Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence

Conjunctive Query Inseparability of *OWL 2 QL* TBoxes

B. Konev,[1] R. Kontchakov,[2] M. Ludwig,[1] T. Schneider,[3] F. Wolter[1] and M. Zakharyaschev[2]

[1] Department of Computer Science
University of Liverpool, UK
–konev,michel.ludwig,wolter˝
@liverpool.ac.uk

[2] Department of CS & IS
Birkbeck College London, UK
–roman,michael˝
@dcs.bbk.ac.uk

[3] Department of Computer Science
Universität Bremen, Germany
tschneider
@informatik.uni-bremen.de

Abstract

The *OWL 2* profile *OWL 2 QL*, based on the *DL-Lite* family of description logics, is emerging as a major language for developing new ontologies and approximating the existing ones. Its main application is ontology-based data access, where ontologies are used to provide background knowledge for answering queries over data. We investigate the corresponding notion of query inseparability (or equivalence) for *OWL 2 QL* ontologies and show that deciding query inseparability is PSPACE-hard and in EXPTIME. We give polynomial time (incomplete) algorithms and demonstrate by experiments that they can be used for practical module extraction.

Introduction

In recent years, ontology-based data access (OBDA) has emerged as one of the most interesting and challenging applications of description logic (Dolby et al. 2008; Heymans et al. 2008; Poggi et al. 2008). The key idea is to use ontologies for enriching data with additional background knowledge, and thereby enable query answering over incomplete and semistructured data from heterogeneous sources via a high-level conceptual interface. The W3C recognised the importance of OBDA by including in the *OWL 2* Web Ontology Language the profile *OWL 2 QL*, which was designed for OBDA with standard relational database systems. *OWL 2 QL* is based on a description logic (DL) that was originally introduced under the name *DL-Lite*$_{\mathcal{R}}$ (Calvanese et al. 2006; 2007) and called *DL-Lite*$^{\mathcal{H}}_{core}$ in the more general classification of (Artale et al. 2009). It can be described as an optimal sub-language of the DL $\mathcal{SROIQ}$, underlying *OWL 2*, which includes most of the features of conceptual models, and for which conjunctive query answering can be done in AC^0 for data complexity.

Thus, *DL-Lite*$^{\mathcal{H}}_{core}$ is becoming a major language for developing ontologies, and a target language for translation and approximation of existing ontologies formulated in more expressive DLs (Pan and Thomas 2007; Botoeva, Calvanese, and Rodriguez-Muro 2010). One of the consequences of this development is that *DL-Lite*$^{\mathcal{H}}_{core}$ ontologies turn out to be larger and more complex than originally envisaged. As a result, reasoning support for ontology engineering tasks such as composing, re-using, comparing, and extracting ontologies—which so far has been only analysed for expressive DLs (Cuenca Grau et al. 2008; Stuckenschmidt, Parent, and Spaccapietra 2009), $\mathcal{EL}$ (Lutz and Wolter 2010) and *DL-Lite* dialects (Kontchakov, Wolter, and Zakharyaschev 2010) without role inclusions—is becoming increasingly important for *DL-Lite*$^{\mathcal{H}}_{core}$ as well.

In the context of OBDA, the basic notion underlying many ontology engineering tasks is Σ-*query inseparability*: for a signature (a set of concept and role names) Σ, two ontologies are deemed to be inseparable if they give the same answers to any conjunctive query over any data formulated in Σ. Thus, in applications using Σ-queries and data, one can safely replace any ontology by a Σ-query inseparable one. Note that the relativisation to Σ is very important here. For example, one cannot expect modules of an ontology to be query inseparable from the whole ontology for *arbitrary* queries and data sets, whereas this should be the case if we restrict the query and data language to the module's signature or a specified subset thereof. Similarly, when comparing two versions of one ontology, the subtle and potentially problematic differences are those that concern queries over their common symbols, rather than all symbols occurring in these versions. In applications where ontologies are built using imported parts, a stronger notion of inseparability is required: two ontologies are *strongly* Σ-*query inseparable* if they give the same answers to Σ-queries and data when imported to an arbitrary context ontology formulated in Σ.

The aim of this paper is to (*i*) investigate the computational complexity of deciding (strong) Σ-query inseparability for *DL-Lite*$^{\mathcal{H}}_{core}$ ontologies, (*ii*) develop efficient (though incomplete) algorithms for practical inseparability checking, and (*iii*) analyse the performance of the algorithms for the challenging task of minimal module extraction.

One of our surprising discoveries is that the analysis of Σ-query inseparability for (seemingly 'harmless' and computationally well-behaved) *DL-Lite*$^{\mathcal{H}}_{core}$ ontologies requires drastically different logical tools compared with the previously considered DLs. It turns out that the new syntactic ingredient—the interaction of role inclusions and inverse roles—makes deciding (strong) query inseparability PSPACE-hard, as opposed to the known CONP and Π^p_2-completeness results for *DL-Lite* dialects without role inclusions (Kontchakov, Wolter, and Zakharyaschev 2010). On the other hand, the obtained EXPTIME upper bound is actually the first known decidability result for strong inseparability, which goes beyond the 'essentially' Boolean logic and might additionally indicate a way of solving the open problem of strong Σ-query inseparability for $\mathcal{EL}$ (Lutz and Wolter 2010). For *DL-Lite*$_{core}$ ontologies (*without* role inclusions), strong Σ-query inseparability is shown to be only NLOGSPACE-complete. We give (incomplete) polynomial time algorithms checking (strong) Σ-inseparability and demonstrate, by a set of minimal module extraction experiments, that they are (*i*) complete for many existing *DL-Lite*$^{\mathcal{H}}_{core}$ ontologies and signatures, and (*ii*) sufficiently fast to be used in module extraction algorithms that require thousands of Σ-query inseparability checks. All omitted proofs can be found at www.dcs.bbk.ac.uk/~roman.

