Approximating OWL-DL Ontologies

Jeff Z. Pan and **Edward Thomas**
Department of Computing Science, University of Aberdeen
UK

Abstract

Efficient query answering over ontologies is one of the most useful and important services to support Semantic Web applications. Approximation has been identified as a potential way to reduce the complexity of query answering over OWL DL ontologies. Existing approaches are mainly based on syntactic approximation of ontological axioms and queries. In this paper, we propose to recast the idea of knowledge compilation into approximating OWL DL ontologies with DL-Lite ontologies, against which query answering has only polynomial data complexity. We identify a useful category of queries for which our approach guarantees also completeness. Furthermore, this paper reports on the implementation of our approach in the ONTOSEARCH2 system and preliminary, but encouraging, benchmark results which compare ONTOSEARCH2's response times on a number of queries with those of existing ontology reasoning systems.

Introduction

Ontologies play a key role in the Semantic Web (Berners-Lee, Hendler, & Lassila 2001), for which W3C has standardised a Web Ontology Language OWL (Patel-Schneider, Hayes, & Horrocks 2004); in this paper, we are interested in the OWL DL sub-language ('DL' for Description Logic) due to the existence of reasoning support. A growing library of ontologies is available online, covering a wide range of human knowledge. For this large body of knowledge to be usable by Semantic Web applications, a framework that allows efficient query answering over ontologies is required.

How to provide efficient querying answering service for expressive DLs has been an important open problem in KR. It has been shown that the complexity of ontology entailment in $\mathcal{SHOIN}(\mathbf{D}^+)$, i.e., OWL DL, is NEXPTIME. This indicates query answering over OWL DL ontologies is at least NEXPTIME. Approximation has been identified as a potential way to reduce the complexity of reasoning over OWL DL ontologies. Existing approaches (Stuckenschmidt & van Harmelen 2002; Wache, Groot, & Stuckenschmidt 2005; Hitzler & Vrandecic 2005; Groot, Stuckenschmidt, & Wache 2005; Hurtado, Poulovassilis, & Wood 2006) are mainly based on syntactic approximation of ontological axioms and queries. All these approaches could introduce unsound answers. To the best of our knowledge, we have not seen any published framework on sound (and possibly incomplete) approximations on ontology query answering, not to mention efficient ones.

In this paper, we propose to recast the idea of knowledge compilation (Selman & Kautz 1996) into semantic approximation of OWL DL ontologies. The idea of knowledge compilation is simple: users can write statements in an expressive representation language; such statements can be complied into a restricted language that allows efficient inference. In this way, users do not have to use less expressive language which might be too limited for practical applications. In (Selman & Kautz 1996), Selman and Kautz showed how propositional logical theories can be compiled into Horn theories that approximate the original information; they also applied this idea on subsumption reasoning for the Description Logic $\mathcal{FL}$. In this paper, we investigate applying knowledge compilation on query answering over OWL DL ontologies. Namely, we propose approximating OWL DL ontologies (or simply source ontologies) with corresponding DL-Lite (Calvanese *et al.* 2004; 2005) ontologies (or simply target ontologies), against which query answering has only polynomial data complexity, and provide algorithms to compute target DL-Lite ontologies.

One major advantage of our approach is that DL-Lite query engines (which make use of SQL engines after some query rewriting) can be exploited to provide soundness preserving query answering services for OWL DL ontologies, since all entailments, of a source OWL DL ontology, that are expressible in DL-Lite are kept in the target ontology. Although the size of target ontologies is only polynomial w.r.t. the signatures of source ontologies, it still takes a considerable amount of time to calculate target ontologies if target ontologies are computed naively, in particular when the source ontology is huge. To solve the problem, we propose several optimisation algorithms to reduce some unnecessary entailment checkings and to reduce storage of some redundant DL-Lite axioms. Furthermore, we identify an important category of queries for which our approach guarantees both soundness and completeness. We have implemented our approach in the ONTOSEARCH2 system. Our experiments show that the optimisations significantly reduce the time for computing target ontologies. Our evaluation with the Lehigh University Benchmark shows that ONTOSEARCH2 outperforms a number of contemporary ontology reasoning systems.

DL-based Ontologies and Query Answering

Due to the limitation of space, we do not provide a formal introduction of Description Logics (DLs), but rather point the reader to (Baader *et al.* 2002). It should be noted that, even for the smallest propositionally closed DL $\mathcal{ALC}$ (which only provides the class constructors $\neg C, C \sqcap D, C \sqcup D, \exists R.C$ and $\forall R.C$), the complexity of logical entailment is EXPTIME. Recently, Calvanese et al proposed DL-Lite (Calvanese *et al.* 2004) which can express most features in UML class diagrams but still has a low reasoning overhead (worst case polynomial time, compared to worst case exponential time in the case of most widely used Description Logics). DL-Lite supports the following axioms:

1. class inclusion axioms: $B \sqsubseteq C$ where B is a basic class $B := \mathsf{A} \mid \exists R \mid \exists R^{-}$ and C is a general class $C := B \mid \neg B \mid C1 \sqcap C2$ (where A denotes an named class and R denotes a named property);
2. functional property axioms: $Func(R), Func(R^{-})$, where R is a named property;
3. individual axioms: $B(\mathtt{a}), R(\mathtt{a}, \mathtt{b})$ where a and b are named individuals.

