Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Lower and Upper Bounds for SPARQL Queries over OWL Ontologies

Birte Glimm and **Yevgeny Kazakov**
University of Ulm, Germany
<firstname.surname>@uni-ulm.de

Ilianna Kollia and **Giorgos Stamou**
National Technical University of Athens, Greece
ilianna2@mail.ntua.gr, gstam@cs.ntua.gr

Abstract

The paper presents an approach for optimizing the evaluation of SPARQL queries over OWL ontologies using SPARQL's OWL Direct Semantics entailment regime. The approach is based on the computation of lower and upper bounds, but we allow for much more expressive queries than related approaches. In order to optimize the evaluation of possible query answers in the upper but not in the lower bound, we present a query extension approach that uses schema knowledge from the queried ontology to extend the query with additional parts. We show that the resulting query is equivalent to the original one and we use the additional parts that are simple to evaluate for restricting the bounds of subqueries of the initial query. In an empirical evaluation we show that the proposed query extension approach can lead to a significant decrease in the query execution time of up to four orders of magnitude.

Introduction

Query answering—the computation of answers to users' queries w.r.t. ontologies and data—is an important task in the context of the Semantic Web. Ontologies formulated in the Web Ontology Language OWL (Motik, Patel-Schneider, and Parsia 2009) correspond to Description Logic (DL) knowledge bases (KBs) (Baader et al. 2007), i.e., to theories in a decidable fragment of First-Order Logic. Many Description Logic reasoners support some form of query answering, but while much effort has been spent on optimizing standard reasoning tasks for expressive DLs, e.g., the computation of subsumption relations between concepts (unary predicates), less attention has been given to optimizing the evaluation of more complex query patterns. The need for an efficient evaluation of complex queries over OWL ontologies becomes more urgent with the standardization of the SPARQL 1.1 Entailment Regimes (Glimm and Ogbuji 2013), which extend the SPARQL Query Language (Harris and Seaborne 2013) with the capability of querying also for answers that are entailed by the queried ontology, whereas without entailment regimes only explicitly stated knowledge is taken into account. In this paper, we propose a way of exploiting the knowledge in the ontology to more efficiently evaluate complex queries over OWL 2 DL ontologies under the OWL Direct Semantics entailment regime of SPARQL 1.1.

Since computing the answers to a query over an expressive knowledge base is computationally very costly, approximation techniques have been proposed that use a weakened version of the KB to compute a lower bound (yields sound but potentially incomplete results) and a strengthened version to compute an upper bound (yields complete but potentially unsound results) for the query answers (Zhou et al. 2014; Pan, Thomas, and Zhao 2009; Ren, Pan, and Zhao 2010). The weakened and strengthened versions are typically approximated in a simpler logic, for which more efficient reasoning algorithms can be used. Another well-known technique is to compute the bounds from a pre-model, i.e., a complete and clash-free tableau generated by a DL reasoner (Kollia and Glimm 2012). Facts that are derived deterministically are used as lower bound, while also non-deterministically derived facts are considered for the upper bound. For simple queries, the lower bound allows for simply reading off the answers, while answers in the "gap", i.e., potential answers in the upper but not the lower bound, usually have to be checked individually by performing a consistency check with a fully fledged OWL 2 DL reasoner. Following Zhou et al. (2014), we refer to the technique of first computing lower and upper bounds and then checking the answers in the gap as *hybrid approach*. Since performing many consistency checks is unlikely to work well in practice, Zhou et al. also propose optimizations such as extracting relevant fragments of the ontology for checking a potential answer and an adaptation of the summarization technique (Dolby et al. 2009; 2007).

In this paper, we also follow the hybrid approach, but we significantly extend the form of allowed queries. Furthermore, we present a way of using the knowledge in the ontology to extend the query such that the additionally added parts allow for better query planning and an overall faster query evaluation. Compared to query evaluation with standard static query planning (Kollia and Glimm 2013) our empirical analysis with a prototypical implementation shows an improvement of up to four orders of magnitude in the query execution times due to the proposed optimizations.

The idea of modifying the query for better query planning and evaluation has been presented before in the database

community. However, the proposed methods do not involve background knowledge, they rather present query approximations that are based on query forms with better complexity characteristics for query evaluation (like acyclicity, bounded treewidth, etc.) (Barceló, Libkin, and Romero 2012), or on queries expressed in less expressive query languages (for example without self-joins) (Fink and Olteanu 2011), or on queries for which the answers have been pre-computed (views) (Halevy 2001). On the other hand, there is a lot of work for query extension using semantic data descriptions, but mainly in the area of information retrieval focusing more on the representation of the query context for query expansion and refinement, and not on faster query evaluation (Bhogal, Macfarlane, and Smith 2007; Munir, Odeh, and McClatchey 2012; Luna, Revoredo, and Cozman 2010; Grootjen and van der Weide 2006).

Preliminaries

SPARQL's basic building blocks are so-called basic graph patterns (BGPs), which can then be combined using operators such as `UNION` or filters using the `FILTER` keyword. While the evaluation of BGPs is performed over the queried ontology, all other operators simply combine solutions obtained by evaluating the BGPs. Hence, we focus on the evaluation of BGPs in this paper. In the context of the OWL Direct Semantics Entailment Regime of SPARQL 1.1 (Glimm and Ogbuji 2013), BGPs correspond to "extended" OWL ontologies/KBs, where one can use variables in place of concept names (unary predicates), role names (binary predicates) or individual names (constants), which is why we next define the syntax of "extended" KBs in the DL $\mathcal{ALC}$ of which standard $\mathcal{ALC}$ KBs are a subset.

Definition 1 (Syntax). *The* syntax of an extended $\mathcal{ALC}$ knowledge base *or* query *is defined using a signature* (T_C, T_R, T_I)*, consisting of countably infinite disjoint sets* T_C *of* concept terms*,* T_R *of* role terms*, and* T_I *of* individual terms*. The set* T_C *(*T_R*,* T_I*) consists of the disjoint union of countably infinite sets of* concept (role, individual) names N_C *(*N_R*,* N_I*) and* concept (role, individual) variables V_C *(*V_R*,* V_I*). We further allow for the two special concepts* $\top$ *and* $\bot$*. Complex concept templates can be composed from these basic elements by using negation* $(\neg C)$*, conjunction* $(C_1 \sqcap C_2)$*, disjunction* $(C_1 \sqcup C_2)$*, or by quantification over a role* $(\forall r.C$ *and* $\exists r.C)$ *with* $C_{(i)} \in T_C$ *and* $r \in T_R$*. An* axiom template *(short: template) has the form* $C \sqsubseteq D$*,* $C(t)$*, or* $r(t, t')$ *with* C, D *concept templates,* r *a role term, and* t, t' *individual terms. If a concept template or an axiom template does not contain any variable, we call it* ground *or simply a* concept *or an* axiom*.*

An extended knowledge base $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ *is a finite set of axiom templates, where* $\mathcal{T}$ *is an* extended TBox *consisting of axiom templates of the form* $C \sqsubseteq D$ *and* $\mathcal{A}$ *is an* extended ABox *consisting of the remaining axiom templates. A (standard)* knowledge base *(short: KB) is an extended knowledge base that does not contain variables, that is, a set of axioms.*

