual position in $Chase(\mathcal{O})$, then $e^{I_{Can(\mathcal{O})}} = \mathtt{T}$ otherwise $e^{I_{Can(\mathcal{O})}} = \mathtt{F}$, 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 $e \notin Exp^{\mathcal{O}}_C$ then $\cdot^{E_{Can(\mathcal{O})}}$ is undefined for e; otherwise, if $e \in V^{\mathcal{O}}_C$, then $e^{E_{Can(\mathcal{O})}} = \{e_1 \mid \mathtt{ClassAssertion}(e\ e_1) \in Chase(\mathcal{O})\}$, and similarly for the other cases of $e \in Exp^{\mathcal{O}}_C$.
- ${\mathcal{I}_o}^{Can(\mathcal{O})}$ 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 the $\mathtt{OWL\,2\,QL}$ datatype map.

Let us now show that $Can(\mathcal{O})$ plays a crucial role in representing the set of all models of $\mathcal{O}$. To this aim, we need to introduce the notion of *extended homomorphism*. If M is a model of $\mathcal{O}$, 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 extended homomorphism from $Can(\mathcal{O})$ to M if for every $e \in Exp^{\mathcal{O}}$, $\Psi(e) = {e^{\mathcal{I}_o}}^M$, for every $v \in \mathtt{D}$, $\Psi(v) = v$, and all the semantic properties that are expressible in terms of $\mathtt{Hi\,(OWL\,2\,QL)}$ axioms and hold in $Can(\mathcal{O})$, are preserved in M under Ψ. For example, for every $e_1, e_2 \in \Delta_o^{Can(\mathcal{O})}$ such that $e_1^{E_{Can(\mathcal{O})}} \subseteq e_2^{E_{Can(\mathcal{O})}}$, $\Psi(e_1)^{E_M} \subseteq \Psi(e_2)^{E_M}$.

Proposition 3 *If $\mathcal{O}$ is TBox-complete, then for every model M of $\mathcal{O}$, there exists an extended homomorphism from $Can(\mathcal{O})$ to M.*

Two aspects are essential for the proof of the above theorem. First, the functions $\cdot^{E_{Can(\mathcal{O})}}, \cdot^{R_{Can(\mathcal{O})}}, \cdot^{A_{Can(\mathcal{O})}}, \cdot^{T_{Can(\mathcal{O})}}$ are undefined for each newly introduced unknown individual taken from $\mathcal{S}_o$. This is possible with HOS because, unlike in other semantics, such functions are partial. Indeed, when the above functions are total, each newly introduced unknown individual is a class, that either (*i*) is empty, in which case it represents a subset of every class in the ontology, or (*ii*) has at least one instance. Both cases prevent the existence of an extended homomorphism from $Can(\mathcal{O})$ to every model of the ontology. Second, it is crucial that for every class C that is non-empty in $\mathcal{O}$, we have added a new instance for C in $Chase(\mathcal{O})$ (all similar additions for non-empty object properties are also essential). Such new instances are harmless for ABox atoms, but are crucial for TBox axioms, because they prevent TBox axioms that are not implied by $\mathcal{O}$ to be satisfied by $Can(\mathcal{O})$. Based on Proposition 3, one can show the following.

Proposition 4 *If $\mathcal{O}$ is TBox-complete, then $\mathcal{O} \models Q$ if and only if $Can(\mathcal{O}) \models MG(Q, Exp^{\mathcal{O}})$.*

We finally address the problem of checking whether $Can(\mathcal{O}) \models MG(Q, Exp^{\mathcal{O}})$. First notice that $Can(\mathcal{O}) \models MG(Q, Exp^{\mathcal{O}})$ if and only if $Can(\mathcal{O}) \models Q'$ for some ground conjunctive query Q' which is a disjunct of $MG(Q, Exp^{\mathcal{O}})$. So, it remains to illustrate a method for checking whether $Can(\mathcal{O})$ satisfies a metaground conjunctive query.

