Lower and Upper Bounds for SPARQL Queries over OWL Ontologies

Birte Glimm[1] Yevgeny Kazakov[1] Ilianna Kollia[2] and Giorgos Stamou[2]

[1] Ulm University, Germany, <firstname.surname>@uni-ulm.de
[2] National Technical University of Athens, Greece, ilianna2@mail.ntua.gr, gstam@cs.ntua.gr

Introduction

The recent standardization of the SPARQL 1.1 Entailment Regimes [2], which extend the SPARQL Query Language [3] with the capability of querying also for implicit knowledge makes the need for an efficient evaluation of complex queries over OWL ontologies urgent. We present an approach for optimizing the evaluation of SPARQL queries over OWL ontologies using SPARQL's OWL Direct Semantics entailment regime. Such queries consist of *axiom templates*, i.e., Description Logic (DL) axioms with variables in place of concept, role and individual names. Answers to such queries are mappings of query (concept, role or individual) variables to corresponding (concept, role or individual) names that instantiate the axiom templates to axioms entailed by the queried knowledge base (KB).

Since computing query answers over an expressive KB 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 results [9, 7, 8]. Another well-known technique is to compute the bounds from a complete and clash-free tableau generated by a DL reasoner [4, 5]. Deterministically derived facts are used as lower bound, while also non-deterministically derived ones are considered for the upper bound. 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.

While we also use bounds, we allow for much more expressive queries than related approaches. To optimize the evaluation of possible query answers in the gap, we present a query extension approach that uses the TBox of the queried KB 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. More details about our method as well as more evaluation results can be found in the extended version of our paper [1].

Improving Bounds via Query Extension

We will show the intuition of the proposed query extension method through an example. Let $\mathcal{K}$ be a KB, A, B, C be concept names, r a role name, a, b, c, d individual names, x

an individual variable, Y a concept variable and

$$\mathcal{T} = \{\, B \sqsubseteq A \sqcup C,\ \exists r.B \sqsubseteq C\} \quad \mathcal{A} = \{A(a),\ B(b),\ r(a,b),\ A(d)\}$$
$$q = \{A(x), \exists r.Y(x), Y \sqsubseteq B\}$$

Note that the only answer for q over $\mathcal{K}$ is the mapping $\{\mu \mid \mu(x) = a, \mu(Y) = B\}$.

First we compute bounds for axiom templates of q over $\mathcal{K}$. Since $\{A(a),\ A(d)\} \subseteq \mathcal{K}$ we have $\mathcal{K} \models A(a)$ and $\mathcal{K} \models A(d)$. 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 $\{A(x)\}$, we can use a model $\mathcal{I}$ of $\mathcal{K}$. It is easy to check that $\mathcal{K}$ 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 $\mathcal{I} \not\models A(c)$. Thus, from this model alone one can conclude that $\mathcal{K} \not\models A(c)$ and hence that 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 KB and retrieving subsumption relationships for 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, i.e., in our example we have $\mathcal{L} = \mathcal{U} = \{\mu \mid \mu(Y) \in \{\bot, B\}\}$.

Our query extension method uses additionally the notion of a *subquery bound*,which provides a range for those answers of a subquery (subset of templates) of q that are sufficient to evaluate q. The intuition of our method is that 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 q can be reduced using q'. The proposed algorithm can be summarized as follows: First, we replace every (concept, role, individual) variable in q with a fresh distinct (concept, role, individual) name. For our example, consider a mapping μ such that $\mu(x) = a_x$, $\mu(Y) = A_Y$ with a_x an individual and A_Y a concept name. Then we materialize and classify $\mathcal{K}$ plus $\mu(q)$, where $\mu(q)$ denotes the result of replacing each variable x in q with $\mu(x)$, i.e., we compute all concept assertions $A(a)$, role assertions $r(a, b)$, and subsumptions $A \sqsubseteq B$ with atomic concepts and roles entailed by $\mathcal{K} \cup \mu(q)$. Afterwards, we replace names back with their corresponding variables in the extended KB. In our example $\mathcal{T} \cup \mu(q) = \{B \sqsubseteq A \sqcup C, \exists r.B \sqsubseteq C, A(a_x), \exists r.A_Y(a_x), A_Y \sqsubseteq B\} \models C(a_x)$. Thus, for $q' = \{C(x)\}$, we have $\mathcal{K} \cup \mu(q) \models C(a_x) = \mu(q')$, and it holds that the query q has the same answers for $\mathcal{K}$ as the extended query $q \cup q' = \{A(x),\ \exists r.Y(x),\ Y \sqsubseteq B,\ C(x)\}$. In the end, we compute query bounds for the templates in q' and use them to improve the subquery bounds for templates in q. Using again the model $\mathcal{I}$ for $\mathcal{K}$, 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 $\{A(x)\}$ of q to $\mathcal{U}$. Since $\mathcal{U}$ is a subset of the lower bound for $\{A(x)\}$ (computed in the beginning of the section), this subquery can be evaluated without performing any further entailment tests. The new upper bound $\mathcal{U}$ can also be used to further reduce the upper bound for the subquery $\{\exists r.Y(x)\}$ of q to