A query q *is a set of templates and the* size $|q|$ *of* q *is the number of templates in* q*. If* q *contains only templates of the form* $A(t)$ *or* $r(t, t')$ *with* $A \in N_C$*,* $r \in N_R$*, and* $t, t' \in T_I$*, we call* q *a* conjunctive instance query*. We denote by* $vars(q)$ *the set of all (concept, role, and individual) variables occurring in* q*. If* $vars(q) = \emptyset$*, we call* q *a* ground *or* Boolean query*.*

$$\mathcal{K} = \mathcal{T} \cup \mathcal{A}, \text{ where}$$
$$\mathcal{T} = \{ B \sqsubseteq A \sqcup C,\ \exists r.B \sqsubseteq C\}$$
$$\mathcal{A} = \{A(a),\ B(b),\ r(a, b),\ A(d)\}$$
$$q = \{A(x), \exists r.Y(x), Y \sqsubseteq B\}$$
$$r \in N_R,\ \{A, B, C\} \subseteq N_C,\ \{a, b, c, d\} \subseteq N_I,\ Y \in V_C,\ x \in V_I$$

Figure 1: The (standard) knowledge base $\mathcal{K}$ and the query q used in the examples of this paper.

Note that the above definition can easily be extended to allow for more expressive Description Logics constructors such as nominals (concepts defined as an enumeration of individuals), role chains (e.g., for expressing transitivity), or (qualified) cardinality constraints (counting quantifiers in First-Order Logic) (Horrocks, Kutz, and Sattler 2006). We omit the introduction of these features for brevity, but the presented results are straightforwardly applicable.

Example 1. *Figure 1 gives an example of a standard knowledge base* $\mathcal{K}$ *and a query* q*. The knowledge base* $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ *consists of two TBox axioms and four ABox axioms. The query* q *consists of three axiom templates containing variables* $vars(q) = \{x, Y\}$*, where* x *is an individual variable, and* Y *is a concept variable.*

The query answers are then mappings such that the query instantiated by the mappings yields axioms that are entailed by the queried knowledge base. To make this more precise, we next define the semantics of (extended) KBs and queries.

Definition 2 (Semantics). *An* interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *consists of a non-empty set* $\Delta^{\mathcal{I}}$*, the* domain *of* $\mathcal{I}$*, and an* interpretation function $\cdot^{\mathcal{I}}$*, that assigns to each* $A \in N_C$ *a subset* $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$*, to each* $r \in N_R$ *a binary relation* $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$*, and to each* $a \in N_I$ *an element* $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$*. We further have* $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ *and* $\bot^{\mathcal{I}} = \emptyset$*. The interpretation is extended to (ground) complex concepts as follows:* $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$*,* $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$*,* $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$*,* $(\forall r.C)^{\mathcal{I}} = \{\delta \in \Delta^{\mathcal{I}} \mid \langle \delta, \delta' \rangle \in r^{\mathcal{I}} \rightarrow \delta' \in C^{\mathcal{I}}\}$*,* $(\exists r.C)^{\mathcal{I}} = \{\delta \in \Delta^{\mathcal{I}} \mid \exists \delta' \in C^{\mathcal{I}} : \langle \delta, \delta' \rangle \in r^{\mathcal{I}}\}$*.*

An interpretation $\mathcal{I}$ satisfies *a (ground) axiom* α *(notation:* $\mathcal{I} \models \alpha$*) as follows:* $\mathcal{I} \models C \sqsubseteq D$ *if* $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$*,* $\mathcal{I} \models C(a)$ *if* $a^{\mathcal{I}} \in C^{\mathcal{I}}$*, and* $\mathcal{I} \models r(a, b)$ *if* $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$*.* $\mathcal{I}$ *is a* model *of a (standard) knowledge base* $\mathcal{K}$ *(notation:* $\mathcal{I} \models \mathcal{K}$*) if* $\mathcal{I} \models \alpha$ *for every axiom* $\alpha \in \mathcal{K}$*. If such a model for* $\mathcal{K}$ *exists, we say that* $\mathcal{K}$ *is* consistent*.* $\mathcal{K}$ entails *an axiom* α *(notation:* $\mathcal{K} \models \alpha$*) if* $\mathcal{I} \models \alpha$ *for every model* $\mathcal{I}$ *of* $\mathcal{K}$*.*

A mapping *is a (partial) function* $\mu\colon V_C \cup V_R \cup V_I \rightarrow N_C \cup N_R \cup N_I$ *such that* $\mu(x) \in N_C$ *for each* $x \in V_C$*,* $\mu(x) \in N_R$ *for each* $x \in V_R$*, and* $\mu(x) \in N_I$ *for each* $x \in V_I$*. We use* $dom(\mu)$ *to denote the* domain *of a function* μ*. Two mappings* μ_1 *and* μ_2 *are* compatible *if* $\mu_1(x) = \mu_2(x)$ *for every* $x \in dom(\mu_1) \cap dom(\mu_2)$*. In this case,* the union of μ_1 and μ_2 *is the mapping* $\mu = \mu_1 \cup \mu_2$ *with* $dom(\mu) = dom(\mu_1) \cup dom(\mu_2)$

defined by $\mu(x) = \mu_1(x)$ *for* $x \in dom(\mu_1)$ *and* $\mu(x) = \mu_2(x)$ *for* $x \in dom(\mu_2)$. *If* M_1 *and* M_2 *are two sets of mappings then the* join of M_1 and M_2 *is the set of mappings* $M_1 \bowtie M_2 = \{\mu_1 \cup \mu_2 \mid \mu_1 \in M_1,\ \mu_2 \in M_2, \mu_1 \text{ and } \mu_2 \text{ are compatible}\}$.

Given a query q, *we denote with* $\mu(q)$ *the result of replacing each variable* $x \in dom(\mu)$ *with* $\mu(x)$. *If* $vars(q) \subseteq dom(\mu)$, *we write* $\mathcal{I}, \mu \models q$ *if* $\mathcal{I}$ *satisfies every (ground) axiom template of* $\mu(q)$. *A mapping* μ *is a* certain answer *for a query* q *over a (standard) KB* $\mathcal{K}$, *written* $\mathcal{K}, \mu \models q$, *if* $\mathcal{I}, \mu \models q$ *for each* $\mathcal{I}$ *such that* $\mathcal{I} \models \mathcal{K}$. *We set* $\mathsf{ans}(\mathcal{K}, q) = \{\mu \mid \mathcal{K}, \mu \models q,\ dom(\mu) = vars(q)\}$. *Queries* q_1 *and* q_2 *are* equivalent w.r.t. $\mathcal{K}$ *if* $\mathsf{ans}(\mathcal{K}, q_1) = \mathsf{ans}(\mathcal{K}, q_2)$.

Note that if $q = q_1 \cup q_2$ then $\mathsf{ans}(\mathcal{K}, q) = \mathsf{ans}(\mathcal{K}, q_1) \bowtie \mathsf{ans}(\mathcal{K}, q_2)$.