Proposition 5 *If $\mathcal{O}$ is TBox-complete and Q is a metaground conjunctive query, then $Can(\mathcal{O}) \models Q$ if and only if $\mathcal{O} \models int(Q)$ and $Can(\mathcal{O}) \models ext(Q)$, where $ext(Q)$ and $int(Q)$ denote the conjunctions of ABox and TBox atoms of Q, respectively.*

Propositions 4 and 5 suggest an algorithm, based on blind metagrounding, that we call NAIVE. Given a TBox-complete ontology $\mathcal{O}$, the set $Exp^{\mathcal{O}}$, and a query Q, NAIVE computes $MG(Q, Exp^{\mathcal{O}})$ and answers "true" if and only if for some conjunctive query Q' that is a disjunct of $MG(Q, Exp^{\mathcal{O}})$, it holds that: (*i*) all the atoms in $int(Q')$ are logically implied by $\mathcal{O}$, and (*ii*) $\mathcal{O} \models ext(Q')$. Hence, query answering over TBox-complete ontologies can be done by using any off-the-shelf $\mathtt{OWL\,2\,QL}$ reasoner. The following theorem characterizes the complexity of the problem.

Theorem 6 *Query answering over TBox-complete ontologies is in AC^0 w.r.t. ABox complexity,* PTIME *w.r.t. ontology complexity, and NP-complete w.r.t. combined complexity.*

4 General ontologies

In this section, we address query answering over general ontologies, by first providing a non-deterministic algorithm, and then studying the complexity of the problem.

Let us first introduce the notions of violation set and ontology completion, both crucial to characterize the sets of models of $\mathcal{O}$ under HOS. If α is a negative axiom in $\mathcal{U}^{\mathcal{O}}$,

we call *violation set* of α w.r.t. $\mathcal{O}$, a minimal set of ABox axioms $\mathcal{V}_{\alpha,\mathcal{O}}$, whose entities in predicate position belong to $V_N^{\mathcal{O}}$, and entities in individual position are IRIs not in $V_N^{\mathcal{O}}$, specific for α, and such that $\alpha \cup \mathcal{V}_{\alpha,\mathcal{O}}$ is unsatisfiable. For example, a violation set of $\alpha_1 = \texttt{DisjointClasses}(C_1\ C_2)$ is the set $\{\texttt{ClassAssertion}(C_1\ s_{\alpha_1}), \texttt{ClassAssertion}(C_2\ s_{\alpha_1})\}$ where s_{α_1} is an IRI not in $V_N^{\mathcal{O}}$. Also, if σ is a subset of $\mathcal{U}^{\mathcal{O}}$, the *$\sigma$-completion of* $\mathcal{O}$, denoted $\mathcal{O}^\sigma$, is the ontology $\mathcal{O} \cup \sigma \cup \mathcal{C}^{\mathcal{U}^{\mathcal{O}}\setminus\sigma}$, where $\mathcal{C}^{\mathcal{U}^{\mathcal{O}}\setminus\sigma}$ is the union of the violation sets of axioms in $\mathcal{U}^{\mathcal{O}}$ that are not in σ. Intuitively, $\mathcal{O}^\sigma$ is obtained from $\mathcal{O}$ by adding all axioms in σ and suitable axioms in such a way that all axioms in $\mathcal{U}^{\mathcal{O}}$ but not in σ are violated. Note that $\mathcal{O}^\sigma$ may not be satisfiable, and is TBox-complete. The following theorem shows the importance of ontology completion.

Theorem 7 *$\mathcal{O} \not\models Q$ if and only if there exists a subset σ of $\mathcal{U}^{\mathcal{O}}$ such that $\mathcal{O}^\sigma \not\models Q$.*

From Theorem 7 it is easy to derive a nondeterministic algorithm for checking whether the certain answer to Q is false: non deterministically guess a subset σ of $\mathcal{U}^{\mathcal{O}}$, and then check whether $\mathcal{O}^\sigma \not\models Q$. The following characterizes query answering complexity over general ontologies.