A conjunctive query (CQ) q is of the form $q(X) \leftarrow \exists Y.conj(X, Y, Z)$ or simply $q(X) \leftarrow conj(X, Y, Z)$, where $q(X)$ is called the head, $conj(X, Y, Z)$ is called the body, X are called the distinguished variables, Y are existentially quantified variables called the non-distinguished variables, Z are individual names, and $conj(X, Y, Z)$ is a conjunction of atoms of the form $\mathsf{A}(v)$, $R(v_1, v_2)$, where A, R are respectively *named* classes and *named* properties, v, v_1 and v_2 are *individual* variables in X and Y or individual names in Z. As usual, an interpretation $\mathcal{I}$ satisfies an ontology $\mathcal{O}$ if it satisfies all the axioms in $\mathcal{O}$; in this case, we say $\mathcal{I}$ is a model of $\mathcal{O}$. Given an evaluation $[X \mapsto S]$, if every model $\mathcal{I}$ of $\mathcal{O}$ satisfies $q_{[X \mapsto S]}$, we say $\mathcal{O}$ entails $q_{[X \mapsto S]}$; in this case, S is called a *solution* of q. A disjunctive query (DQ) is a set of conjunctive queries sharing the same head. Theoretically, allowing only named classes and properties as atoms is not a restriction, as we can always define such named classes and properties in ontologies. Practically, this should not be an issue as querying against *named* relations is a usual practice when people query over relational databases.

After some careful query rewriting by DL-Lite reasoners (Calvanese *et al.* 2005), query answering over DL-Lite ontologies can be carried out by an SQL engine, so as to take advantage of existing query optimisation strategies and algorithms provided by modern database management systems.

Knowledge Compilation

Selman and Kautz illustrated the idea of knowledge compilation in (Selman & Kautz 1996) by showing how a propositional theory can be compiled into a tractable form consisting of a set of Horn clauses. As a logically equivalent set of Horn clauses does not always exist, they proposed to use Horn lower-bound and Horn upper-bound to approximate the original theory.

Let Σ be a set of clauses (the original theory), the sets Σ_{lb} and Σ_{ub} of Horn clauses are respectively a Horn lower-bound and a Horn upper-bound of Σ iff $\mathcal{M}(\Sigma_{\mathrm{lb}}) \subseteq \mathcal{M}(\Sigma) \subseteq \mathcal{M}(\Sigma_{\mathrm{ub}})$, or, equivalently, $\Sigma_{\mathrm{lb}} \models \Sigma \models \Sigma_{\mathrm{ub}}$. This says the Horn lower-bound is logically stronger than the original theory and the Horn upper-bound is logically weaker than it. A Horn lower-bound Σ_{glb} is a greatest Horn lower-bound iff there is no set Σ' of Horn clauses such that $\mathcal{M}(\Sigma_{\mathrm{lb}}) \subset \mathcal{M}(\Sigma') \subseteq \mathcal{M}(\Sigma)$. A Horn upper-bound Σ_{lub} is a least Horn upper-bound iff there is no set Σ' of Horn clauses such that $\mathcal{M}(\Sigma) \subseteq \mathcal{M}(\Sigma') \subset \mathcal{M}(\Sigma_{\mathrm{lub}})$.

Approximations of OWL DL Ontologies

Semantic Approximations

In this section, we apply the idea of knowledge compilation on semantically approximating a source ontology $\mathcal{O}_1$ in a more expressive DL $\mathcal{L}_1$ (source language) with its least upper-bound $\mathcal{O}_2$ in a less expressive DL $\mathcal{L}_2$ (target language). In this setting, we have $\mathcal{O}_1 \models \mathcal{O}_2$, which distinguishes semantic approximations from syntactic ones, as the latter cannot guarantee soundness (see the example below).

Example 1 *Suppose we want to approximate the OWL DL ontology $\mathcal{O}_1 = \{\geqslant 2R \sqsubseteq \mathsf{C}, R \sqsubseteq S, R(\mathtt{a}, \mathtt{b})\}$ with an $\mathcal{ALC}$ ontology. Using syntactic approximations, one can approximate $\mathcal{O}_1$ with $\mathcal{O}_2 = \{\exists R \sqsubseteq \mathsf{C}, R(a, b)\}$, in which the number restriction $\geqslant 2R$ is syntactically approximated as $\exists R$ and $R \sqsubseteq S$ is ignored as it is not expressible in $\mathcal{ALC}$. One drawback of this approximation is that $\mathcal{O}_2$ entails* $\mathsf{C}(a)$, *which is not entailed by the source ontology $\mathcal{O}_1$.*

In the rest of the paper, we use $\mathbf{N}_C$, $\mathbf{N}_P$ and $\mathbf{N}_I$ to denote the set of named classes, named properties and named individuals, respectively, used in $\mathcal{O}_1$. We always assume $\mathbf{N}_C$, $\mathbf{N}_P$ and $\mathbf{N}_I$ are finite, as $\mathcal{O}_1$ is a finite set of axioms.

The following definition provides the notion of least upper-bound in our setting; i.e., we consider all $\mathcal{L}_2$ axioms that are entailed by $\mathcal{O}_1$.