Σ-Query Entailment and Inseparability

We begin by formally defining the description logic *DL-Lite*$^{\mathcal{H}}_{core}$, underlying *OWL 2 QL*, and the notions of Σ-query inseparability and Σ-query entailment. The language of *DL-Lite*$^{\mathcal{H}}_{core}$ contains countably infinite sets of *individual names* a_i, *concept names* A_i, and *role names* P_i. *Roles* R and *concepts* B of this language are defined by:

$$R \quad ::= \quad P_i \quad | \quad P_i^-,$$
$$B \quad ::= \quad \bot \quad | \quad \top \quad | \quad A_i \quad | \quad \exists R.$$

A *DL-Lite*$^{\mathcal{H}}_{core}$ *TBox*, $\mathcal{T}$, is a finite set of *inclusions*

$$B_1 \sqsubseteq B_2, \;\; R_1 \sqsubseteq R_2, \;\; B_1 \sqcap B_2 \sqsubseteq \bot, \;\; R_1 \sqcap R_2 \sqsubseteq \bot,$$

where B_1, B_2 are concepts and R_1, R_2 roles. An *ABox*, $\mathcal{A}$, is a finite set of *assertions* of the form $B(a_i)$, $R(a_i, a_j)$ and $a_i \neq a_j$, where a_i and a_j are individual names, B a concept and R a role. $\mathsf{Ind}(\mathcal{A})$ will stand for the set of individual names occurring in $\mathcal{A}$. Taken together, $\mathcal{T}$ and $\mathcal{A}$ constitute the *DL-Lite*$^{\mathcal{H}}_{core}$ *knowledge base* (KB, for short) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. The sub-language of *DL-Lite*$^{\mathcal{H}}_{core}$ without role inclusions $R_1 \sqsubseteq R_2$ is denoted by *DL-Lite*$_{core}$ (Calvanese et al. 2007).

The semantics of *DL-Lite*$^{\mathcal{H}}_{core}$ is defined as usual in DL (Baader et al. 2003). We only note that, in interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, we do not have to comply with the unique name assumption, that is, we can have $a_i^{\mathcal{I}} = a_j^{\mathcal{I}}$ for $i \neq j$. We write $\mathcal{I} \models \alpha$ to say that an inclusion or assertion α is true in $\mathcal{I}$. The interpretation $\mathcal{I}$ is a *model* of a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \alpha$ for all $\alpha \in \mathcal{T} \cup \mathcal{A}$. $\mathcal{K}$ is *consistent* if it has a model. A concept B is said to be $\mathcal{T}$*-consistent* if $(\mathcal{T}, \{B(a)\})$ has a model. $\mathcal{K} \models \alpha$ means that $\mathcal{I} \models \alpha$ for all models $\mathcal{I}$ of $\mathcal{K}$.

A *conjunctive query* (CQ) $\boldsymbol{q}(x_1, \dots, x_n)$ is a first-order formula $\exists y_1 \dots \exists y_m \, \varphi(x_1, \dots, x_n, y_1, \dots, y_m)$, where φ is constructed, using only $\wedge$, from atoms of the form $B(t)$ and $R(t_1, t_2)$, with B being a concept, R a role, and t_i being an individual name or a variable from the list $x_1, \dots, x_n, y_1, \dots, y_m$. The variables in $\vec{x} = x_1, \dots, x_n$ are called *answer variables* of $\boldsymbol{q}$. We say that an n-tuple $\vec{a} \subseteq \mathsf{Ind}(\mathcal{A})$ is an *answer* to $\boldsymbol{q}$ in an interpretation $\mathcal{I}$ if $\mathcal{I} \models \boldsymbol{q}[\vec{a}]$ (here we regard $\mathcal{I}$ to be a first-order structure); $\vec{a}$ is a *certain answer* to $\boldsymbol{q}$ over a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \boldsymbol{q}[\vec{a}]$ for all models $\mathcal{I}$ of $\mathcal{K}$; in this case we write $\mathcal{K} \models \boldsymbol{q}[\vec{a}]$.

To define the main notions of this paper, consider two KBs $\mathcal{K}_1 = (\mathcal{T}_1, \mathcal{A})$ and $\mathcal{K}_2 = (\mathcal{T}_2, \mathcal{A})$. For example, the $\mathcal{T}_i$ are different versions of some ontology, or one of them is a refinement of the other by means of new axioms. The question we are interested in is whether they give the same answers to queries formulated in a certain signature, say, in the common vocabulary of the $\mathcal{T}_i$ or in a vocabulary relevant to an application. To be precise, by a *signature*, Σ, we understand any finite set of concept and role names. A concept (inclusion, TBox, etc.) all concept and role names of which are in Σ is called a Σ*-concept* (*inclusion*, etc.). We say that $\mathcal{K}_1$ Σ*-query entails* $\mathcal{K}_2$ if, for *all* Σ-queries $\boldsymbol{q}(\vec{x})$ and all $\vec{a} \subseteq \mathsf{Ind}(\mathcal{A})$, $\mathcal{K}_2 \models \boldsymbol{q}[\vec{a}]$ implies $\mathcal{K}_1 \models \boldsymbol{q}[\vec{a}]$. In other words: any certain answer to a Σ-query given by $\mathcal{K}_2$ is also given by $\mathcal{K}_1$.

As the ABox is typically not fixed or known at the ontology design stage, we may have to compare the TBoxes over *arbitrary* Σ-ABoxes rather than a fixed one, which gives the following central definition of this paper.

Definition 1. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be TBoxes and Σ a signature. $\mathcal{T}_1$ Σ*-query entails* $\mathcal{T}_2$ if $(\mathcal{T}_1, \mathcal{A})$ Σ-query entails $(\mathcal{T}_2, \mathcal{A})$ for any Σ-ABox $\mathcal{A}$. $\mathcal{T}_1$ and $\mathcal{T}_2$ are Σ*-query inseparable* if they Σ-query entail each other, in which case we write $\mathcal{T}_1 \equiv_\Sigma \mathcal{T}_2$.

In many applications, Σ-query inseparability is enough to ensure that $\mathcal{T}_1$ can be safely replaced by $\mathcal{T}_2$. However, if they are developed as part of a larger ontology or are meant to be imported in other ontologies, a stronger notion is required:

Definition 2. $\mathcal{T}_1$ *strongly* Σ*-query entails* $\mathcal{T}_2$ if $\mathcal{T} \cup \mathcal{T}_1$ Σ-query entails $\mathcal{T} \cup \mathcal{T}_2$, for all Σ-TBoxes $\mathcal{T}$. $\mathcal{T}_1$ and $\mathcal{T}_2$ are *strongly* Σ*-query inseparable* if they strongly Σ-query entail each other, in which case we write $\mathcal{T}_1 \equiv^s_\Sigma \mathcal{T}_2$.

The following example illustrates the difference between Σ-query and strong Σ-query inseparability. For further discussion and examples, we refer the reader to (Cuenca Grau et al. 2008; Kontchakov, Wolter, and Zakharyaschev 2010).

Example 3. Let $\mathcal{T}_2 = \{\top \sqsubseteq \exists R, \exists R^- \sqsubseteq B, B \sqcap A \sqsubseteq \bot\}$, $\mathcal{T}_1 = \emptyset$ and $\Sigma = \{A\}$. $\mathcal{T}_1$ and $\mathcal{T}_2$ are Σ-query inseparable. However, they are not strongly Σ-query inseparable. Indeed, for the Σ-TBox $\mathcal{T} = \{\top \sqsubseteq A\}$, $\mathcal{T}_1 \cup \mathcal{T}$ is consistent, while $\mathcal{T}_2 \cup \mathcal{T}$ is inconsistent, and so $\mathcal{T}_1 \cup \mathcal{T}$ does not Σ-query entail $\mathcal{T}_2 \cup \mathcal{T}$, as witnessed by the query $\boldsymbol{q} = \bot$.