Theorem 8 *Query answering is coNP-complete w.r.t. TBox complexity, Π_2^p-complete w.r.t. combined complexity, and AC^0 w.r.t. ABox complexity.*

Proof (Sketch) The ontology and the combined complexity upper bounds can be easily derived by considering the above nondeterministic algorithm and Theorem 6. As for the ABox complexity, one can show that given an ontology $\mathcal{O}$, for every query Q, answering Q over $\mathcal{O}$ can be reduced to the problem of answering a first-order query Q' over an ontology $\mathcal{O}'$, where the size of Q' does not depend on the ABox and the size of $\mathcal{O}'$ is linear in the size of the ABox. In a nutshell, $\mathcal{O}'$ is built as follows. First, for every subset σ of $\mathcal{U}^{\mathcal{O}}$, we compute the ontology $(\mathcal{O}^\sigma)'$ by substituting in $\mathcal{O}^\sigma$ every occurrence of an entity E that is not an individual with E_σ, and by adding the axiom $\texttt{SubClassOf}(E\ E_\sigma)$ if E is a class, and $\texttt{SubObjectPropertyOf}(E\ E_\sigma)$ if E is an object property. Then, $\mathcal{O}'$ is defined as the union of the ABox of $\mathcal{O}$ with $(\mathcal{O}^\sigma)'$, for every σ. As for Q', we define, for every σ, the query $(Q_\sigma)'$ that is the rewriting of $Q_\sigma \cup Q_{unsat,\sigma}$ w.r.t. $\mathcal{O}'$, where Q_σ is obtained by renaming the entities occurring in Q accordingly to $(\mathcal{O}^\sigma)'$, and $Q_{unsat,\sigma}$ is the query that checks whether $(\mathcal{O}^\sigma)'$ is unsatisfiable (such query can be easily derived from the TBox of $(\mathcal{O}^\sigma)'$). We then obtain Q' as the union of all $(Q_\sigma)'$. It can be shown that $(Q_\sigma)'$ is a first-order query such that $\mathcal{O} \not\models Q$ iff $(Q_\sigma)'$ is false when evaluated over the ABox of $\mathcal{O}$.

As for the TBox complexity lower bound, the proof is by reduction from 3-SAT. Given a 3-CNF formula F, we define an ontology $\mathcal{O}_F$ with an individual g, one class c_i for each clause c_i, classes $d_j, p_j, \overline{p_j}$ for each letter p_j, classes B, D, object properties $R_1, \ldots, R_5$. The axioms of $\mathcal{O}_F$ are $D(g), \texttt{DisjointClasses}(B\ D)$, plus suitable axioms representing the literals in the various clauses (e.g., a clause c_1 with literals $p_2, \neg p_4, p_5$, is represented by $R_1(c_1, p_2), R_2(c_1, \overline{p_4}), R_3(c_1, p_5)$ plus $R_4(c_1, D), R_5(c_1, D)$), and suitable axioms representing complementary literals (e.g., $R_4(d_2, p_2), R_5(d_2, \overline{p_2})$ plus $R_1(d_2, B), R_2(d_2, B), R_3(d_2, B)$ for literals on the letter p_2). The idea is that there is a one-to-one correspondence between each model M of F and the models of $\mathcal{O}_F$ where each class p_i (or $\overline{p_i}$) is disjoint from D iff p_i (or $\overline{p_i}$) is false in M. We then define a query Q_F simply asking whether there exists a class c_i such that all classes corresponding to the literals in the clause c_i are disjoint from D, or a class d_j such that both p_j and $\overline{p_j}$ are not disjoint from D. It can be shown that F is satisfiable iff the certain answer to Q_F w.r.t. O_F is false. coNP-completeness follows from the fact that the size of $\mathcal{O}_F$ is polynomial w.r.t. to the size of F, and the size of Q_F is constant.

Finally, to show the combined complexity lower-bound, the proof is by reduction from the problem of checking the satisfiability of a Quantified Boolean Formula of the form $\forall x_1, \ldots, x_n \exists y_1, \ldots, y_m c_1 \wedge \cdots \wedge c_k$ to query answering in $\texttt{Hi(OWL 2 QL)}$, in such a way that both the TBox of the resulting ontology and the query have polynomial size with respect to the size of the formula. □

5 Optimization

In this section we present an optimized algorithm, called LAZY METAGROUNDING (LMG), for query answering over TBox-complete ontologies. LMG generalizes the idea of the algorithm proposed in [Lenzerini *et al.*, 2014] that was targeted to instance queries over $Hi(DL\text{-}Lite)$ ontologies. We then conclude the section by presenting a set of experiments showing that LMG outperforms the NAIVE algorithm, based on blind metagrounding.We point out that assuming to deal with TBox-complete ontologies is reasonable in practice. Indeed, once the domain analysis has highlighted the set of negative axioms that are certain, an automated procedure can be devised, that adds suitable facts to make the remaining negative axioms violated. We argue that, in most cases, such an addition achieves TBox-completeness without changing the intended models of the ontology.