Definition 1 ***(Entailment Set)*** *The* entailment set *of $\mathcal{O}_1$ w.r.t. $\mathcal{L}_2$, denoted as $\mathbf{ES}(\mathcal{O}_1, \mathcal{L}_2)$, is the set which contains* all *$\mathcal{L}_2$ axioms (constructed by using only vocabulary in $\mathbf{N}_C$, $\mathbf{N}_P$ and $\mathbf{N}_I$) that are entailed by $\mathcal{O}_1$.*

There are two remarks here: (1) $\mathbf{ES}(\mathcal{O}_1, \mathcal{L}_2)$ could be infinite. For example, given $\mathcal{O}_1 = \{\top \sqsubseteq \forall R.(\exists R.\top), \mathrm{Trans}(R), R(a, b)\}$, there exist infinite axioms (such as $\exists R.\top(a)$, $\exists R.(\exists R.\top)(a)$, $\exists R.(\exists R.(\exists R.\top))(a)$, ...) in $\mathbf{ES}(\mathcal{O}_1, \mathcal{ALC})$. Whether $\mathbf{ES}(\mathcal{O}_1, \mathcal{L}_2)$ is finite depends on the target language $\mathcal{L}_2$. (2) If $\mathbf{ES}(\mathcal{O}_1, \mathcal{L}_2)$ is finite, it is a candidate for $\mathcal{O}_2$.

We can construct entailment sets from the corresponding axiom sets (see the following definition).

Definition 2 *(Axiom Set) The* axiom set *of $\mathcal{L}_2$ w.r.t. $\mathbf{N}_C$, $\mathbf{N}_P$ and $\mathbf{N}_I$, denoted as $\mathbf{AS}(\mathcal{L}_2, \mathbf{N}_C, \mathbf{N}_P, \mathbf{N}_I)$, is the set which contains all $\mathcal{L}_2$ axioms that are constructed by using only vocabulary in $\mathbf{N}_C$, $\mathbf{N}_P$ and $\mathbf{N}_I$.*

It should be noted that axiom sets are not always finite. For example, $\mathbf{AS}(\mathcal{ALC}, \{\mathsf{A}\}, \{R\}, \{\mathtt{b}\})$ is not finite, as it contains infinite individual axioms of the form $\exists R.\mathsf{A}(\mathtt{b})$, $\exists R.\exists R.\mathsf{A}(\mathtt{b})$, $\exists R.\exists R.\exists R.\mathsf{A}(\mathtt{b})$, ... Although it is still possible to restrict our attention to axioms with limited length in $\mathcal{ALC}$, in this paper, we are mainly interested in Description Logics ($\mathcal{L}_2$) which come with a finite axiom set $\mathbf{AS}(\mathcal{L}_2, \mathbf{N}_C, \mathbf{N}_P, \mathbf{N}_I)$. In particular, we choose DL-Lite, due to its nice trade-off between expressive power and efficiency. The following lemma shows that $\mathbf{AS}(\text{DL-Lite}, \mathbf{N}_C, \mathbf{N}_P, \mathbf{N}_I)$ is finite.

Lemma 3 *Given the finite sets of $\mathbf{N}_C$, $\mathbf{N}_P$ and $\mathbf{N}_I$, the axiom set of DL-Lite w.r.t. $\mathbf{N}_C$, $\mathbf{N}_P$ and $\mathbf{N}_I$, i.e. $\mathbf{AS}(\textit{DL-Lite}, \mathbf{N}_C, \mathbf{N}_P, \mathbf{N}_I)$, is finite.*

Proof: Given the DL-Lite syntax for constructing class descriptions, there are only one way ($C := C_1 \sqcap C_2$) to construct class descriptions with infinite length, but we can always find some finite class descriptions which are equivalent to them, by e.g. disallowing repeated basic classes in conjunctive class descriptions. In fact, after normalising DL-Lite class axioms into the following two forms: $B_1 \sqsubseteq B_2$ and $B_1 \sqsubseteq \neg B_2$ (Calvanese *et al.* 2005), there exists no way to construct class descriptions with infinite length. ∎

We conclude this sub-section by presenting an algorithm to compute entailment sets w.r.t. DL-Lite. Given an $\mathcal{L}_1$ ontology $\mathcal{O}_1$ and its vocabulary $\mathbf{N}_C$, $\mathbf{N}_P$ and $\mathbf{N}_I$, the following Algorithm A-1 calculates $\mathbf{ES}(\mathcal{O}_1, \text{DL-Lite})$; Algorithm A-1 uses Algorithm A-2 and entailment-check$(\mathcal{O}_1, \alpha)$ to calculate $\mathbf{AS}(\text{DL-Lite}, \mathbf{N}_C, \mathbf{N}_P, \mathbf{N}_I)$ and check $\mathcal{L}_1$ entailments, respectively. In this paper, we set $\mathcal{L}_1$ as OWL DL, while in general it can be any DL.