From now on, we shall focus our attention mainly on the more basic notion of Σ-query entailment.

Σ-Query Entailment and Σ-Homomorphisms

In this section, we characterise Σ-query entailment between *DL-Lite*$^{\mathcal{H}}_{core}$ TBoxes semantically in terms of (partial) Σ-homomorphisms between certain canonical models. Then, in the next section, we use this characterisation to investigate the complexity of deciding Σ-query entailment.

The *canonical model*, $\mathcal{M}_{\mathcal{K}}$, of a consistent KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ gives correct answers to all CQs. In general, $\mathcal{M}_{\mathcal{K}}$ is infinite; however, it can be folded up into a small *generating model* $\mathcal{G}_{\mathcal{K}} = (\mathcal{I}_{\mathcal{K}}, \leadsto_{\mathcal{K}})$ consisting of a finite interpretation $\mathcal{I}_{\mathcal{K}}$ and a *generating relation* $\leadsto_{\mathcal{K}}$ that defines the unfolding.

Let $\sqsubseteq^*_{\mathcal{T}}$ be the reflexive and transitive closure of the role inclusion relation given by $\mathcal{T}$, and let $[R] = \{S \mid R \equiv^*_{\mathcal{T}} S\}$, where $R \equiv^*_{\mathcal{T}} S$ stands for '$R \sqsubseteq^*_{\mathcal{T}} S$ and $S \sqsubseteq^*_{\mathcal{T}} R$.' We write $[R] \leq_{\mathcal{T}} [S]$ if $R \sqsubseteq^*_{\mathcal{T}} S$; thus, $\leq_{\mathcal{T}}$ is a partial order on the set $\{[R] \mid R \text{ a role in } \mathcal{T}\}$. For each $[R]$, we introduce a *witness* $w_{[R]}$ and define a *generating relation* $\leadsto_{\mathcal{K}}$ on the set of these witnesses together with $\mathsf{Ind}(\mathcal{A})$ by taking:

$a \rightsquigarrow_{\mathcal{K}} w_{[R]}$ if $a \in \mathsf{Ind}(\mathcal{A})$ and $[R]$ is $\leq_{\mathcal{T}}$-minimal such that $\mathcal{K} \models \exists R(a)$ and $\mathcal{K} \not\models R(a,b)$ for any $b \in \mathsf{Ind}(\mathcal{A})$;

$w_{[S]} \rightsquigarrow_{\mathcal{K}} w_{[R]}$ if $[R]$ is $\leq_{\mathcal{T}}$-minimal with $\mathcal{T} \models \exists S^- \sqsubseteq \exists R$ and $[S^-] \neq [R]$.

A role R is *generating in* $\mathcal{K}$ if there are $a \in \mathsf{Ind}(\mathcal{A})$ and $R_1, \dots, R_n = R$ such that $a \rightsquigarrow_{\mathcal{K}} w_{[R_1]} \rightsquigarrow_{\mathcal{K}} \cdots \rightsquigarrow_{\mathcal{K}} w_{[R_n]}$.

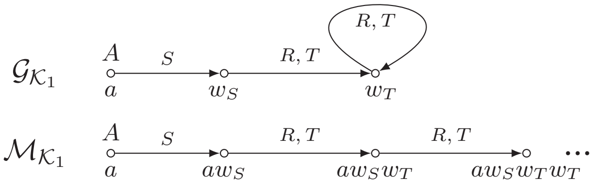
The interpretation $\mathcal{I}_{\mathcal{K}}$ is now defined as follows:

$$\begin{aligned}
\Delta^{\mathcal{I}_{\mathcal{K}}} &= \mathsf{Ind}(\mathcal{A}) \cup \{w_{[R]} \mid R \text{ is generating in } \mathcal{K}\},\\
a^{\mathcal{I}_{\mathcal{K}}} &= a, \text{ for all } a \in \mathsf{Ind}(\mathcal{A}),\\
A^{\mathcal{I}_{\mathcal{K}}} &= \{a \mid \mathcal{K} \models A(a)\} \cup \{w_{[R]} \mid \mathcal{T} \models \exists R^- \sqsubseteq A\},\\
P^{\mathcal{I}_{\mathcal{K}}} &= \{(a,b) \mid \text{there is } R(a,b) \in \mathcal{A} \text{ s.t. } R \sqsubseteq^*_{\mathcal{T}} P\} \cup\\
&\ \{(x, w_{[R]}) \mid x \rightsquigarrow_{\mathcal{K}} w_{[R]} \text{ and } [R] \leq_{\mathcal{T}} [P]\} \cup\\
&\ \{(w_{[R]}, x) \mid x \rightsquigarrow_{\mathcal{K}} w_{[R]} \text{ and } [R] \leq_{\mathcal{T}} [P^-]\}.
\end{aligned}$$

$\mathcal{G}_{\mathcal{K}}$ can be constructed in polynomial time in $|\mathcal{K}|$, and it is not hard to see that $\mathcal{I}_{\mathcal{K}} \models \mathcal{K}$. To construct the *canonical model* $\mathcal{M}_{\mathcal{K}}$ giving the correct answers to all CQs, we unfold the generating model $\mathcal{G}_{\mathcal{K}} = (\mathcal{I}_{\mathcal{K}}, \rightsquigarrow_{\mathcal{K}})$ along $\rightsquigarrow_{\mathcal{K}}$. A *path* in $\mathcal{G}_{\mathcal{K}}$ is a finite sequence $aw_{[R_1]} \cdots w_{[R_n]}$, $n \geq 0$, such that $a \in \mathsf{Ind}(\mathcal{A})$, $a \rightsquigarrow_{\mathcal{K}} w_{[R_1]}$ and $w_{[R_i]} \rightsquigarrow_{\mathcal{K}} w_{[R_{i+1}]}$, for $i < n$. Denote by $\mathsf{path}(\mathcal{G}_{\mathcal{K}})$ the set of all paths in $\mathcal{G}_{\mathcal{K}}$ and by $\mathsf{tail}(\sigma)$ the last element in $\sigma \in \mathsf{path}(\mathcal{G}_{\mathcal{K}})$. $\mathcal{M}_{\mathcal{K}}$ is defined by taking:

$$\begin{aligned}
\Delta^{\mathcal{M}_{\mathcal{K}}} &= \mathsf{path}(\mathcal{G}_{\mathcal{K}}),\\
a^{\mathcal{M}_{\mathcal{K}}} &= a, \text{ for all } a \in \mathsf{Ind}(\mathcal{A}),\\
A^{\mathcal{M}_{\mathcal{K}}} &= \{\sigma \mid \mathsf{tail}(\sigma) \in A^{\mathcal{I}_{\mathcal{K}}}\},\\
P^{\mathcal{M}_{\mathcal{K}}} &= \{(a,b) \in \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}) \mid (a,b) \in P^{\mathcal{I}_{\mathcal{K}}}\} \cup\\
&\ \{(\sigma, \sigma \cdot w_{[R]}) \mid \mathsf{tail}(\sigma) \rightsquigarrow_{\mathcal{K}} w_{[R]},\ [R] \leq_{\mathcal{T}} [P]\} \cup\\
&\ \{(\sigma \cdot w_{[R]}, \sigma) \mid \mathsf{tail}(\sigma) \rightsquigarrow_{\mathcal{K}} w_{[R]},\ [R] \leq_{\mathcal{T}} [P^-]\}.
\end{aligned}$$

Example 4. The models $\mathcal{G}_{\mathcal{K}_1}$ for $\mathcal{K}_1 = (\mathcal{T}_1, \{A(a)\})$ with $\mathcal{T}_1 = \{A \sqsubseteq \exists S, \exists S^- \sqsubseteq \exists T, \exists T^- \sqsubseteq \exists T, T \sqsubseteq R\}$, and $\mathcal{M}_{\mathcal{K}_1}$ look as follows ($\rightsquigarrow_{\mathcal{K}_1}$ in $\mathcal{G}_{\mathcal{K}_1}$ is depicted as $\rightarrow$):



Our first result states that $\mathcal{M}_{\mathcal{K}}$ gives correct answers to all conjunctive queries:

Theorem 5. *For all consistent DL-Lite$^{\mathcal{H}}_{core}$ KBs $\mathcal{K}$, CQs $\boldsymbol{q}(\vec{x})$ and tuples $\vec{a} \subseteq \mathsf{Ind}(\mathcal{A})$, where $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, we have $\mathcal{K} \models \boldsymbol{q}[\vec{a}]$ iff $\mathcal{M}_{\mathcal{K}} \models \boldsymbol{q}[\vec{a}]$.*

Thus, to decide Σ-query entailment between KBs $\mathcal{K}_1$ and $\mathcal{K}_2$, it suffices to check whether $\mathcal{M}_{\mathcal{K}_2} \models \boldsymbol{q}[\vec{a}]$ implies $\mathcal{M}_{\mathcal{K}_1} \models \boldsymbol{q}[\vec{a}]$ for all Σ-queries $\boldsymbol{q}(\vec{x})$ and tuples $\vec{a}$. This relationship between $\mathcal{M}_{\mathcal{K}_2}$ and $\mathcal{M}_{\mathcal{K}_1}$ can be characterised semantically in terms of finite Σ-homomorphisms.

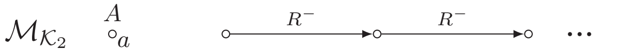
For an interpretation $\mathcal{I}$ and a signature Σ, the Σ-*types* $\boldsymbol{t}^{\mathcal{I}}_{\Sigma}(x)$ and $\boldsymbol{r}^{\mathcal{I}}_{\Sigma}(x,y)$, for $x, y \in \Delta^{\mathcal{I}}$, are given by:

$$\begin{aligned}
\boldsymbol{t}^{\mathcal{I}}_{\Sigma}(x) &= \{\Sigma\text{-concept } B \mid x \in B^{\mathcal{I}}\},\\
\boldsymbol{r}^{\mathcal{I}}_{\Sigma}(x,y) &= \{\Sigma\text{-role } R \mid (x,y) \in R^{\mathcal{I}}\}.
\end{aligned}$$

A Σ-*homomorphism* from an interpretation $\mathcal{I}$ to $\mathcal{I}'$ is a function $h\colon \Delta^{\mathcal{I}} \to \Delta^{\mathcal{I}'}$ such that $h(a^{\mathcal{I}}) = a^{\mathcal{I}'}$, for all individual names a interpreted in $\mathcal{I}$, $\boldsymbol{t}^{\mathcal{I}}_{\Sigma}(x) \subseteq \boldsymbol{t}^{\mathcal{I}'}_{\Sigma}(h(x))$ and $\boldsymbol{r}^{\mathcal{I}}_{\Sigma}(x,y) \subseteq \boldsymbol{r}^{\mathcal{I}'}_{\Sigma}(h(x), h(y))$, for all $x, y \in \Delta^{\mathcal{I}}$.

It is well-known that answers to Σ-CQs are preserved under Σ-homomorphisms. Thus, if there is a Σ-homomorphism from $\mathcal{M}_{\mathcal{K}_2}$ to $\mathcal{M}_{\mathcal{K}_1}$, then $\mathcal{K}_1$ Σ-query entails $\mathcal{K}_2$. However, the converse does not hold in general.

Example 6. Take $\mathcal{T}_1$ from Example 4, and let $\mathcal{T}_2$ be the result of replacing R in $\mathcal{T}_1$ with R^-. Let $\Sigma = \{A, R\}$ and $\mathcal{K}_i = (\mathcal{T}_i, \{A(a)\})$. Then the Σ-reduct of $\mathcal{M}_{\mathcal{K}_1}$ does not contain a Σ-homomorphic image of the Σ-*reduct* of $\mathcal{M}_{\mathcal{K}_2}$, depicted below. On the other hand, it is easily seen that



$\mathcal{T}_1$ and $\mathcal{T}_2$ are Σ-query inseparable. Note that the Σ-reduct of $\mathcal{M}_{\mathcal{K}_2}$ contains points that are not reachable from the ABox by Σ-roles. In fact, using König's Lemma, one can show that if every point in $\mathcal{M}_{\mathcal{K}_2}$ is reachable from the ABox by a path of Σ-roles, then $\mathcal{K}_1$ Σ-query entails $\mathcal{K}_2$ iff there exists a Σ-homomorphism from $\mathcal{M}_{\mathcal{K}_2}$ to $\mathcal{M}_{\mathcal{K}_1}$.