Example 2. *It is easy to check that the KB* $\mathcal{K}$ *in Figure 1 has a model* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *with* $\Delta^{\mathcal{I}} = \{d_1, d_2, d_3, d_4\}$, $a^{\mathcal{I}} = d_1$, $b^{\mathcal{I}} = d_2$, $c^{\mathcal{I}} = d_3$, $d^{\mathcal{I}} = d_4$, $A^{\mathcal{I}} = \{d_1, d_2, d_4\}$, $B^{\mathcal{I}} = \{d_2\}$, $C^{\mathcal{I}} = \{d_1\}$, *and* $r^{\mathcal{I}} = \{\langle d_1, d_2\rangle\}$. *Note that* $\mathcal{I} \models A(a)$ *but* $\mathcal{I} \not\models B(a)$. *Hence,* $\mathcal{K} \not\models A \sqsubseteq B$. *Thus, for the mapping* μ *with* $\mu(Y) = A$, *we have* $\mathcal{K}, \mu \not\models \{Y \sqsubseteq B\}$. *The same also holds for every* μ *with* $\mu(Y) \in \{C, \top\}$. *Thus,* $\mathsf{ans}(\mathcal{K}, \{Y \sqsubseteq B\}) = \{\bot, B\}$. *Similarly, one can show that* $\mathsf{ans}(\mathcal{K}, \{A(x)\}) = \{\mu \mid \mu(x) \in \{a, d\}\}$ *since* $\mathcal{K} \models A(a)$, $\mathcal{K} \models A(d)$, *but* $\mathcal{K} \not\models A(b)$ *and* $\mathcal{K} \not\models A(c)$, *and that* $\mathsf{ans}(\mathcal{K}, \{\exists r.Y(x)\}) = \{\mu \mid \mu(x) = a,\ \mu(Y) \in \{B, \top\}\}$ *since* $\mathcal{K} \models \exists r.B(a)$ *and* $\mathcal{K} \models \exists r.\top(a)$, *but* $\mathcal{K} \not\models \exists r.A(a)$, $\mathcal{K} \not\models \exists r.B(b)$, *etc. Hence, for* $q = \{A(x),\ \exists r.Y(x),\ Y \sqsubseteq B\}$ *in Figure 1 we obtain* $\mathsf{ans}(\mathcal{K}, q) = \mathsf{ans}(\mathcal{K}, \{A(x)\}) \bowtie \mathsf{ans}(\mathcal{K}, \{\exists r.Y(x)\}) \bowtie \mathsf{ans}(\mathcal{K}, \{Y \sqsubseteq B\}) = \{\mu \mid \mu(x) = a,\ \mu(Y) = B\}$.

Bounds for Queries and Subqueries

Since entailment checking is the standardized reasoning task for OWL reasoners, a naive way to evaluate queries is to instantiate the query with all possible query mappings and check entailment of the resulting axioms using an OWL reasoner. Since the number of possible mappings can be very large, this method is not efficient even with small queries. Therefore, practical algorithms try to reduce the number of entailment tests by determining using other techniques, which mappings can or cannot satisfy the query. We present this general idea using the notion of a query bound.

Definition 3 (Query Bound). *Let* $\mathcal{K}$ *be a knowledge base and* q *a query. A* bound for q w.r.t. $\mathcal{K}$ *is a pair of sets* $[\mathcal{L}; \mathcal{U}]$ *of mappings such that* $\mathcal{L} \subseteq \mathsf{ans}(\mathcal{K}, q) \subseteq \mathcal{U}$. *We call* $\mathcal{L}$ *the* lower *and* $\mathcal{U}$ *the* upper bound *for* q *over* $\mathcal{K}$. *The bound* $[\mathcal{L}; \mathcal{U}]$ *is* exact *if* $\mathcal{L} = \mathcal{U}$ *(and hence* $= \mathsf{ans}(\mathcal{K}, q)$*).*

We usually assume that $\mathcal{K}$ is fixed and refer to bounds for queries without mentioning $\mathcal{K}$.

Example 3. *Consider again the KB* $\mathcal{K}$ *in Figure 1. Since* $\{A(a), A(d)\} \subseteq \mathcal{K}$ *we have* $\mathcal{K} \models A(a)$ *and* $\mathcal{K} \models A(d)$. *Hence* $\mathsf{ans}(\mathcal{K}, \{A(x)\})$ *contains at least all* μ *such that* $\mu(x) \in \{a, d\}$. *That is, we can find the lower bound* $\mathcal{L} = \{\mu \mid \mu(x) \in \{a, d\}\}$ *for the query* $\{A(x)\}$ *over* $\mathcal{K}$ *without performing any tests.*

To find an upper bound for $\{A(x)\}$, *consider the model* $\mathcal{I}$ *constructed in Example 2. Note that* $\mathcal{I} \not\models A(c)$. *Thus, from this model alone one can conclude that* $\mathcal{K} \not\models A(c)$ *and hence that* $\mathsf{ans}(\mathcal{K}, \{A(x)\}) \subseteq \{\mu \mid \mu(x) \in \{a, b, d\}\}$. *Thus, the set* $\mathcal{U} = \{\mu \mid \mu(x) \in \{a, b, d\}\}$ *provides an upper bound for the query* $\{A(x)\}$ *over* $\mathcal{K}$.

Although the model $\mathcal{I}$ *can be similarly used for finding an upper bound for complex templates, such as* $\{\exists r.Y(x)\}$, *in general it can only be found by iterating over all possible mappings for* x *and* Y *and checking which instances of this template are entailed by the model. Therefore, in practice, one does not compute the bounds for complex templates.*

The bounds for the query $\{Y \sqsubseteq B\}$ *can be computed by classifying the knowledge base and retrieving subsumption relationships with concept* B. *Since for classification one usually needs to consider just the (relatively small) TBox* $\mathcal{T}$, *the bounds for this query can be computed exactly. That is, in our example we have* $\mathcal{L} = \mathcal{U} = \{\bot, B\}$.

Query bounds can be also extracted by weakening and strengthening of the knowledge bases in tractable fragments (Zhou et al. 2014). Lower query bounds for simple instance queries can also be obtained by inspecting which assertions are derived deterministically when testing satisfiability of $\mathcal{K}$ using (hyper-)tableau procedures (Kollia and Glimm 2012). Using the computed bounds for simple queries, one can derive the bounds for more complex queries as follows.

Lemma 1. *Let* $q = q_1 \cup q_2$, $[\mathcal{L}_1; \mathcal{U}_1]$ *be a query bound for* q_1 *and* $[\mathcal{L}_2; \mathcal{U}_2]$ *a query bound for* q_2. *Then* $[\mathcal{L}_1 \bowtie \mathcal{L}_2; \mathcal{U}_1 \bowtie \mathcal{U}_2]$ *is a query bound for* $q_1 \cup q_2$.