Algorithm A-1: DL-Lite-ES($\mathcal{O}_1$,$\mathbf{N}_C, \mathbf{N}_P, \mathbf{N}_I$)

```
1: let $ES := \emptyset$ //initialise ES
2: let $AS :=$ DL-Lite-AS($\mathbf{N}_C, \mathbf{N}_P, \mathbf{N}_I$) //calculate AS
3: for every axiom $\alpha$ in AS do
4:     if entailment-check($\mathcal{O}_1, \alpha$) = true then
5:         $ES := ES \cup \{\alpha\}$
6:     end if
7: end for//call an $\mathcal{L}_1$ reasoner to check if $\mathcal{O}_1 \models \alpha$
8: return $ES$
```

Algorithm A-2: DL-Lite-AS($\mathcal{O}_1$,$\mathbf{N}_C, \mathbf{N}_P, \mathbf{N}_I$)

```
1: let $AS := \emptyset$ //initialise AS
2: let $BS := \mathbf{N}_C$ //initialise BS, the set of basic classes
3: for every property name $R$ in $\mathbf{N}_P$ do
4:     $AS := AS \cup \{\text{Func}(R), \text{Func}(R^-)\}$ //add DL-Lite property axioms into $AS$
5:     $BS := BS \cup \{\exists R, \exists R^-\}$ //add existential class descriptions into $BS$
6: end for
7: let $CS := BS$ //initialise CS, the set of general classes
8: for every class $B$ in $BS$ do
9:     $CS := CS \cup \{\neg B\}$
10: end for
11: for every class $B$ in $BS$ do
12:     for every class $C$ in $CS$ do
13:         $AS := AS \cup \{B \sqsubseteq C\}$ //add DL-Lite class axioms into $AS$
14:     end for
15: end for
16: for every class $C$ in $CS$ do
17:     for every individual name a in $\mathbf{N}_I$ do
18:         $AS := AS \cup \{C(\mathtt{a})\}$ //add DL-Lite class assertions into $AS$
19:     end for
20: end for
21: for every property $R$ in $\mathbf{N}_P$ do
22:     for every individual name a in $\mathbf{N}_I$ do
23:         for every individual name b in $\mathbf{N}_I$ do
24:             $AS := AS \cup \{R(\mathtt{a},\mathtt{b}), R(\mathtt{b},\mathtt{a}), R^-(\mathtt{a},\mathtt{b}), R^-(\mathtt{b},\mathtt{a})\}$ //add DL-Lite property assertions into $AS$
25:         end for
26:     end for
27: end for
28: return $AS$
```

The Algorithm A-2 needs some explanation and clarification. BS and CS are the sets of basic classes and general classes in DL-Lite, respectively. As DL-Lite class axioms can be normalised into the following two forms: $B_1 \sqsubseteq B_2$ and $B_1 \sqsubseteq \neg B_2$ (Calvanese *et al.* 2005),[1] we do not have to consider class conjunctions when we construct CS (lines 7-10). Algorithm A-2 shows $\mathbf{AS}(\text{DL-Lite}, \mathbf{N}_C, \mathbf{N}_P, \mathbf{N}_I)$ consists of $2 \cdot |\mathbf{N}_P|$ property axioms, $2 \cdot (|\mathbf{N}_C| + 2 \cdot |\mathbf{N}_P|)^2$ class axioms and $2 \cdot (|\mathbf{N}_C| + 2 \cdot |\mathbf{N}_P|) \cdot |\mathbf{N}_I| + |\mathbf{N}_P| \cdot |\mathbf{N}_I|^2$ individual axioms. In other words, the size of $\mathbf{AS}(\text{DL-Lite}, \mathbf{N}_C, \mathbf{N}_P, \mathbf{N}_I)$ is polynomial w.r.t. those of $\mathbf{N}_C$, $\mathbf{N}_P$ and $\mathbf{N}_I$.

Query Answering over Entailment Sets

In this sub-section, we investigate query answering over entailment sets. In order to evaluate from the theoretical point of view query answering based on semantic approximation, we are mostly interested in the soundness and completeness of the solutions. Given an OWL DL ontology $\mathcal{O}_1$ and an arbitrary query q, we denote the set of solutions of q over $\mathcal{O}_1$ as $\mathbf{S}_{q,\mathcal{O}_1}$ and the set of solutions of q over $\mathbf{ES}(\mathcal{O}_1, \text{DL-Lite})$ as $\mathbf{S}_{q,\mathbf{ES}(\mathcal{O}_1,\text{DL-Lite})}$. We say $\mathbf{S}_{q,\mathbf{ES}(\mathcal{O}_1,\text{DL-Lite})}$ is sound if $\mathbf{S}_{q,\mathbf{ES}(\mathcal{O}_1,\text{DL-Lite})} \subseteq \mathbf{S}_{q,\mathcal{O}_1}$. We say $\mathbf{S}_{q,\mathbf{ES}(\mathcal{O}_1,\text{DL-Lite})}$ is complete if $\mathbf{S}_{q,\mathcal{O}_1} \subseteq \mathbf{S}_{q,\mathbf{ES}(\mathcal{O}_1,\text{DL-Lite})}$. We say $\mathbf{S}_{q,\mathbf{ES}(\mathcal{O}_1,\text{DL-Lite})}$ is sound and complete if $\mathbf{S}_{q,\mathbf{ES}(\mathcal{O}_1,\text{DL-Lite})} = \mathbf{S}_{q,\mathcal{O}_1}$.