Because of this, we say that $\mathcal{I}$ is *finitely Σ-homomorphically embeddable into* $\mathcal{I}'$ if, for every finite subinterpretation $\mathcal{I}_1$ of $\mathcal{I}$, there exists a Σ-homomorphism from $\mathcal{I}_1$ to $\mathcal{I}'$. Now one can show:

Theorem 7. *Let $\mathcal{K}_1$ and $\mathcal{K}_2$ be consistent DL-Lite$^{\mathcal{H}}_{core}$ KBs. Then $\mathcal{K}_1$ Σ-query entails $\mathcal{K}_2$ iff $\mathcal{M}_{\mathcal{K}_2}$ is finitely Σ-homomorphically embeddable into $\mathcal{M}_{\mathcal{K}_1}$.*

Theorem 7 does not yet give a satisfactory semantic characterisation of Σ-query entailment between TBoxes, as one still has to consider infinitely many Σ-ABoxes. However, using the fact that inclusions in *DL-Lite*$^{\mathcal{H}}_{core}$, different from disjointness axioms, involve only *one* concept or role in the left-hand side and making sure that the TBoxes entail the same Σ-inclusions, one can show that it is enough to consider *singleton* Σ-ABoxes of the form $\{B(a)\}$. Denote the models $\mathcal{G}_{(\mathcal{T},\{B(a)\})}$ and $\mathcal{M}_{(\mathcal{T},\{B(a)\})}$ by $\mathcal{G}^{B}_{\mathcal{T}}$ and $\mathcal{M}^{B}_{\mathcal{T}}$, respectively. We thus obtain the following characterisation of Σ-entailment between *DL-Lite*$^{\mathcal{H}}_{core}$ TBoxes $\mathcal{T}_1$, $\mathcal{T}_2$:

Theorem 8. *$\mathcal{T}_1$ Σ-query entails $\mathcal{T}_2$ iff*

(p) *$\mathcal{T}_2 \models \alpha$ implies $\mathcal{T}_1 \models \alpha$, for all Σ-inclusions α;*

(h) *$\mathcal{M}^{B}_{\mathcal{T}_2}$ is finitely Σ-homomorphically embeddable into $\mathcal{M}^{B}_{\mathcal{T}_1}$, for all $\mathcal{T}_1$-consistent Σ-concepts B.*

By applying condition **(p)** to $B \sqsubseteq \bot$, we obtain that every $\mathcal{T}_1$-consistent Σ-concept B is also $\mathcal{T}_2$-consistent.

Complexity of Σ-Query Entailment

We use Theorem 8 to show that deciding Σ-query entailment for *DL-Lite*$^{\mathcal{H}}_{core}$ TBoxes is PSPACE-hard and in EXPTIME.

Recall that subsumption in *DL-Lite*$^{\mathcal{H}}_{core}$ is NLOGSPACE-complete (Calvanese et al. 2007; Artale et al. 2009); so condition **(p)** of Theorem 8 can be checked in polynomial time.

And, since there are at most $2 \cdot |\Sigma|$ singleton Σ-ABoxes, we can concentrate on the complexity of checking finite Σ-homomorphic embeddability of canonical models for singleton ABoxes.

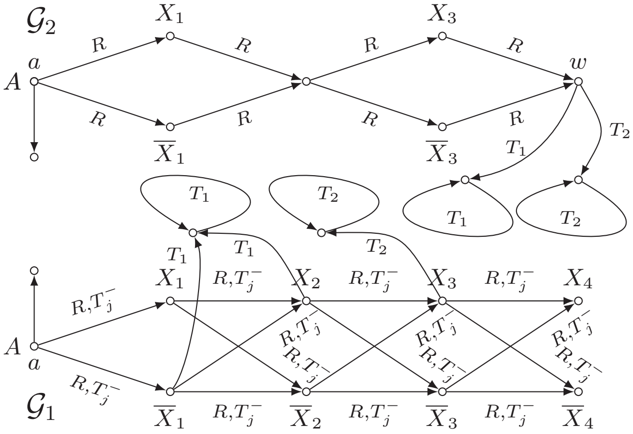
We begin by considering *DL-Lite$_{core}$*, which does not contain role inclusions. In this case, the existence of Σ-homomorphisms between canonical models can be expressed solely in terms of the types of the points in these models; cf. (Kontchakov, Wolter, and Zakharyaschev 2010). Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be *DL-Lite$_{core}$* TBoxes and Σ a signature.

Theorem 9. *$\mathcal{T}_1$ Σ-query entails $\mathcal{T}_2$ iff* **(p)** *holds and, for every $\mathcal{T}_1$-consistent Σ-concept B and every $x \in \Delta^{\mathcal{I}^B_{\mathcal{T}_2}}$, there is $x' \in \Delta^{\mathcal{I}^B_{\mathcal{T}_1}}$ with $\boldsymbol{t}_\Sigma^{\mathcal{I}^B_{\mathcal{T}_2}}(x) \subseteq \boldsymbol{t}_\Sigma^{\mathcal{I}^B_{\mathcal{T}_1}}(x')$.*

The criterion of Theorem 9 can be checked in polynomial time, in NLOGSPACE, to be more precise. Thus:

Theorem 10. *Checking Σ-query entailment for TBoxes in DL-Lite$_{core}$ is* NLOGSPACE*-complete.*

However, if role inclusions become available, the picture changes dramatically: not only do we have to compare the Σ-types of points in the canonical models, but also the Σ-*paths* to these points. To illustrate, consider the generating models $\mathcal{G}_1$, $\mathcal{G}_2$ below, where the arrows represent the generating relations, and the concept names A, X_i, $\overline{X}_i$ and the role names R and T_j are all symbols in Σ. The model $\mathcal{G}_2$ contains 4 R-paths from a to w, which are further



extended by the infinite T_j-paths. The paths π from a to w can be homomorphically mapped to distinct R-paths $h(\pi)$ in $\mathcal{G}_1$ starting from a. But the extension of such a π with the infinite T_j-chain can only be mapped to a *suffix of* $h(\pi)$ (backward, along T_j^-)—because we have to map paths in the unfolding $\mathcal{M}_2$ of $\mathcal{G}_2$ to paths in $\mathcal{M}_1$—and then to a T_j-loop in $\mathcal{G}_1$. But to check whether this can be done, we may have to 'remember' the whole path π.