Proof. It is easy to show that if $M_1 \subseteq M_1'$ and $M_2 \subseteq M_2'$ are sets of mappings then $M_1 \bowtie M_2 \subseteq M_1' \bowtie M_2'$. Since $\mathsf{ans}(\mathcal{K}, q_1 \cup q_2) = \mathsf{ans}(\mathcal{K}, q_1) \bowtie \mathsf{ans}(\mathcal{K}, q_2)$, from $\mathcal{L}_1 \subseteq \mathsf{ans}(\mathcal{K}, q_1) \subseteq \mathcal{U}_1$ and $\mathcal{L}_2 \subseteq \mathsf{ans}(\mathcal{K}, q_2) \subseteq \mathcal{U}_2$ we obtain $\mathcal{L}_1 \bowtie \mathcal{L}_2 \subseteq \mathsf{ans}(\mathcal{K}, q_1 \cup q_2) \subseteq \mathcal{U}_1 \bowtie \mathcal{U}_2$. □

The above lemma gives rise to a query answering algorithm that first retrieves the query bounds for each axiom template in the query. Second, the join of all query bounds is computed. Third, the mappings that are in the upper but not in the lower bound are checked using an OWL reasoner. Since computing a join of (potentially large) sets of mappings can be expensive, one can instead try to reduce the bounds for axiom templates using other parts of the query.

Definition 4 (Subquery Bound). *If* $q = q_1 \cup q_2$, *then we say that* q_1 *is a* subquery *of* q. *The pair of sets* $[\mathcal{L}_1; \mathcal{U}_1]$ *is a* subquery bound for q_1 in q over $\mathcal{K}$ *if* $\mathcal{L}_1 \subseteq (\mathsf{ans}(\mathcal{K}, q_1) \cap \mathcal{U}_1)$ *and* $\mathsf{ans}(\mathcal{K}, q) = (\mathsf{ans}(\mathcal{K}, q_1) \cap \mathcal{U}_1) \bowtie \mathsf{ans}(\mathcal{K}, q_2)$.

Intuitively, a subquery bound provides a range for those answers of the subquery q_1 that are sufficient to evaluate the query q. If the query $q = q_1 \cup q_2$ is clear from the context, we usually call $[\mathcal{L}_1; \mathcal{U}_1]$ a subquery bound of q_1 without mentioning q (and $\mathcal{K}$ as for query bounds). Clearly, if $q = q_1 \cup q_2$ and $[\mathcal{L}_1; \mathcal{U}_1]$ is a query bound for q_1, then it is also subquery bound for q_1 in q. The converse property does not hold as shown in the next example.

Example 4. *Consider* $\mathcal{K}$ *from Figure 1 and* $q = q_1 \cup q_2$ *with* $q_1 = \{A(x)\}$ *and* $q_2 = \{\exists r.Y(x)\}$. *As shown in Example 2,* $\mathsf{ans}(\mathcal{K}, q_1) = \{\mu \mid \mu(x) \in \{a, d\}\}$ *and* $\mathsf{ans}(\mathcal{K}, q_2) = \{\mu \mid \mu(x) = a,\ \mu(Y) \in \{B, \top\}\}$. *It is easy to see that* $\mathsf{ans}(\mathcal{K}, q) = \mathsf{ans}(\mathcal{K}, q_1) \bowtie \mathsf{ans}(\mathcal{K}, q_2) = \mathsf{ans}(\mathcal{K}, q_2)$.

Now, consider $\mathcal{L}_1 = \mathcal{U}_1 = \{\mu \mid \mu(x) = a\}$. *Clearly,* $\mathcal{L}_1 \subseteq \mathsf{ans}(\mathcal{K}, q_1) \cap \mathcal{U}_1$. *Also,* $(\mathsf{ans}(\mathcal{K}, q_1) \cap \mathcal{U}_1) \bowtie \mathsf{ans}(\mathcal{K}, q_2) = \mathcal{U}_1 \bowtie \mathsf{ans}(\mathcal{K}, q_2) = \mathsf{ans}(\mathcal{K}, q_2) = \mathsf{ans}(\mathcal{K}, q)$. *Hence* $[\mathcal{L}_1; \mathcal{U}_1]$ *is a subquery bound for* q_1 *in* q. *It is, however, not a query bound for* q_1 *since* $\mathsf{ans}(\mathcal{K}, q_1) \not\subseteq \mathcal{U}_1$.

Lemma 2. *Let* $q = q_1 \cup q_2 \cup q_3$ *and let* $[\mathcal{L}_1; \mathcal{U}_1]$, $[\mathcal{L}_2; \mathcal{U}_2]$ *be subquery bounds for* q_1 *and* q_2 *w.r.t.* q, *respectively. Then* $\mathsf{ans}(\mathcal{K}, q) = (\mathsf{ans}(\mathcal{K}, q_1) \cap \mathcal{U}_1) \bowtie (\mathsf{ans}(\mathcal{K}, q_2) \cap \mathcal{U}_2) \bowtie \mathsf{ans}(\mathcal{K}, q_3)$.

Proof. Clearly, $(\mathsf{ans}(\mathcal{K}, q_1) \cap \mathcal{U}_1) \bowtie (\mathsf{ans}(\mathcal{K}, q_2) \cap \mathcal{U}_2) \bowtie \mathsf{ans}(\mathcal{K}, q_3) \subseteq \mathsf{ans}(\mathcal{K}, q_1) \bowtie \mathsf{ans}(\mathcal{K}, q_2) \bowtie \mathsf{ans}(\mathcal{K}, q_3) = \mathsf{ans}(\mathcal{K}, q)$, so it remains to show the converse $\mathsf{ans}(\mathcal{K}, q) \subseteq (\mathsf{ans}(\mathcal{K}, q_1) \cap \mathcal{U}_1) \bowtie (\mathsf{ans}(\mathcal{K}, q_2) \cap \mathcal{U}_2) \bowtie \mathsf{ans}(\mathcal{K}, q_3)$.