Theorem 4 *Given an OWL DL ontology $\mathcal{O}_1$ and an arbitrary query q over $\mathcal{O}_1$, $\mathbf{S}_{q,\mathbf{ES}(\mathcal{O}_1,\textit{DL-Lite})}$ is sound.*

Proof: Immediate consequence of the definition of $\mathbf{ES}(\mathcal{O}_1, \text{DL-Lite})$ and the fact that OWL DL is monotonic. ∎

It remains to investigate how complete $\mathbf{S}_{q,\mathbf{ES}(\mathcal{O}_1,\text{DL-Lite})}$ is. In general, it is possibly incomplete (see the following example) as the complexity of query answering in DL-Lite is obviously lower than that of OWL DL.

[1]For example, $B \sqsubseteq C_1 \sqcap C_2$ can be normalised to $B \sqsubseteq C_1$ and $B \sqsubseteq C_1$.

Example 2 *Let us consider a query $q(x) \leftarrow R(x,y) \wedge C(y)$ over an OWL DL ontology $\mathcal{O}_1 = \{\exists R.C(\mathtt{a})\}$. Since $\mathcal{O}_1 \models q_{[x\mapsto \mathtt{a}]}$, we have $\mathbf{S}_{q,\mathcal{O}_1} = \{x \mapsto \mathtt{a}\}$.*

As the entailment set $\mathbf{ES}(\mathcal{O}_1, \text{DL-Lite}) = \{\exists R(\mathtt{a})\}$ does not entail $q_{[x\mapsto \mathtt{a}]}$, we have $\mathbf{S}_{q,\mathbf{ES}(\mathcal{O}_1,\text{DL-Lite})} = \emptyset$.

However, if we transform $q(x) \leftarrow R(x,y) \wedge C(y)$ in Example 2 into one without non-distinguished variables, we can simply add y into the head: $q'(x,y) \leftarrow R(x,y) \wedge C(y)$. Now we have $\mathbf{S}_{q',\mathcal{O}_1} = \emptyset$ because, given $\mathcal{O}_1$, it is not possible to evaluate the variable y to any named individuals. The following theorem shows that $\mathbf{S}_{q,\mathbf{ES}(\mathcal{O}_1,\text{DL-Lite})}$ is complete for arbitrary conjunctive queries q without non-distinguished variables. Intuitively, this is because all atoms of the form $\mathsf{A}(\mathtt{e})$ and $R(\mathtt{e},\mathtt{f})$, where $\mathtt{e}$ and $\mathtt{f}$ are named individuals, that are entailed by $\mathcal{O}_1$ are stored in $\mathbf{ES}(\mathcal{O}_1, \text{DL-Lite})$.

Theorem 5 *Given a consistent OWL DL ontology $\mathcal{O}_1$ and an arbitrary conjunctive query q over $\mathcal{O}_1$ that has no non-distinguished variables, $\mathbf{S}_{q,\mathbf{ES}(\mathcal{O}_1,\text{DL-Lite})} = \mathbf{S}_{q,\mathcal{O}_1}$.*

Proof (sketch) : Let q be of the form $q(X) \leftarrow conj(X,\emptyset,Z)$, where $conj(X,\emptyset,Z)$ is a conjunction of atoms which are of the form $\mathsf{A}(v)$ or $R(v_1,v_2)$, A is a class name, R is a property name and all v, v_1, v_2 are either variables in X or individual names in Z. For each solution $S \in \mathbf{S}_{q,\mathcal{O}_1}$, we have $\mathcal{O}_1 \models q_{[X\mapsto S]}$; i.e., for each atom $\mathsf{A}(v)$ ($R(v_1,v_2)$), we have an evaluation $v \mapsto \mathtt{o}$ ($v_1 \mapsto \mathtt{o}_1, v_2 \mapsto \mathtt{o}_2$) such that $\mathcal{O}_1 \models \mathsf{A}(o)$ ($\mathcal{O}_1 \models R(\mathtt{o}_1,\mathtt{o}_2)$). Due to the definition of entailment sets, we have $\mathbf{ES}(\mathcal{O}_1, \text{DL-Lite}) \models \mathsf{A}(\mathtt{o})$ ($\mathbf{ES}(\mathcal{O}_1, \text{DL-Lite}) \models R(\mathtt{o}_1,\mathtt{o}_2)$). Therefore, we have $\mathbf{ES}(\mathcal{O}_1, \text{DL-Lite}) \models q_{[X\mapsto S]}$; i.e., $S \in \mathbf{S}_{q,\mathbf{ES}(\mathcal{O}_1,\text{DL-Lite})}$. Similarly, we can show, for each solution $S \in \mathbf{S}_{q,\mathbf{ES}(\mathcal{O}_1,\text{DL-Lite})}$, we have $S \in \mathbf{S}_{q,\mathcal{O}_1}$. ■

It is important to point out that the restriction of disallowing non-distinguished variables is not really unusual: (1) the well known traditional DL reasoner Racer (supporting both TBox and ABox) does not allow them in its query language nRQL (Haarslev, Moller, & Wessel); (2) the KAON2 DL reasoner (Hustadt, Motik, & Sattler 2004), which is mainly designed to provide efficient ABox reasoning and query answering, does not support non-distinguished variables either. To the best of our knowledge, this restriction is the price we should pay for scalable query answering over ontologies.