To see that $\mathcal{G}_1$ and $\mathcal{G}_2$ can be given by *DL-Lite$^{\mathcal{H}}_{core}$* TBoxes, fix a quantified Boolean formula $\mathsf{Q}_1 X_1 \dots \mathsf{Q}_n X_n \bigwedge_{j=1}^m C_j$, where $\mathsf{Q}_i \in \{\forall, \exists\}$ and the C_j are clauses over the variables X_i. Let $\Sigma = \{A, X_i, \overline{X}_i, R, T_j \mid i \leq n,\ j \leq m\}$ and let $\mathcal{T}_1$ contain the inclusions

$$\begin{gathered} A \sqsubseteq \exists S_0^-, \quad \exists S_{i-1}^- \sqsubseteq \exists Q_i^k, \\ \exists (Q_i^k)^- \sqsubseteq X_i^k, \qquad Q_i^k \sqsubseteq S_i, \qquad S_i \sqsubseteq R, \\ X_i^k \sqsubseteq \exists R_j \ \text{ if } k = 0, \neg X_i \in C_j \text{ or } k = 1, X_i \in C_j, \\ \exists R_j^- \sqsubseteq \exists R_j, \qquad R_j \sqsubseteq T_j, \qquad S_i \sqsubseteq T_j^-, \end{gathered}$$

and $\mathcal{T}_2$ the inclusions

$$\begin{gathered} A \sqsubseteq \exists S_0^-, \qquad \exists S_{i-1}^- \sqsubseteq \begin{cases} \exists Q_i^k, & \text{if } \mathsf{Q}_i = \forall, \\ \exists S_i, & \text{if } \mathsf{Q}_i = \exists, \end{cases} \\ \exists (Q_i^k)^- \sqsubseteq X_i^k, \qquad Q_i^k \sqsubseteq S_i, \qquad S_i \sqsubseteq R, \\ \exists S_n^- \sqsubseteq \exists P_j, \qquad \exists P_j^- \sqsubseteq \exists P_j, \qquad P_j \sqsubseteq T_j, \end{gathered}$$

for all $i \leq n$, $j \leq m$ and $k = 1, 2$. The generating models $\mathcal{G}^A_{\mathcal{T}_1}$ and $\mathcal{G}^A_{\mathcal{T}_2}$, restricted to Σ, look like $\mathcal{G}_1$ and $\mathcal{G}_2$ in the picture above, respectively. Moreover, one can show that $\mathcal{M}^A_{\mathcal{T}_2}$ is (finitely) Σ-homomorphically embeddable into $\mathcal{M}^A_{\mathcal{T}_1}$ iff the QBF above is satisfiable. As satisfiability of QBFs is known to be PSPACE-complete, we obtain:

Theorem 11. *Σ-query entailment for DL-Lite$^{\mathcal{H}}_{core}$ TBoxes is* PSPACE*-hard.*

On the other hand, the problem whether $\mathcal{M}_{\mathcal{K}_2}$ is finitely Σ-homomorphically embeddable into $\mathcal{M}_{\mathcal{K}_1}$ can be reduced to the emptiness problem for alternating two-way automata, which belongs to EXPTIME (Vardi 1998). In a way similar to (Vardi 1998; Grädel and Walukiewicz 1999), where these automata were employed to prove EXPTIME-decidability of the modal μ-calculus with converse and the guarded fixed point logic of finite width, one can use their ability to 'remember' paths (in the sense illustrated in the example above) to obtain the EXPTIME upper bound:

Theorem 12. *Checking Σ-query entailment for DL-Lite$^{\mathcal{H}}_{core}$ TBoxes is in* EXPTIME.

The precise complexity of Σ-query entailment for *DL-Lite$^{\mathcal{H}}_{core}$* TBoxes is still unknown. To put the obtained results into perspective, let us recall that deciding Σ-query entailment for ontologies in the DL *DL-Lite$^{\mathcal{N}}_{horn}$* is CONP-complete (Kontchakov, Wolter, and Zakharyaschev 2010). Compared to *DL-Lite$^{\mathcal{H}}_{core}$*, *DL-Lite$^{\mathcal{N}}_{horn}$* allows (unqualified) number restrictions and conjunctions in the left-hand side of concept inclusions, but does not have role inclusions, that is: *DL-Lite$^{\mathcal{N}}_{horn}$* $\cap$ *DL-Lite$^{\mathcal{H}}_{core}$* $=$ *DL-Lite$_{core}$*. The data complexity of answering CQs is the same for all three languages under the UNA: AC^0. However, the computational properties of these logics become different as far as Σ-query entailment is concerned: NLOGSPACE-complete for *DL-Lite$_{core}$*, CONP-complete for *DL-Lite$^{\mathcal{N}}_{horn}$*, and between PSPACE and EXPTIME for *DL-Lite$^{\mathcal{H}}_{core}$*. It may be of interest to note that Σ-query entailment for *DL-Lite$^{\mathcal{N}}_{bool}$*, allowing full Booleans as concept constructs, is Π_2^p-complete.

Strong Σ-Query Entailment

It is pretty straightforward to construct an exponential time algorithm checking strong Σ-query entailment between *DL-Lite$^{\mathcal{H}}_{core}$* TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$: enumerate all Σ-TBoxes $\mathcal{T}$

and check whether $\mathcal{T}_1 \cup \mathcal{T}$ Σ-query entails $\mathcal{T}_2 \cup \mathcal{T}$. As there are quadratically many Σ-inclusions, this algorithm calls the Σ-query entailment checker $2^{|\Sigma|^2}$ times, in the worst case. We now show that one can do much better than that.

First, it turns out that instead of expensive Σ-query entailment checks for the TBoxes $\mathcal{T}_i \cup \mathcal{T}$, it is enough to check consistency (in polynomial time). More precisely, suppose $\mathcal{T}_1$ Σ-query entails $\mathcal{T}_2$. One can show then that $\mathcal{T}_1$ does not strongly Σ-query entail $\mathcal{T}_2$ iff there exist a Σ-TBox $\mathcal{T}$ and a Σ-concept B such that $(\mathcal{T}_1 \cup \mathcal{T}, \{B(a)\})$ is consistent but $(\mathcal{T}_2 \cup \mathcal{T}, \{B(a)\})$ is not (see Example 3 above).

Moreover, checking consistency for all Σ-TBoxes $\mathcal{T}$ can further be reduced—using the primitive form of *DL-Lite*$_{core}^{\mathcal{H}}$ axioms—to checking consistency for all *singleton* Σ-TBoxes $\mathcal{T}$. Thus, we obtain the following:

Theorem 13. *Suppose that $\mathcal{T}_1$ Σ-query entails $\mathcal{T}_2$. Then $\mathcal{T}_1$ does not strongly Σ-query entail $\mathcal{T}_2$ iff there is a Σ-concept B and a Σ-TBox $\mathcal{T}$ with a single inclusion of the form $B_1 \sqsubseteq B_2$ or $R_1 \sqsubseteq R_2$ such that $(\mathcal{T}_1 \cup \mathcal{T}, \{B(a)\})$ is consistent but $(\mathcal{T}_2 \cup \mathcal{T}, \{B(a)\})$ is inconsistent.*

So, if we already know that $\mathcal{T}_1$ Σ-query entails $\mathcal{T}_2$, then checking whether this entailment is actually *strong* can be done in polynomial time (and NLogSpace). The proof, based on both semantical and proof-theoretic constructions, is given in the full version of the paper www.dcs.bbk.ac.uk/˜roman/owl2ql-modules. Theorem 13 is crucial for the implementation of an efficient strong Σ-query entailment checker, as discussed in the section on our experiments below.