Since $[\mathcal{L}_1; \mathcal{U}_1]$ is a subquery bound for q_1, $\mathsf{ans}(\mathcal{K}, q) = (\mathsf{ans}(\mathcal{K}, q_1) \cap \mathcal{U}_1) \bowtie \mathsf{ans}(\mathcal{K}, q_2 \cup q_3) = (\mathsf{ans}(\mathcal{K}, q_1) \cap \mathcal{U}_1) \bowtie \mathsf{ans}(\mathcal{K}, q_2) \bowtie \mathsf{ans}(\mathcal{K}, q_3)$. Hence, for every $\mu \in \mathsf{ans}(\mathcal{K}, q)$ there exist $\mu_1 \in \mathsf{ans}(\mathcal{K}, q_1) \cap \mathcal{U}_1$, $\mu_2 \in \mathsf{ans}(\mathcal{K}, q_2)$, and $\mu_3 \in \mathsf{ans}(\mathcal{K}, q_3)$ that are compatible such that $\mu = \mu_1 \cup \mu_2 \cup \mu_3$. Likewise, since $[\mathcal{L}_2; \mathcal{U}_2]$ is a subquery bound for q_2, there exist $\mu'_1 \in \mathsf{ans}(\mathcal{K}, q_1)$, $\mu'_2 \in \mathsf{ans}(\mathcal{K}, q_2) \cap \mathcal{U}_2$, and $\mu'_3 \in \mathsf{ans}(\mathcal{K}, q_3)$ that are compatible such that $\mu = \mu'_1 \cup \mu'_2 \cup \mu'_3$. Since $\mathrm{dom}(\mu_i) = \mathrm{vars}(q_i) = \mathrm{dom}(\mu'_i)$ $(1 \leq i \leq 3)$, from $\mu_1 \cup \mu_2 \cup \mu_3 = \mu'_1 \cup \mu'_2 \cup \mu'_3$ we obtain $\mu_1 = \mu'_1$, $\mu_2 = \mu'_2$, and $\mu_3 = \mu'_3$. In particular $\mu = \mu_1 \cup \mu'_2 \cup \mu_3 \in (\mathsf{ans}(\mathcal{K}, q_1) \cap \mathcal{U}_1) \bowtie (\mathsf{ans}(\mathcal{K}, q_2) \cap \mathcal{U}_2) \bowtie \mathsf{ans}(\mathcal{K}, q_3)$, which proves $\mathsf{ans}(\mathcal{K}, q) \subseteq (\mathsf{ans}(\mathcal{K}, q_1) \cap \mathcal{U}_1) \bowtie (\mathsf{ans}(\mathcal{K}, q_2) \cap \mathcal{U}_2) \bowtie \mathsf{ans}(\mathcal{K}, q_3)$. □

Theorem 1 illustrates how the bound of one subquery (q_1) can be reduced using the bound of another subquery (q_2). Note that the subqueries q_1 and q_2 can be arbitrary, e.g., empty, singleton templates or sets of possibly overlapping templates.

Theorem 1. *Let* $q = q_1 \cup q_2 \cup q_3$, *let* $[\mathcal{L}_1; \mathcal{U}_1]$, $[\mathcal{L}_2; \mathcal{U}_2]$ *be subquery bounds for* q_1 *and* q_2 *w.r.t.* q, *respectively, and let* $M_1 \subseteq \mathcal{U}_1$ *be such that* $M_1 \bowtie \mathcal{U}_2 = \emptyset$. *Then* $[\mathcal{L}_1 \setminus M_1; \mathcal{U}_1 \setminus M_1]$ *is also a subquery bound for* q_1.

Proof. Clearly, if $\mathcal{L}_1 \subseteq (\mathsf{ans}(\mathcal{K}, q_1) \cap \mathcal{U}_1)$ then $\mathcal{L}_1 \setminus M_1 \subseteq (\mathsf{ans}(\mathcal{K}, q_1) \cap (\mathcal{U}_1 \setminus M_1))$. It remains, therefore, to show that $\mathsf{ans}(\mathcal{K}, q) = (\mathsf{ans}(\mathcal{K}, q_1) \cap (\mathcal{U}_1 \setminus M_1)) \bowtie \mathsf{ans}(\mathcal{K}, q_2) \bowtie \mathsf{ans}(\mathcal{K}, q_3)$. By Lemma 2, $\mathsf{ans}(\mathcal{K}, q) = (\mathsf{ans}(\mathcal{K}, q_1) \cap \mathcal{U}_1) \bowtie (\mathsf{ans}(\mathcal{K}, q_2) \cap \mathcal{U}_2) \bowtie \mathsf{ans}(\mathcal{K}, q_3) = (\mathsf{ans}(\mathcal{K}, q_1) \cap (\mathcal{U}_1 \setminus M_1)) \bowtie (\mathsf{ans}(\mathcal{K}, q_2) \cap \mathcal{U}_2) \bowtie \mathsf{ans}(\mathcal{K}, q_3) \cup (\mathsf{ans}(\mathcal{K}, q_1) \cap M_1) \bowtie (\mathsf{ans}(\mathcal{K}, q_2) \cap \mathcal{U}_2) \bowtie \mathsf{ans}(\mathcal{K}, q_3) = (\mathsf{ans}(\mathcal{K}, q_1) \cap (\mathcal{U}_1 \setminus M_1)) \bowtie (\mathsf{ans}(\mathcal{K}, q_2) \cap \mathcal{U}_2) \bowtie \mathsf{ans}(\mathcal{K}, q_3)$ since $(\mathsf{ans}(\mathcal{K}, q_1) \cap M_1) \bowtie (\mathsf{ans}(\mathcal{K}, q_2) \cap \mathcal{U}_2) \bowtie \mathsf{ans}(\mathcal{K}, q_3) \subseteq M_1 \bowtie \mathcal{U}_2 \bowtie \mathsf{ans}(\mathcal{K}, q_3) = \emptyset$. □

Example 5. *Consider the knowledge base* $\mathcal{K}$ *and the query* $q = \{\exists r.Y(x), A(x), Y \sqsubseteq B\}$ *from Figure 1. Assume that we have found the query bound* $[\mathcal{L}_2; \mathcal{U}_2]$ *for* $q_2 = \{A(x)\}$ *as in Example 3, namely,* $\mathcal{L}_2 = \{\mu \mid \mu(x) \in \{a, d\}\}$ *and* $\mathcal{U}_2 = \{\mu \mid \mu(x) \in \{a, b, d\}\}$. *As mentioned before,* $[\mathcal{L}_2; \mathcal{U}_2]$ *is also a subquery bound for* q_2 *in* q. *Let* $[\mathcal{L}_1; \mathcal{U}_1]$ *be the trivial subquery bound for* $q_1 = \{\exists r.Y(x)\}$ *in* q, *i.e.,* $\mathcal{L}_1 = \emptyset$ *and* $\mathcal{U}_1 = \{\mu \mid \mu(x) \in \{a, b, c, d\}, \mu(Y) \in \{\bot, A, B, C, \top\}\}$. *Furthermore, let* $M_1 = \{\mu \mid \mu(x) = c\}$. *Clearly,* $M_1 \bowtie \mathcal{U}_2 = \emptyset$. *Hence, by Theorem 1,* $[\mathcal{L}_1 \setminus M_1; \mathcal{U}_1 \setminus M_1] = [\emptyset; \{\mu \mid \mu(x) \in \{a, b, d\}, \mu(Y) \in \{\bot, A, B, C, \top\}\}]$ *is also a subquery bound for* q_1 *in* q. *Similarly, using the (exact) query bound* $[\mathcal{L}_3; \mathcal{U}_3]$ *for* $q_3 = \{Y \sqsubseteq B\}$ *with* $\mathcal{L}_3 = \mathcal{U}_3 = \{\mu \mid \mu(Y) \in \{\bot, B\}\}$, *one can further reduce the upper bound for* q_1 *to* $\{\mu \mid \mu(x) \in \{a, b, d\}, \mu(Y) \in \{\bot, B\}\}$.

Improving Bounds via Query Extension

As seen from the previous section, subquery bounds can be improved using bounds of other subqueries of this query. Thus, if a query q can be extended to an *equivalent* query $q \cup q'$, the number of reasoner calls performed for evaluating the axiom templates in q can be reduced using q'. In this section we consider a method by which one can extend queries based on a ground version of q together with axioms from the queried knowledge base. The proposed query extension method is similar to the method for deciding containment between conjunctive queries (Calvanese, De Giacomo, and Lenzerini 2008), with the main difference (apart from allowing concept and role variables) that instead of checking query containment, we construct a query contained in the given query ourselves.