Similarly to the NAIVE algorithm, LMG aims at exploiting existing OWL 2 QL reasoners as blackboxes computing answers to instance queries without metavariables. However, LMG improves NAIVE by significantly reducing the number of queries to be evaluated, by adopting an optimization strategy targeted to a scenario where the ABox is stored in a relational database ATB (this corresponds to the OBDM scenario, where the so-called "virtual ABox" is expressed by mappings [Lenzerini, 2011]), and additional relations in such database store the logical closure of the TBox (e.g., the table SubClassOf stores all pairs C_1, C_2 such that $\mathcal{O} \models \texttt{SubClassOf}(C_1 C_2)$).

While presenting LMG, we go beyond boolean queries, and deal with queries with distinguished variables. Coherently with SPARQL we sanction that in the answers to a query $Q(\vec{X})$, the distinguished variables $\vec{X}$ are bound to IRIs. Given an ontology $\mathcal{O}$ and a query $Q(X_d)$, LMG applies two functions: (*i*) BUILDQUERYPLAN, computing a sequence S of *annotated queries* encoding a query plan, (*ii*) and EXECUTEQUERYPLAN, executing S over $\mathcal{O}$.

The BUILDQUERYPLAN function. First, based on $Q(\vec{X_d})$, the

function constructs a graph G as follows. The nodes of G are the atoms of Q and there is an edge from a_1 to a_2 if and only if at least one of the following occurs: (*i*) a_1 is a TBox atom, a_2 is an ABox atom and a_1 involves a variable occurring in a_2; (*ii*) a_1 and a_2 are ABox atoms and a_1 involves a variable in individual position that occurs in predicate position in a_2; (*iii*) there is a variable x of Q that is not a metavariable, is not in X_d, and occurs in individual position both in a_1 and a_2 (in this case, there will be an edge also from a_2 to a_1). Intuitively, the function defines an edge from a_1 to a_2 to indicate that a_1 has to be evaluated before a_2, according to the following observations: (1) in order to exploit an OWL 2 QL reasoner one needs to instantiate the metavariables (see condition), (2) since the number of TBox axioms logically implied is typically less than the number of ABox axioms logically implied, to minimize the instantiations of the metavariables, it is preferable to evaluate the TBox atoms first (see condition (*i*)), and (3) in order to provide correct answers, atoms sharing existential variables in individual positions should be evaluated together (see condition (*iii*)).

Second, BUILDQUERYPLAN annotates each atom a of G with an integer $d(a)$ indicating its *depth*, by using the following strategy: *(a)* assign 0 to each TBox atom and set $k \leftarrow 1$; *(b)* do the following until the graph is empty: *(b.1)* remove from G atoms with depth $k-1$ together with their incoming edges; *(b.2)* assign k to each atom n occurring in a strongly connected component C; *(b.3)* $k \leftarrow k+1$.

Third, the function computes a sequence of queries $S = (q_0, \ldots, q_m)$, where m is the maximal depth of atoms in G, and q_i is such that its body is the conjunction of all atoms at depth i and its distinguished variables are all variables in its body that are in $\vec{X}_d$ or are metavariables in Q. Intuitively, BUILDQUERYPLAN aims at minimizing the number of conjunctive queries to be evaluated by an OWL 2 QL reasoner, by maximizing the number of atoms of the original query that can be evaluated within the same conjunctive query, based on the ordering induced by the edges of G.

The EXECUTEQUERYPLAN function. Based on S, the function progressively builds the answers to Q by storing into an intermediate relation A_i the result of executing the join operator of relational algebra between A_{i-1} and the answers to q_i. The goal of A_i is twofold: on one hand it maintains a projection of the answers to Q (if any) computed so far by evaluating the portion of Q consisting of the subqueries $q_0, \ldots q_i$; on the other hand, it provides bindings for the metavariables of q_{i+1} to obtain a metaground query evaluable using standard OWL 2 QL reasoners. Given the query plan $S = (q_0(\vec{w}_o), \ldots, q_m(\vec{w}_m))$, EXECUTEQUERYPLAN proceeds as follows.