Incomplete Algorithm for Σ-Query Entailment

The complex interplay between role inclusions and inverse roles, required in the proof of PSpace-hardness, appears to be too artificial compared to how roles are used in 'real-world' ontologies. For example, in conceptual modelling, the number of roles is comparable with the number of concepts, but the number of role inclusions is normally very small (see the table in the next section). For this reason, instead of a complete exponential time Σ-query entailment checker, we have implemented a polynomial time correct but incomplete algorithm, which is based on testing simulations between transition systems.

Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be *DL-Lite*$_{core}^{\mathcal{H}}$ TBoxes, Σ a signature, B a Σ-concept. Denote $\mathcal{K}_i = (\mathcal{T}_i, \{B(a)\})$ and $\mathcal{I}_i = \mathcal{I}_{\mathcal{K}_i}$, $i = 1, 2$. A relation $\rho \subseteq \Delta^{\mathcal{I}_2} \times \Delta^{\mathcal{I}_1}$ is called a *Σ-simulation of $\mathcal{G}_{\mathcal{K}_2}$ in $\mathcal{G}_{\mathcal{K}_1}$* if the following conditions hold:

(s1) the domain of ρ is $\Delta^{\mathcal{I}_2}$ and $(a^{\mathcal{I}_2}, a^{\mathcal{I}_1}) \in \rho$;

(s2) $\boldsymbol{t}_\Sigma^{\mathcal{I}_2}(x) \subseteq \boldsymbol{t}_\Sigma^{\mathcal{I}_1}(x')$, for all $(x, x') \in \rho$;

(s3) if $x \rightsquigarrow_{\mathcal{K}_2} w_{[R]}$ and $(x, x') \in \rho$, then there is $y' \in \Delta^{\mathcal{I}_1}$ such that $(w_{[R]}, y') \in \rho$ and $S \in \boldsymbol{r}_\Sigma^{\mathcal{I}_1}(x', y')$ for every Σ-role S with $[R] \leq_{\mathcal{T}_2} [S]$.

We call ρ a *forward Σ-simulation* if it satisfies **(s1)**, **(s2)** and the condition **(s3′)**, which strengthens **(s3)** with the extra requirement: $y' = w_{[T]}$, for some role T, with $x' \rightsquigarrow_{\mathcal{K}_1} w_{[T]}$ and $[T] \leq_{\mathcal{T}_1} [S]$ for every Σ-role S with $[R] \leq_{\mathcal{T}_2} [S]$.

Example 14. In Example 6, there is a Σ-simulation of $\mathcal{G}_{\mathcal{K}_2}$ in $\mathcal{G}_{\mathcal{K}_1}$, but no forward Σ-simulation exists. The same applies to $\mathcal{G}_2$ and $\mathcal{G}_1$ in the proof of the PSpace lower bound.

In contrast to finite Σ-homomorphic embeddability of $\mathcal{M}_{\mathcal{K}_2}$ in $\mathcal{M}_{\mathcal{K}_1}$, the problem of checking the existence of (forward) Σ-simulations of $\mathcal{G}_{\mathcal{K}_2}$ in $\mathcal{G}_{\mathcal{K}_1}$ is tractable and well understood from the literature on program verification (Baier and Katoen 2007). Consider now the following conditions, which can be checked in polynomial time:

(y) condition **(p)** holds and there is a *forward* Σ-simulation of $\mathcal{G}_{\mathcal{T}_2}^{B}$ in $\mathcal{G}_{\mathcal{T}_1}^{B}$, for every $\mathcal{T}_1$-consistent Σ-concept B;

(n) condition **(p)** does not hold or there is no Σ-simulation of $\mathcal{G}_{\mathcal{T}_2}^{B}$ in $\mathcal{G}_{\mathcal{T}_1}^{B}$, for any $\mathcal{T}_1$-consistent Σ-concept B.

Theorem 15. *Let $\mathcal{T}_1, \mathcal{T}_2$ be DL-Lite$_{core}^{\mathcal{H}}$ TBoxes and Σ a signature. If* **(y)** *holds, then $\mathcal{T}_1$ Σ-query entails $\mathcal{T}_2$. If* **(n)** *holds, then $\mathcal{T}_1$ does not Σ-query entail $\mathcal{T}_2$.*

Thus, an algorithm checking conditions (**y**) and (**n**) can be used as a correct but incomplete Σ-query entailment checker. It cannot be complete since neither (**y**) nor (**n**) holds in Example 14. On the other hand, condition **(n)** proves to be a criterion of Σ-query entailment in two important cases:

Theorem 16. *Suppose that* (*a*) *$\mathcal{T}_1$ and $\mathcal{T}_2$ are DL-Lite$_{core}$ TBoxes, or* (*b*) *$\mathcal{T}_1 = \emptyset$ and $\mathcal{T}_2$ is a DL-Lite$_{core}^{\mathcal{H}}$ TBox. Then condition* **(n)** *holds iff $\mathcal{T}_1$ does not Σ-query entail $\mathcal{T}_2$.*

The case $\mathcal{T}_1 = \emptyset$ is of interest for module extraction and safe module import, which will be discussed in the next section.

Experiments

Checking (strong) Σ-query entailment has multiple applications in ontology versioning, re-use, and extraction. We have used the algorithms, suggested by Theorems 15 and 13, for *minimal module extraction* to see how efficient they are in practice and whether the incompleteness of the **(y)**–**(n)** conditions is problematic. Extracting minimal modules from medium-sized real-world ontologies requires thousands of calls of the (strong) Σ-query entailment checker, and thus provides a tough test for our approach.

For a TBox $\mathcal{T}$ and a signature Σ, a subset $\mathcal{M} \subseteq \mathcal{T}$ is

- a *Σ-query module* of $\mathcal{T}$ if $\mathcal{M} \equiv_\Sigma \mathcal{T}$;
- a *strong Σ-query module* of $\mathcal{T}$ if $\mathcal{M} \equiv_\Sigma^s \mathcal{T}$;
- a *depleting Σ-query module* of $\mathcal{T}$ if $\emptyset \equiv_{\Sigma \cup \mathsf{sig}(\mathcal{M})}^s \mathcal{T} \setminus \mathcal{M}$, where $\mathsf{sig}(\mathcal{M})$ is the signature of $\mathcal{M}$.