Theorem 2 (Query Extension). *Let* $\mathcal{K}$ *be a knowledge base,* q *a query, and* μ *a mapping with* $dom(\mu) = vars(q)$ *such that for each* $x \in dom(\mu)$, $\mu(x)$ *does not occur in* $\mathcal{K}$ *and for each* $x, y \in dom(\mu)$, $x \neq y$ *implies* $\mu(x) \neq \mu(y)$. *Then for every query* q' *such that* $vars(q') \subseteq dom(\mu)$ *and* $\mathcal{K} \cup \mu(q) \models \mu(q')$, *we have* $\mathsf{ans}(\mathcal{K}, q) = \mathsf{ans}(\mathcal{K}, q \cup q')$.

Proof. Clearly, $\mathsf{ans}(\mathcal{K}, q \cup q') \subseteq \mathsf{ans}(\mathcal{K}, q)$, so it remains to show that $\mathsf{ans}(\mathcal{K}, q) \subseteq \mathsf{ans}(\mathcal{K}, q \cup q')$. Take any $\mu' \in \mathsf{ans}(\mathcal{K}, q)$, i.e., $\mathcal{K} \models \mu'(q)$. To prove that $\mu' \in \mathsf{ans}(\mathcal{K}, q \cup q')$, we need to show that $\mathcal{K} \models \mu'(q \cup q')$. Since $\mathcal{K} \models \mu'(q)$, it is left to prove that $\mathcal{K} \models \mu'(q')$. Let ρ be such that $\mu'(x) = \rho(\mu(x))$ for every $x \in \mathrm{vars}(q) = \mathrm{dom}(\mu)$. That is, ρ replaces each value of $\mu(x)$ with the corresponding value of $\mu'(x)$. Since $\mu(x) \neq \mu(y)$ for each $x \neq y$, $x, y \in \mathrm{dom}(\mu)$, such function ρ always exists. Then $\mathcal{K} \cup \mu(q) \models \mu(q')$ implies $\rho(\mathcal{K} \cup \mu(q)) \models \rho(\mu(q')) = \mu'(q')$. Since $\rho(\mathcal{K} \cup \mu(q)) = \rho(\mathcal{K}) \cup \rho(\mu(q))$, $\rho(\mathcal{K}) = \mathcal{K}$ (because $\mu(x)$ does not occur in $\mathcal{K}$ for every $x \in \mathrm{dom}(\mu)$) and $\mathcal{K} \models \mu'(q) = \rho(\mu(q))$, we obtain $\mathcal{K} \models \mu'(q')$ what was required to show. □

Example 6. *For the KB* $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ *and query* $q = \{A(x), \exists r.Y(x), Y \sqsubseteq B\}$ *in Figure 1, consider a mapping* μ *such that* $\mu(x) = a_x$, $\mu(Y) = A_Y$ *with* $a_x \in N_I, A_Y \in N_C$. *Clearly the conditions of Theorem 2 are satisfied. Note that* $\mathcal{T} \cup \mu(q) = \{B \sqsubseteq A \sqcup C, \exists r.B \sqsubseteq C, A(a_x), \exists r.A_Y(a_x), A_Y \sqsubseteq B\} \models C(a_x)$. *Thus, for* $q' = \{C(x)\}$, *we have* $\mathcal{K} \cup \mu(q) \models C(a_x) = \mu(q')$, *and by Theorem 2, the query* q *has the same answers for* $\mathcal{K}$ *as the extended query* $q \cup q' = \{A(x), \exists r.Y(x), Y \sqsubseteq B, C(x)\}$.

Using Theorem 2, the improved algorithm for evaluating a query q over a knowledge base $\mathcal{K}$ can now be described as follows:

1. Replace every (concept, role, individual) variable in q with a fresh distinct (concept, role, individual) name. Let μ be the mapping that performs this replacement.
2. Add the resulting axioms $\mu(q)$ to $\mathcal{K}$ and perform materialization, i.e., compute all concept assertions $A(a)$, role

assertions $r(a, b)$, and subsumptions $A \sqsubseteq B$ with atomic concepts and roles entailed by $\mathcal{K} \cup \mu(q)$. Let $\mathcal{K}'$ be the resulting set of such entailed axioms.
3. Apply the reversed mapping q^{-1} to $\mathcal{K}'$, i.e., replace each $\mu(x)$ (freshly introduced in Step 1) in $\mathcal{K}'$ with x. Let q' be the query obtained by this replacement, i.e., $\mu(q') = \mathcal{K}'$.
4. Compute the query bounds for the templates in q' and use them to improve the subquery bounds for the templates in q using Theorem 1 for $q \cup q'$.
5. Evaluate q using the improved subquery bounds.

Note that in the above procedure, instead of computing the materialization of $\mathcal{K} \cup \mu(q)$ one can alternatively compute the materialization for $\mathcal{K}_1 \cup \mu(q)$ for some $\mathcal{K}_1 \subseteq \mathcal{K}$. Indeed, for the query q' obtained in this way, we have $\mathcal{K}_1 \cup \mu(q) \models \mu(q')$, which implies $\mathcal{K} \cup \mu(q) \models \mu(q')$, and thus q is equivalent to $q \cup q'$ w.r.t. $\mathcal{K}$ by Theorem 2. In practice, to improve performance of query extension one could take $\mathcal{K}_1$ to be the TBox $\mathcal{T}$ of $\mathcal{K}$ (like in Example 6), or even a subset of $\mathcal{T}$ consisting of axioms that fall into some tractable fragment.

Similarly, not all of the computed axioms of the materialization $\mathcal{K}'$ should necessarily be converted to axiom templates of q'. For example, if $\mathcal{K}_1 \models A \sqsubseteq B$ and $\mathcal{K}'$ contains $A(\mu(x))$, then $\mathcal{K}'$ will also contain $B(\mu(x))$. However, the template $B(x)$ is less useful than $A(x)$ in the query extension since $\mathsf{ans}(\mathcal{K}, \{A(x)\}) \subseteq \mathsf{ans}(\mathcal{K}, \{B(x)\})$, and thus the query bound for $B(x)$ is likely to be worse than the query bounds for $A(x)$. That is, $B(x)$ is unlikely to further reduce the subquery bounds for the templates of q after $A(x)$. Thus, in practice, we use only direct instance and subsumption relations of $\mathcal{K}'$ for the query extension q'.