- It evaluates q_0 over the ATB database, and stores its answers into A_0.
- For every $i = 1, \ldots, m$, let $\vec{v}_i$ be the tuple of variables appearing in $\vec{w}_i$ or $\vec{v}_{i-1}$ that are input variables of q_{i+1} ($i < m$), or are in $\vec{X}_d$ of Q ($\vec{v}_0$ is assumed to be simply $\vec{w}_0$). The function first builds the query q_i'' as the union of the various INST$(q_i, \vec{w_i'} \leftarrow \vec{t})$, where $\vec{w_i'}$ are the input variables of q_i, and $\vec{t}$ is a tuple in the projection of A_{i-1} onto such input variables. Then, to cope with metavariables in q_i'' that are not input variable of q_i, it computes the metagrounding of q_i'', evaluates it and stores its answers in G_i. Finally, the function computes A_i by executing the join between $A_{i-1}(\vec{v}_{i-1})$ and G_i.
- The function returns the set of tuples of IRIs in A_m disregarding possible complex expressions in A_m.

Theorem 9 *Given a TBox-complete ontology $\mathcal{O}$ and a query Q,* LMG *terminates and computes the certain answers to Q over $\mathcal{O}$.*

Clearly, LMG is in PTIME w.r.t. data and ontology complexity, and in NP w.r.t. combined complexity.

Experiments. To test the practical applicability of LMG, we have conducted a set of experiments on query answering using both LMG, and NAIVE. In particular, we compared the evaluation time of 8 queries, over an ontology containing about 100000 ABox axioms, and 260 TBox axioms.

The tests have been performed using Mastro [Calvanese *et al.*, 2011] as OWL 2 QL reasoner with java 7 on a Dell computer equipped with an Intel i7 2.70GHz processor, assigning 2GB memory to Java. The results of the experiments, reported in Table 1, show that, except for query $Q_1(x, y)$ (which is the query **select** \$$x$ \$$y$ {ClassAssertion(\$$x$ \$$y$)}, for which LMG and NAIVE resort to evaluate the same set of queries over Mastro), LMG outperforms NAIVE in terms of evaluation time, thanks to the smarter instantiation of the metavariables performed by LMG. We point out that such a speed up increases with the number of variables occurring both in individual and predicate positions.

	LMG		NAIVE	
query	**# of metaground queries**	**time**	**# of metaground queries**	**time**
Q_1	132	0.58 s	132	0.58 s
Q_2	4401	0.92 s	15376	15.12 min
Q_3	8619	1.33 s	> 70000	timeout 3 h
Q_4	12793	1.59 s	> 70000	timeout 3 h
Q_5	114	0.36 s	> 15000	timeout 3 h
Q_6	25	0.2 s	> 60000	timeout 3 h
Q_7	85	0.19 s	> 14000	timeout 3 h
Q_8	50	0.34 s	132	2.43 min

Table 1: Comparison between LMG and NAIVE

6 Conclusion

We plan to continue our work along several directions. We aim at studying interesting subclasses of queries, and characterize their complexity. For example, we already know that answering queries whose TBox atoms are ground is PTIME in ontology complexity, exactly like in [De Giacomo *et al.*, 2011]. Furthermore, we plan to increase the expressive power of Hi(OWL 2 QL) by extending both the ontology language (for example, with mechanisms for expressing integrity constraints), and the query language (for example, by allowing the use of atoms with data types as predicates), while keeping the same computational complexity of metaquerying. Also, we aim at studying further optimization techniques for query answering, targeted towards storage structures for the ABox different from the one considered here.

Acknowlegments

Work supported by the EU under the FP7 project "Optique" – grant n. FP7-318338, by MIUR under the SIR project "MODEUS" – grant n. RBSI14TQHQ, and by Regione Lazio under the project "MAGISTER".

References

[Arenas *et al.*, 2014] Marcelo Arenas, Georg Gottlob, and Andreas Pieris. Expressive languages for querying the semantic web. In *Proc. of the 33rd ACM SIGACT SIGMOD SIGAI Symp. on Principles of Database Systems (PODS)*, pages 14–26, 2014.

[Calvanese *et al.*, 2007] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.