Optimisations

The results in the previous section indicate SQL engines are sufficient to answer disjunctive queries over the target DL-Lite ontologies, since all entailments, of a source OWL DL ontology, that are expressible in DL-Lite are kept in the target ontology. Some early evaluation showed that it could take a considerable amount of time to calculate the entailment set if the source ontology is huge. In this sub-section, we investigate some possible answers of the two questions: (1) Can we reduce the time to calculate the entailment set by avoiding some unnecessary entailment tests? (2) Is it possible to reduce the storage of redundant axioms?

Firstly, we can optimise class subsumption checking by making use of the classification reasoning service. Namely, for each general class $C \in CS$ (see Algorithm A-2), we add the definition $\mathsf{A}_C \equiv C$ into $\mathcal{O}_1$; we call the extended ontology $\mathcal{O}_{1'}$. It is important to note that adding such definitions does not change the semantics of the ontology $\mathcal{O}_1$, but simply introduce names for the DL-Lite general classes. Accordingly, for any pair of classes $C, D \in CS$, $C \sqsubseteq D$ iff $\mathsf{A}_C \sqsubseteq \mathsf{A}_D$, which can be checked by using the classification service.

Secondly, we can make use of the entailed class hierarchy of $\mathcal{O}_{1'}$ to optimise class assertion checking. Given the entailed class hierarchy $\mathcal{H}$ of $\mathcal{O}_{1'}$, the set of general DL-Lite classes (w.r.t. $\mathcal{O}_1$) CS and the set of individual names $\mathbf{N}_I$ in $\mathcal{O}_1$, Algorithm A-3 calculates the set of class assertions in $\mathbf{ES}(\mathcal{O}_1, \text{DL-Lite})$.

Algorithm A-3: DL-Lite-ES-CA($\mathcal{H}$,CS, $\mathbf{N}_I$)

```
1: let $CA := \emptyset$ //initialise CA
2: for every individual name o in $\mathbf{N}_I$ do
3:    init(check-queue)
4:    Checked := $\emptyset$
5:    push (check-queue, leaf-nodes($\mathcal{H}$)) //check leaf-nodes first
6:    while (A:=pop(check-queue)) do
7:       if A $\notin$ Checked then
8:          if entailment-check($\mathcal{O}_1$, A(o)) = true then
9:             $CA := CA \cup \{\mathsf{A}(\mathtt{o})\}$ //add A(o) into $CA$
10:            Ancestors := ancestors($\mathcal{H}$, A)
11:            for every class name N in Ancestors do
12:               $CA := CA \cup \{\mathsf{N}(\mathtt{o})\}$ //and related axioms for
                  all ancestors of A
13:            end for
14:            Checked := Checked $\cup$ Ancestors //ancestors are
               marked as checked
15:         else
16:            push (check-queue, parents($\mathcal{H}$,A)) //otherwise, add
               parents in the queue
17:         end if
18:      end if
19:   end while
20: end for
21: return $CA$
```

The basic idea of Algorithm A-3 is that if $\mathcal{O}_1 \models \mathsf{A}(\mathtt{o})$, then we have $\mathcal{O}_1 \models \mathsf{N}(\mathtt{o})$ for any $\mathsf{A} \sqsubseteq \mathsf{N}$ (according to $\mathcal{H}$), which reduces the number of OWL DL entailment checking. More specifically, for each individual in $\mathbf{N}_I$, we initialise the check-queue (making it empty), and then push the leaf nodes of $\mathcal{H}$ into the check-queue. If $\mathcal{O}_1 \not\models \mathsf{A}(\mathtt{o})$ (A is the head of the queue), we add parents of A into the check-queue; otherwise, we add the related entailed axioms into CA.

Thirdly, we can make use of the instance of classes of the forms $\exists R$ and $\exists R^-$ to optimise property assertion checking. Given the set $\mathbf{N}_P$ of property names and the set $\mathbf{N}_I$ of individual names in $\mathcal{O}_1$, Algorithm A-4 calculates the set of property assertions in $\mathbf{ES}(\mathcal{O}_1, \text{DL-Lite})$.

Algorithm A-4: DL-Lite-ES-PA($\mathbf{N}_P$, $\mathbf{N}_I$)

```
1: let $PA := \emptyset$ //initialise CA
2: for every property name $R$ in $\mathbf{N}_P$ do
```

```
3:      candidate-1 := instance-of($\exists R$)
4:      candidate-2 := instance-of($\exists R^-$)
5:      for every individual a in candidate-1 do
6:          for every individual b in candidate-1 do
7:              if entailment-check($\mathcal{O}_1, R(\mathtt{a}, \mathtt{b})$) = true then
8:                  $PA := PA \cup \{R(\mathtt{a}, \mathtt{b}), R^-(\mathtt{b}, \mathtt{a})\}$
9:              end if
10:         end for
11:     end for
12: end for
13: return $PA$
```

The basic idea of Algorithm A-4 is that in order to calculate instances of a property R, we do not have to try every possible combinations of individuals pairs. Indeed, we can simply consider individuals which are instances of $\exists R$ as candidates of the first argument, and similarly only consider individuals which are instances of $\exists R^-$ as candidates of the second argument.