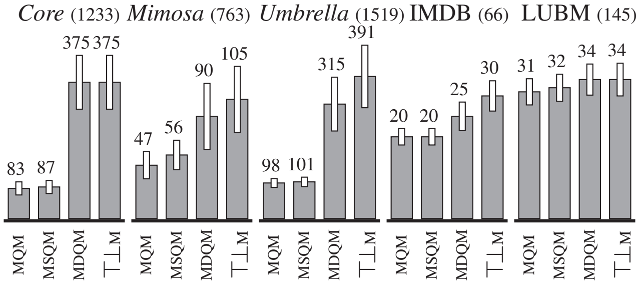
We are concerned with computing a *minimal* (w.r.t. $\subseteq$) Σ-query (MQM), a *minimal* strong Σ-query (MSQM), and the (uniquely determined) *minimal* depleting Σ-query (MDQM) module of $\mathcal{T}$. The general extraction algorithms, which call Σ-query entailment checkers, are taken from (Kontchakov, Wolter, and Zakharyaschev 2010). For MQMs and MSQMs, the number of calls to the checker coincides with the number of inclusions in $\mathcal{T}$. For MDQMs (where one of the TBoxes given to the checker is empty, and so the checker is complete, by Theorem 16), the number of checker calls is quadratic in the number of inclusions in $\mathcal{T}$.

We extracted modules from *OWL 2 QL* approximations of 3 commercial software applications called *Core*, *Umbrella* and *Mimosa* (the original ontologies use a few axioms that are not expressible *OWL 2 QL*). *Mimosa* is a specialisation of the MIMOSA OSA-EAI specification[1] for container

[1] htpp://www.mimosa.org/?q=resources/specs/osa-eai-v321

shipping. *Core* is based on a supply-chain management system used by the bookstore chain Ottakar's (now merged with Waterstone's), and *Umbrella* on a research data validation and processing system used by the Intensive Care National Audit and Research Centre[2] The original *Core* and *Umbrella* were used for the experiments in (Kontchakov, Wolter, and Zakharyaschev 2010). For comparison, we extracted modules from *OWL 2 QL* approximations of the well-known IMDB and LUBM ontologies. For each of these ontologies, we randomly generated 20 signatures Σ of 5 concept and 5 roles names. We extracted Σ-MQMs, MSQMs, MDQMs as well as the $\top\bot$-module (Cuenca Grau et al. 2008) from the whole *Mimosa*, IMDB and LUBM ontologies. For the larger *Umbrella* and *Core* ontologies, we first computed the $\top\bot$-modules, and then employed them to further extract MQMs, MSQMs, MDQMs, which are all contained in the $\top\bot$-modules. The average size of the resulting modules and its standard deviation is shown below:

ontology	*Mimosa*	*Core*	*Umbrella*	IMDB	LUBM
concept incl.	710	1214	1506	45	136
role incl.	53	19	13	21	9
concept nm.	106	82	79	14	43
role names	145	76	64	30	31



Details of the experiments and ontologies are available at www.dcs.bbk.ac.uk/˜roman/owl2ql-modules. Here we briefly comment on efficiency and incompleteness. Checking Σ-query inseparability turned out to be very fast: a single call of the checker never took more than 1s for our ontologies. For strong Σ-query inseparability, the maximal time was less than 1 min. For comparisons with the empty TBox, the maximal time for strong Σ-query inseparability tests was less than 10s. In the hardest case, *Mimosa*, the average total extraction times were 2.5mins for MQMs, 140mins for MSQMs, and 317mins for MDQMs. Finally, only in 9 out of about 75,000 calls, the Σ-query entailment checker was not able to give a certain answer due to incompleteness of the **(y)–(n)** condition, in which case the inclusions in question were added to the module.

Outlook

We have demonstrated that, despite its PSPACE-hardness, (strong) Σ-query inseparability can be decided efficiently for real-world *OWL 2 QL* ontologies. It would be of interest to explore (*i*) whether (some of) our techniques can be extended to more expressive DLs such as $\textit{DL-Lite}^{\mathcal{N}}_{horn}$ or even $\mathcal{ELI}$, and (*ii*) how the algorithms deciding inseparability can be utilised for analysing and visualising the difference between ontology versions if two ontologies are not Σ-query inseparable, as required by ontology versioning systems (Noy and Musen 2002).

[2] http://www.icnarc.org

Acknowledgments

This work was partially supported by the U.K. EPSRC grants EP/H05099X/1 and EP/H043594/1.

References

Artale, A.; Calvanese, D.; Kontchakov, R.; and Zakharyaschev, M. 2009. The *DL-Lite* family and relations. *Journal of Artificial Intelligence Research* 36:1–69.

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P., eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

Baier, C., and Katoen, J.-P. 2007. *Principles of Model Checking*. MIT Press.

Botoeva, E.; Calvanese, D.; and Rodriguez-Muro, M. 2010. Expressive approximations in *DL-Lite* ontologies. In *Proc. of AIMSA*, 21–31. Springer.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2006. Data complexity of query answering in description logics. In *Proc. of KR*, 260–270.

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

Cuenca Grau, B.; Horrocks, I.; Kazakov, Y.; and Sattler, U. 2008. Modular reuse of ontologies: Theory and practice. *JAIR* 31:273–318.

Dolby, J.; Fokoue, A.; Kalyanpur, A.; Ma, L.; Schonberg, E.; Srinivas, K.; and Sun, X. 2008. Scalable grounded conjunctive query evaluation over large and expressive knowledge bases. In *Proc. of ISWC*, v. 5318 of *LNCS*, 403–418.

Grädel, E., and Walukiewicz, I. 1999. Guarded fixed point logic. In *Proc. of LICS*, 45–54.

Heymans, S.; Ma, L.; Anicic, D.; Ma, Z.; Steinmetz, *et al.* 2008. Ontology reasoning with large data repositories. In *Ontology Management, Semantic Web, Semantic Web Services, and Business Applications*, Springer. 89–128.

Kontchakov, R.; Wolter, F.; and Zakharyaschev, M. 2010. Logic-based ontology comparison and module extraction, with an application to *DL-Lite*. *Artif. Intell.* 174:1093–1141.

Lutz, C., and Wolter, F. 2010. Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. *J. Symb. Comput.* 45(2):194–228.

Noy, N. F., and Musen, M. A. 2002. Promptdiff: A fixed-point algorithm for comparing ontology versions. In *Proc. of AAAI/IAAI*, 744–750.

Pan, J. Z., and Thomas, E. 2007. Approximating OWL-DL Ontologies. In *Proc. of AAAI*, 1434–1439.

Poggi, A.; Lembo, D.; Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Rosati, R. 2008. Linking data to ontologies. *J. on Data Semantics* X:133–173.

Stuckenschmidt, H.; Parent, C.; and Spaccapietra, S., eds. 2009. *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, v. 5445 of *LNCS*.

Vardi, M. Y. 1998. Reasoning about the past with two-way automata. In *Proc. of ICALP*, v. 1443 of *LNCS*, 628–641.