[Calvanese *et al.*, 2011] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. The Mastro system for ontology-based data access. *Semantic Web J.*, 2(1):43–53, 2011.

[Chen *et al.*, 1993] Weidong Chen, Michael Kifer, and David Scott Warren. HILOG: A foundation for higher-order logic programming. *J. of Logic Programming*, 15(3):187–230, 1993.

[de Carvalho *et al.*, 2015] Victorio Albani de Carvalho, João Paulo A. Almeida, Claudenir M. Fonseca, and Giancarlo Guizzardi. Extending the foundations of ontology-based conceptual modeling with a multi-level theory. In *Proc. of the 34th Int. Conf. on Conceptual Modeling (ER)*, pages 119–133, 2015.

[De Giacomo *et al.*, 2011] Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Higher-order description logics for domain metamodeling. In *Proc. of the 25th AAAI Conf. on Artificial Intelligence (AAAI)*, 2011.

[Glimm *et al.*, 2010] Birte Glimm, Sebastian Rudolph, and Johanna Völker. Integrated metamodeling and diagnosis in OWL 2. In *Proc. of the 9th Int. Semantic Web Conf. (ISWC)*, volume 6496 of *Lecture Notes in Computer Science*, pages 257–272. Springer, 2010.

[Glimm, 2011] Birte Glimm. Using SPARQL with RDFS and OWL entailment. In *RW-11*, pages 137–201. 2011.

[Guizzardi *et al.*, 2015] Giancarlo Guizzardi, João Paulo Andrade Almeida, Nicola Guarino, and Victorio Albani de Carvalho. Towards an ontological analysis of powertypes. In *Proc. of the Joint Ontology Workshops 2015*, 2015.

[Homola *et al.*, 2013] Martin Homola, Jan Kluka, Vojtech Svátek, and Miroslav Vacura. Towards typed higher-order description logics. In *Proc. of the 26th Int. Workshop on Description Logic (DL)*, pages 221–233, 2013.

[Homola *et al.*, 2014] Martin Homola, Jan Kluka, Vojtech Svátek, and Miroslav Vacura. Typed higher-order variant of SROIQ - why not? In *Proc. of the 27th Int. Workshop on Description Logic (DL)*, pages 567–578, 2014.

[Jekjantuk *et al.*, 2010] Nophadol Jekjantuk, Gerd Gröner, and Jeff Z. Pan. Modelling and reasoning in metamodelling enabled ontologies. *Int. J. Software and Informatics*, 4(3):277–290, 2010.

[Kollia and Glimm, 2013] Ilianna Kollia and Birte Glimm. Optimizing SPARQL query answering over OWL ontologies. *J. Artif. Intell. Res. (JAIR)*, 48:253–303, 2013.

[Kontchakov *et al.*, 2014] Roman Kontchakov, Martin Rezk, Mariano Rodriguez-Muro, Guohui Xiao, and Michael Zakharyaschev. Answering SPARQL queries over databases under OWL 2 QL entailment regime. In *Proc. of the 13th Int. Semantic Web Conf. (ISWC)*, pages 552–567, 2014.

[Kubincová *et al.*, 2015] Petra Kubincová, Jan Kluka, and Martin Homola. Towards expressive metamodelling with instantiation. In *Proc. of the 28th Int. Workshop on Description Logic (DL)*, 2015.

[Lenzerini *et al.*, 2014] Maurizio Lenzerini, Lorenzo Lepore, and Antonella Poggi. Practical query answering over *Hi(DL-Lite$_R$)* knowledge bases. In *Proc. of the 27th Int. Workshop on Description Logic (DL)*, volume 1193, 2014.

[Lenzerini *et al.*, 2016] Maurizio Lenzerini, Lorenzo Lepore, and Antonella Poggi. A higher-order semantics for metaquerying in owl 2 ql. In *Proc. of the 15th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR)*, 2016.

[Lenzerini, 2011] Maurizio Lenzerini. Ontology-based data management. In *Proc. of the 20th Int. Conf. on Information and Knowledge Management (CIKM)*, pages 5–6, 2011.

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

[Pan and Horrocks, 2006] Jeff Z. Pan and Ian Horrocks. OWL FA: a metamodeling extension of OWL DL. In *Proc. of the 15th Int. World Wide Web Conf. (WWW)*, pages 1065–1066, 2006.