In order to reduce the storage of redundant individual axioms, we can further optimise Algorithm A-3 by revising line 12 so as to add $\mathsf{N}(\mathtt{o})$ into CA where N is defined as $\exists R$ or $\exists R^-$ which are needed in Algorithm A-4. In this setting, we need a DL-Lite reasoner rather than an SQL engine alone to answer queries against the target ontology.

Implementation and Evaluation

In this section, we evaluate: (1) how effective the optimised algorithms reduce the time for calculating target ontologies; (2) how efficient query answering over target ontologies can be. The evaluation is based on the ONTOSEARCH2 system (Pan & Thomas 2006), which supports ontology searching and querying. The core of ONTOSEARCH2 is an inference engine for the DL-Lite ontology language. ONTOSEARCH2 implements the semantic approximation algorithms and the optimisations techniques (both A-3 and its further optimisation) presented in the previous section. In this section, we label the implementation of A-3 as *the DB version* and its further optimisation as the *DL-Lite version*. The machine that we used is a Dell Precision 370, with Intel Pentium 4 3.0GHz, 1024Mb RAM and 80Gb (ATA) hard disc. We used PELLET 1.3 (Pellet 2003) as the OWL DL reasoner and PostgreSQL 8.1 as the RDBMS.

Optimisation Algorithms

We use the Wine Ontology (Smith, Welty, & McGuinness 2004) and the Lehigh University Benchmark (LUBM) (Guo, Pan, & Heflin 2005) to evaluate the optimisations. The wine ontology contains 79 named classes, 12 individual properties and 206 individuals. The naive Algorithms A-1 and A-2 did not complete within the one hour time limit we had set for this experiment. It took the DL-Lite version only 4 minutes and 14 seconds. For LUBM, we used a dataset containing 6,888,642 individuals (in the case 50 universities). To calculate the target ontology, naive algorithms ran out of memory, while it took the DL-Lite version about 3 hours. We also compare the DB and DL-Lite versions of the optimisations: the DL-Lite version requires about half of the space and time than the DB version needs to construct target

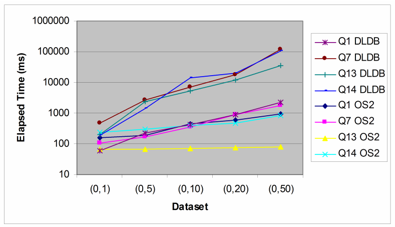

Figure 1: Performance of ONTOSEARCH2 (OS2) compared to DLDB system as ABox size increases using Lehigh University Benchmark

ontologies, while queries executed takes about twice as long as the DB version.

Query Answering

We evaluate the ONTOSEARCH2 system (the DB version) on query answering using LUBM, using generated data sets representing 1, 5, 10, 20 and 50 universities, these are generated using the same seed and index values as used in (Guo, Pan, & Heflin 2004) so we can directly compare the performance of the ONTOSEARCH2 system against the systems (which are incomplete) tested in (Guo, Pan, & Heflin 2004). The smallest dataset (0,1) contains approximately 136,000 triples in 14 RDF files, and the largest dataset (0,50) contains approximately 6,800,000 triples in 999 RDF files. Figure 1 shows how the performance of ONTOSEARCH2 changes with 4 different queries from the LUBM query set.

Furthermore, we also compare ONTOSEARCH2 with the KAON2 (Hustadt, Motik, & Sattler 2004), Pellet (Pellet 2003) and Racer systems (Haarslev & Möller 2001), which provide sound and complete answers and are benchmarked using the LUBM on similar harware in (Motik & Sattler 2006). The two tableaux based reasoners, Pellet and Racer, were unable to complete queries 1, 2 or 3 when more than 2 universities made up the dataset (lubm[0,2]). A comparison with KAON2 again shows the advantage of the approximation based approach: (1) ONTOSEARCH2 is able to complete the queries faster than the KAON2 on all datasets that KAON2 can handle; (2) ONTOSEARCH2 is able to complete the queries using dataset(0,50), while in the cited paper, KAON2 is only able to perform the queries using dataset (0,4) within the constraints.

Related Work

Existing approaches (Stuckenschmidt & van Harmelen 2002; Wache, Groot, & Stuckenschmidt 2005; Groot, Stuckenschmidt, & Wache 2005; Hurtado, Poulovassilis, & Wood 2006) are mainly based on syntactic approximation of ontological axioms and queries. Stuckenschmidt and van Harmelen (Stuckenschmidt & van Harmelen 2002) were probably the first to discuss approximating query answering over Web ontologies. Groot et al. (Groot, Stuckenschmidt, & Wache 2005) reported negative results (in terms of scalability) on using Schaerf and Cadoli's method (Schaerf & Cadoli 1995) on approximating DL classification. Hitzler and Vrandečić (Hitzler & Vrandecic 2005) proposed an approximate ABox reasoning method based on the KAON2 reasoner by disregarding non-Horn features of OWL DL ontologies, with some positive result on scalability. Wache et al. (Wache, Groot, & Stuckenschmidt 2005) showed that Stuckenschmidt and van Harmelen's method, extended with a heuristic for sub- concept selection, has some potential for speeding up instance retrieval. In general, the contributions of syntactic approximation to scalable query answering has not yet been clarified. To the best of our knowledge, semantic approximation is the first approximation technique that guarantees soundness on DL query answering, with clear improvements on scalability. Even in knowledge compilation, lower-bound approximations do not guarantee soundness.