Example 7. *Consider the extended query $q \cup q' = \{A(x), \exists r.Y(x), Y \sqsubseteq B, C(x)\}$ computed in Example 6. Using the model $\mathcal{I}$ for $\mathcal{K}$ from Example 2, since $\mathcal{I} \models C(a)$, but $\mathcal{I} \not\models C(b)$, $\mathcal{I} \not\models C(c)$, and $\mathcal{I} \not\models C(d)$, we can derive the upper bound $\mathcal{U} = \{\mu \mid \mu(x) = a\}$ for the query $\{C(x)\}$. Using this upper bound, it is now possible to reduce the upper bound for the subquery $\{A(x)\}$ to $\mathcal{U}$. Since $\mathcal{U}$ was a subset of the lower bound for $\{A(x)\}$ (see Example 2), this subquery can be evaluated without performing any further entailment tests. The new upper bound $\mathcal{U}$ can be also used to further reduce the upper bound for the subquery $\{\exists r.Y(x)\}$ to $\{\mu \mid \mu(x) = a, \mu(Y) \in \{\bot, B\}\}$. After this reduction, this subquery can be evaluated using just two entailment tests.*

Evaluation

The proposed method has been implemented and evaluated over a set of well-known benchmarking ontologies and relevant datasets, for several forms of queries. Although it can be used, in general, for improving the performance of most hybrid query answering systems, here the evaluation is based on the system described in Kollia et al. (Kollia and Glimm 2013), which, to the best of our knowledge, is the only system that supports the evaluation of complex queries over OWL 2 DL ontologies under the OWL Direct Semantics entailment regime of SPARQL 1.1.

Kollia et al. (Kollia and Glimm 2013) propose a method (referred to as evalStatic) for computing bounds for query templates using the HermiT reasoner (Glimm et al. 2014). In particular, the classified concept hierarchy and the premodel that HermiT constructs are used to compute lower and upper query template bounds as described in Example 3. Using these bounds a sophisticated ordering of the axiom templates in the query is performed. Then, query evaluation starts with the first template, retrieves the mappings from the lower bound and checks all remaining mappings from the upper bound using either dedicated reasoner methods or entailment checks. While the former is quite cheap, involving only memory look-ups, the latter is usually significantly more expensive, involving reasoning procedures. For the next axiom template, the evaluation is analogous with the difference that the join variables of the current template with the previous templates are taken into account.

In our implemented method (referred to as evalExt) we improve the subquery bounds computed in evalStatic using the Algorithm of the previous section. Afterwards, we perform the ordering and evaluation methods of Kollia et al. using the improved subquery bounds. As we will show this improvement of subquery bounds leads to a significant reduction in the query execution times compared to evalStatic.

We evaluated evalStatic and evalExt over the Lehigh University Benchmark (LUBM) (Guo, Pan, and Heflin 2005), the University Ontology Benchmark (UOBM) (Ma et al. 2006), and the Semintec ontology from the Oxford ontology library.[1] We used a range of custom queries since the queries provided for LUBM and UOBM are only simple conjunctive instance queries. Additionally, since the above ontologies do not involve many complex, non-deterministic axioms, we added some complex terminological axioms, in order to show the improvement in cases where most expensive reasoning procedures are involved. All experiments were performed on a Mac OS X Lion machine with a 2.53 GHz Intel Core i7 processor and Java 1.6 allowing 1GB of Java heap space. The ontologies and all code required to perform the experiments are available online.[2]

The evaluation results are shown in Table 1. Column 1 shows the query and extension templates, columns 2 and 3 show the query answering times and the number of performed entailment checks for evalStatic, respectively, and columns 4 and 5 show the respective numbers for evalExt. In all queries the time spent for query extension is negligible compared to the time spent for query evaluation.

For LUBM, we used the first three departments of LUBM(1,0), which consist of 3,883 individuals. For all queries, additional extension templates were derived, which have significantly better query bounds than the complex templates of the queries. This directly translates to a significantly lower number of entailment checks for evalExt and, hence, a reduction in execution times.

The Semintec ontology is more challenging than LUBM since more individuals are involved, in total 17,941. The use of evalStatic leads to a timeout for all queries, which is not very surprising given the large number of individuals and the consequently higher number of required entailment checks. The higher number of required entailment checks

[1] http://www.cs.ox.ac.uk/isg/ontologies/lib/

[2] http://www.image.ece.ntua.gr/~ilianna/AAAI2015.zip

Table 1: Query answering times in seconds (n/a indicates a timeout, > 30 min) and number of performed entailment checks for LUBM(1,0) with the three first departments, Semintec, and UOBM with the first department using evalStatic and evalExt. The extension templates for each query q_i used in evalExt is given in q'_i ($1 \leq i \leq 15$).

	evalStatic		evalExt	
LUBM	time	#entail	time	#entail
$q_1 = \{\exists worksFor.Organization(x)\}$, $q'_1 = \{Employee(x)\}$	452.55	1,398	22.41	217
$q_2 = \{X \sqsubseteq \exists headOf.College, \exists isTaughtBy.X(y)\}$, $q'_2 = \{X \sqsubseteq Dean, CourseTaughtByDean(y)\}$	634.43	1,714	0.26	0
$q_3 = \{X \sqsubseteq Course \sqcap \forall isTakenBy.GraduateStudent, \exists teachingAssistantOf.X(y)\}$, $q'_3 = \{X \sqsubseteq GraduateCourse, TeachingAssistantOfGraduateCourse(y)\}$	614.55	1,635	23.76	158
Semintec				
$q_4 = \{\exists hasOwner.Man(x)\}$, $q'_4 = \{AccountWithAtLeastOneManOwner(x)\}$	n/a		588.99	2,292
$q_5 = \{\exists hasCreditCard.Gold(x)\}$, $q'_5 = \{OwnerOfAtLeastOneGoldCreditCard(x)\}$	n/a		23.16	88
$q_6 = \{X \sqsubseteq Client \sqcap \forall hasSexValue.FemaleSex, \exists isCreditCardOf.X(y)\}$, $q'_6 = \{X \sqsubseteq Woman, CreditCardHeldByWoman(y)\}$	n/a		1,043.28	846
$q_7 = \{X \sqsubseteq Finished \sqcap \forall hasLoanStatusValue.OKStatus, \exists hasLoan.X(y)\}$, $q'_7 = \{X \sqsubseteq NoProblemsFinishedLoan, AccountWithOkFinishedLoan(y)\}$	n/a		410.59	406
UOBM				
$q_8 = \{isAdvisedBy(x,y), GraduateStudent(x), Woman(y)\}$, $q'_8 = \{Professor(y)\}$	20.84	47	10.54	19
$q_9 = \{isTaughtBy(x,y), GraduateCourse(x), Woman(y)\}$, $q'_9 = \{Faculty(y)\}$	21.63	51	12.06	26
$q_{10} = \{teachingAssistantOf(x,y), GraduateCourse(y), Woman(x)\}$, $q'_{10} = \{TeachingAssistant(x)\}$	12.78	32	5.60	12
$q_{11} = \{\exists takesCourse.\top(x), \forall takesCourse.GraduateCourse(x)\}$, $q'_{11} = \{GraduateStudent(x)\}$	569.06	1,694	256.77	1,332
$q_{12} = \{\exists worksFor.Organization(x), Woman(x)\}$, $q'_{12} = \{Employee(x)\}$	n/a		18.36	135
$q_{13} = \{X \sqsubseteq \exists isHeadOf.Department, \exists isTaughtBy.X(y)\}$, $q'_{13} = \{X \sqsubseteq Chair, CourseTaughtByChair(y)\}$	n/a		1.99	4
$q_{14} = \{X \sqsubseteq \exists isHeadOf.College, \exists isTaughtBy.X(y)\}$, $q'_{14} = \{X \sqsubseteq Dean, CourseTaughtByDean(y)\}$	n/a		0.17	0
$q_{15} = \{X \sqsubseteq \exists isHeadOf.College, \exists isAdvisedBy.X(y)\}$, $q'_{15} = \{X \sqsubseteq Dean, PersonAdvisedByDean(y)\}$	n/a		0.21	0