Table 1. Query answering times in seconds (n/a indicates a timeout, > 30 min) and number of performed entailment checks for UOBM with the first department using evalStatic and evalExt

UOBM	evalStatic time	evalStatic #entail	evalExt time	evalExt #entail
$q_1 = \{isAdvisedBy(x, y), GraduateStudent(x), Woman(y)\}$, $q'_1 = \{Professor(y)\}$	20.84	47	10.54	19
$q_2 = \{isTaughtBy(x, y), GraduateCourse(x), Woman(y)\}$, $q'_2 = \{Faculty(y)\}$	21.63	51	12.06	26
$q_3 = \{teachingAssistantOf(x, y), GraduateCourse(y), Woman(x)\}$, $q'_3 = \{TeachingAssistant(x)\}$	12.78	32	5.60	12
$q_4 = \{\exists worksFor.Organization(x), Woman(x)\}$, $q'_4 = \{Employee(x)\}$	n/a		18.36	135
$q_5 = \{X \sqsubseteq \exists isHeadOf.Department, \exists isTaughtBy.X(y)\}$, $q'_5 = \{X \sqsubseteq Chair, CourseTaughtByChair(y)\}$	n/a		1.99	4
$q_6 = \{X \sqsubseteq \exists isHeadOf.College, \exists isAdvisedBy.X(y)$, $q'_6 = \{X \sqsubseteq Dean, PersonAdvisedByDean(y)\}$	n/a		0.21	0

$\{\mu \mid \mu(x) = a, \mu(Y) \in \{\bot, B\}\}$. 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 benchmark ontologies and relevant datasets and for several forms of queries. Although it can be used, in general, for improving the performance of most query answering systems based on query bounds, here the evaluation is based on the system described in Kollia et al. [5], 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. In our implemented method (referred to as evalExt) we improve the subquery bounds computed in evalStatic [5] using the method described in the previous section. Afterwards, we perform the ordering and evaluation methods of Kollia et al. using the improved subquery bounds.

In Table 1 we show the results of the evaluation on the University Ontology Benchmark (UOBM) [6] using a range of custom queries since the queries provided for UOBM are only simple conjunctive instance queries. Column 1 shows the query q_i and extension templates q'_i ($1 \leq i \leq 6$), 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. We observe that 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 reduction in query answering times is up to four orders of magnitude.

References

1. Glimm, B., Kazakov, Y., Kollia, I., Stamou, G.: Lower and upper bounds for SPARQL queries over OWL ontologies. In: Proceedings of the 29th Conference on Artificial Intelligence (AAAI'15) (2015)
2. Glimm, B., Ogbuji, C. (eds.): SPARQL 1.1 Entailment Regimes. W3C Recommendation (2013), available at http://www.w3.org/TR/sparql11-entailment/
3. Harris, S., Seaborne, A. (eds.): SPARQL 1.1 Query Language. W3C Recommendation (2013), available at http://www.w3.org/TR/sparql11-query/
4. Kollia, I., Glimm, B.: Cost based query ordering over OWL ontologies. In: Proceedings of the 11th International Semantic Web Conference (ISWC 2012). Lecture Notes in Computer Science, vol. 7649, pp. 231–246. Springer (2012)
5. Kollia, I., Glimm, B.: Optimizing SPARQL Query Answering over OWL Ontologies. Journal of Artificial Intelligence Research 48, 253–303 (2013)
6. Ma, L., Yang, Y., Qiu, Z., Xie, G., Pan, Y., Liu, S.: Towards a complete OWL ontology benchmark. In: The Semantic Web: Research and Applications, pp. 125–139. Lecture Notes in Computer Science, Springer (2006)
7. Pan, J.Z., Thomas, E., Zhao, Y.: Completeness guaranteed approximation for OWL-DL query answering. In: Proceedings of the 22nd International Workshop on Description Logics (DL'09). CEUR Workshop Proceedings, vol. 477. CEUR-WS.org (2009), http://dblp.uni-trier.de/db/conf/dlog/dlog2009.html#PanTZ09
8. Ren, Y., Pan, J.Z., Zhao, Y.: Soundness preserving approximation for TBox reasoning. In: Proceedings of the 25th National Conference on Artificial Intelligence (AAAI'10). AAAI Press (2010)
9. Zhou, Y., Nenov, Y., Grau, B.C., Horrocks, I.: Pay-as-you-go OWL query answering using a triple store. In: Proceedings of the 28th Conference on Artificial Intelligence (AAAI'14). pp. 1142–1148 (2014)