Conclusion and Outlook

In this paper, we have investigated how to recast the idea of knowledge compilation into semantically approximating OWL DL ontologies. Unlike existing syntactic approximation approach, our approach always guarantees sound answers for conjunctive and disjunctive queries over ontologies. For queries without non-distinguished variables (which modern OWL DL reasoners like Racer and KAON2 disallow anyway), our approach guarantees both soundness and completeness. Our preliminary evaluations in the ONTOSEARCH2 system shows that our approach is very scalable: ONTOSEARCH2 outperforms existing OWL DL reasoners significantly. The semantic approximation approach is very general; our results indicate that user can still enjoy efficient querying answering support when they use expressive ontology languages, such as OWL DL and OWL 1.1.

For our future work, one of the main challenges is to extend the query language to support class descriptions, datatype properties and some datatype predicates/built-ins. Secondly, we will investigate benchmarks for querying answering over (complex) ontologies. Furthermore, it is interesting to see how to make it more complete for queries with non-distinguished variables. Last but not least, we shall investigate further optimisations on calculating target ontologies and how to apply our semantic approximations in other light-weight ontology languages.

Acknowledgement This research was partially supported by (1) the EC project Knowledge Web (IST-2004-507842) and (2) the UK EPSRC Advanced Knowledge Technologies (GR/N15764/01) project. We would also like to thank Prof. Derek Sleeman for helpful discussions on related topics.

References

Baader, F.; McGuiness, D. L.; Nardi, D.; and Patel-Schneider, P., eds. 2002. *Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press.

Berners-Lee, T.; Hendler, J.; and Lassila, O. 2001. The Semantic Web. *Scientific American* 284(5):34–43.

Calvanese, D.; Giacomo, G. D.; Lenzerini, M.; Rosati, R.; and Vetere, G. 2004. DL-Lite: Practical Reasoning for Rich DLs. In *Proc. of the DL2004 Workshop*.

Calvanese, D.; Giacomo, G. D.; Lembo, D.; Lenzerini, M.; ; and Rosati, R. 2005. DL-Lite: Tractable description logics for ontologies. In *Proc. of AAAI 2005*.

Groot, P.; Stuckenschmidt, H.; and Wache, H. 2005. Approximating Description Logic Classification for Semantic Web Reasoning. In *Proc. of ESWC2005*.

Guo, Y.; Pan, Z.; and Heflin, J. 2004. An Evaluation of Knowledge Base Systems for Large OWL Datasets. In *Proc. of ISWC2004*, 274–288.

Guo, Y.; Pan, Z.; and Heflin, J. 2005. LUBM: A Benchmark for OWL Knowledge Base Systems. *Journal of Web Semantics* 3(2):158–182.

Haarslev, V., and Möller, R. 2001. RACER System Description. volume 2083.

Haarslev, V.; Moller, R.; and Wessel, M. Querying the Semantic Web with Racer + nRQL. In *Proc. of the KI-04 Workshop on Applications of Description Logics 2004*.

Hitzler, P., and Vrandecic, D. 2005. Resolution-Based Approximate Reasoning for OWL DL. In *Proc. of the 4th International Semantic Web Conference (ISWC2005)*.

Hurtado, C.; Poulovassilis, A.; and Wood, P. 2006. A Relaxed Approach to RDF Querying. In *Proc. of the 5th International Semantic Web Conference (ISWC-2006)*.

Hustadt, U.; Motik, B.; and Sattler, U. 2004. Reducing $\mathcal{SHIQ}^-$ Description Logic to Disjunctive Datalog Programs. In *Proc. of KR2004*, 152–162.

Motik, B., and Sattler, U. 2006. A Comparison of Reasoning Techniques for Querying Large Description Logic ABoxes. In *Proc. of LPAR 2006*, 227–241.

Pan, J., and Thomas, E. 2006. ONTOSEARCH2: Searching and Querying Web Ontologies. In *Proc. of WWW/Internet 2006*, 211–218.

Patel-Schneider, P. F.; Hayes, P.; and Horrocks, I. 2004. OWL Web Ontology Language Semantics and Abstract Syntax. Technical report, W3C. W3C Recommendation.

Pellet. 2003. `http://www.mindswap.org/2003/pellet/`.

Schaerf, M., and Cadoli, M. 1995. Tractable reasoning via approximation. *Artificial Intelligence* 74:249–310.

Selman, B., and Kautz, H. 1996. Knowledge Compilation and Theory Approximation. *Journal of the ACM (JACM)* 43(2):193–224.

Smith, M. K.; Welty, C.; and McGuinness, D. L. 2004. OWL Web Ontology Language Guide. Technical report, W3C Recommendation.

Stuckenschmidt, H., and van Harmelen, F. 2002. Approximating Terminological Queries. In *Proc. of FQAS2002)*, 329–343.

Wache, H.; Groot, P.; and Stuckenschmidt, H. 2005. Scalable Instance Retrieval for the Semantic Web by Approximation. In *Proc. of WISE-2005 Workshop on Scalable Semantic Web Knowledge Base Systems*.