also affected evalExt, but all queries could be evaluated in the given time limit since the extension templates lead to a significant reduction in the number of possible answers. Note that $X \sqsubseteq Client$ is not an extension template for q_6, although this might be expected from the axiom template $X \sqsubseteq Client \sqcap \forall hasSexValue.FemaleSex$. The axiom template is, however, omitted since we also obtain the extension template $X \sqsubseteq Woman$, Semintec contains $Woman \sqsubseteq Client$, and we only use the direct subsumption relations in evalExt.

In our last evaluation scenario, we used the UOBM benchmark, which contains an ontology that is an extension of the LUBM ontology in terms of expressivity, involving non-determinism. In order to reduce the reasoning time, we removed the nominals that are very hard to deal with and we used the first department of UOBM containing 3,043 individuals. The queries are particularly constructed to use the non-determinism, i.e., our aim is to inspect cases where the lower and the upper bounds do not coincide, which is the case for *GraduateStudent*, *Woman* and *GraduateCourse*. As before, evalExt finds extension templates for all queries, which considerably reduce the subquery bounds for other templates resulting in a significantly reduced number of entailment checks in comparison to evalStatic and, hence, reduced query answering times. For UOBM, we observe the most significant reduction in query answering times, which is up to four orders of magnitude.

Conclusions

In the current paper, we presented an approach for exploiting lower and upper bounds computed for SPARQL queries which are evaluated under the OWL Direct Semantics entailment regime. In particular, we showed how we can build equivalent queries with additional extension templates, whose bounds can be used to restrict the bounds of templates of the initial query. Through our experimental evaluation we showed that the use of these extension templates can lead to a significant reduction in query answering time, which can be up to four orders of magnitude.

Acknowledgments

The presented work was partially supported by the Greek Strategic Reference Framework 2007-2013 Operational Programme 'Competitiveness and Entrepreneurship' under the contract 'DARIAH-ATTIKH: Developing a Greek Research Infrastructure for the Humanities'. Ilianna Kollia and Giorgos Stamou are funded by the Hellenic General Secretariat for Research and Technology (Call 103/2012).

References

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P., eds. 2007. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, second edition.

Barceló, P.; Libkin, L.; and Romero, M. 2012. Efficient approximations of conjunctive queries. In *Proceedings of the 31st Symposium on Principles of Database Systems*, PODS'12, 249–260. ACM.

Bhogal, J.; Macfarlane, A.; and Smith, P. 2007. A review of ontology based query expansion. *Information Processing and Management* 43(4):866 – 886.

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

Dolby, J.; Fokoue, A.; Kalyanpur, A.; Kershenbaum, A.; Schonberg, E.; Srinivas, K.; and Ma, L. 2007. Scalable semantic retrieval through summarization and refinement. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI'07)*, 299–304.

Dolby, J.; Fokoue, A.; Kalyanpur, A.; Schonberg, E.; and Srinivas, K. 2009. Scalable highly expressive reasoner (SHER). *Journal of Web Semantics* 7(4):357–361.

Fink, R., and Olteanu, D. 2011. On the optimal approximation of queries using tractable propositional languages. In *Proceedings of the 14th International Conference on Database Theory*, ICDT'11, 174–185. ACM.

Glimm, B., and Ogbuji, C., eds. 2013. *SPARQL 1.1 Entailment Regimes*. W3C Recommendation. Available at http://www.w3.org/TR/sparql11-entailment/.

Glimm, B.; Horrocks, I.; Motik, B.; Stoilos, G.; and Wang, Z. 2014. Hermit: An OWL 2 reasoner. *Journal of Automated Reasoning* 53:245–269.

Grootjen, F., and van der Weide, T. 2006. Conceptual query expansion. *Data and Knowledge Engineering* 56(2):174 – 193.

Guo, Y.; Pan, Z.; and Heflin, J. 2005. LUBM: A benchmark for OWL knowledge base systems. *Journal of Web Semantics* 3(2-3):158–182.

Halevy, A. Y. 2001. Answering queries using views: A survey. *The VLDB Journal* 10(4):270–294.

Harris, S., and Seaborne, A., eds. 2013. *SPARQL 1.1 Query Language*. W3C Recommendation. Available at http://www.w3.org/TR/sparql11-query/.

Horrocks, I.; Kutz, O.; and Sattler, U. 2006. The even more irresistible $\mathcal{SROIQ}$. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR'06)*, 57–67.

Kollia, I., and Glimm, B. 2012. Cost based query ordering over OWL ontologies. In *Proceedings of the 11th International Semantic Web Conference (ISWC 2012)*, volume 7649 of *Lecture Notes in Computer Science*, 231–246. Springer.

Kollia, I., and Glimm, B. 2013. Optimizing SPARQL Query Answering over OWL Ontologies. *Journal of Artificial Intelligence Research* 48:253–303.

Luna, J. E. O.; Revoredo, K.; and Cozman, F. G. 2010. Semantic query extension through probabilistic description logics. In *Proceedings of the 6th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2010)*, volume 654 of *CEUR Workshop Proceedings*, 49–60. CEUR-WS.org.

Ma, L.; Yang, Y.; Qiu, Z.; Xie, G.; Pan, Y.; and Liu, S. 2006. Towards a complete OWL ontology benchmark. In *The Semantic Web: Research and Applications*, Lecture Notes in Computer Science. Springer. 125–139.

Motik, B.; Patel-Schneider, P. F.; and Parsia, B., eds. 2009. *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax*. W3C Recommendation. Available at http://www.w3.org/TR/owl2-syntax/.

Munir, K.; Odeh, M.; and McClatchey, R. 2012. Ontology-driven relational query formulation using the semantic and assertional capabilities of OWL-DL. *Knowledge-Based Systems* 35:144–159.

Pan, J. Z.; Thomas, E.; and Zhao, Y. 2009. Completeness guaranteed approximation for OWL-DL query answering. In *Proceedings of the 22nd International Workshop on Description Logics (DL'09)*, volume 477 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Ren, Y.; Pan, J. Z.; and Zhao, Y. 2010. Soundness preserving approximation for TBox reasoning. In *Proceedings of the 25th National Conference on Artificial Intelligence (AAAI'10)*. AAAI Press.

Zhou, Y.; Nenov, Y.; Grau, B. C.; and Horrocks, I. 2014. Pay-as-you-go OWL query answering using a triple store. In *Proceedings of the 28th Conference on Artificial Intelligence (AAAI'14)*, 1142–1148.