Can you Tell the Difference between *DL-Lite* Ontologies?

Roman Kontchakov
School of Computer Science and Information Systems
Birkbeck College, London, U.K.

Frank Wolter
Department of Computer Science
University of Liverpool, U.K.

Michael Zakharyaschev
School of Computer Science and Information Systems
Birkbeck College, London, U.K.

Abstract

We develop a formal framework for comparing different versions of *DL-Lite* ontologies. Four notions of difference and entailment between ontologies are introduced and their applications in ontology development and maintenance discussed. These notions are obtained by distinguishing between differences that can be observed among concept inclusions, answers to queries over ABoxes, and by taking into account additional context ontologies. We compare these notions, study their meta-properties, and determine the computational complexity of the corresponding reasoning tasks. Moreover, we show that checking difference and entailment can be automated by means of encoding into QBF satisfiability and using off-the-shelf QBF solvers. Finally, we explore the relationship between the notion of forgetting (or uniform interpolation) and our notions of difference between ontologies.

Introduction

In computer science, ontologies are used to provide a common vocabulary for a domain of interest, together with a description of the relationships between terms built from the vocabulary. Ontology languages based on description logics (DLs) represent ontologies as TBoxes (terminological boxes) containing inclusions between complex concepts over the vocabulary. An increasingly important application of ontologies is management of large amounts of data, where ontologies are used to provide flexible and efficient access to repositories consisting of data sets of instances of concepts and relations. In DLs, such repositories are typically modelled as ABoxes (sets of assertions).

Developing and maintaining ontologies for this and other purposes is a difficult task. When dealing with DLs, the ontology designer is supported by efficient reasoning tools for classification, instance checking and some other reasoning tasks. However, it is generally recognised that this support is not sufficient when ontologies are not developed as 'monolithic entities' but rather result from importing, merging, combining, re-using, refining and extending already existing ontologies. In all those cases, reasoning support for analysing the impact of the respective operation on the ontology would be highly desirable. Typical examples of such 'unorthodox' reasoning services include the following:

- *Comparing versions of ontologies.* The standard `diff` utility is an indispensable tool for comparing files. However, such a purely syntactic operation is of little value if the files contain different versions of ontologies (Noy & Musen 2002) because our concern now is not the syntactic form of their axioms, but the relationships between terms over their common vocabulary Σ these ontologies *imply*. The reasoning service we need in this case is to compare the logical consequences over Σ of different versions of ontologies.
- *Ontology refinement.* When refining an ontology by adding new axioms, one usually wants to preserve the relationships between terms of a certain part Σ of its vocabulary. The reasoning service required in such a case is to check whether the refined ontology has precisely the same logical consequences over Σ as the original one.
- *Ontology re-use.* When importing an ontology, one wants to *use* its vocabulary Σ as originally defined. However, relationships between terms over Σ may change due to some axioms in the importing ontology. So, again, we need a reasoning service capable of checking whether new logical consequences over Σ are derivable (this service has been termed *safety checking* in (Grau *et al.* 2007b)).

In all these and many other cases, we are interested in comparing the relationships between terms over some vocabulary (or *signature*) Σ two ontologies imply. This gives rise to the two main notions we investigate in this paper: Σ-*difference* and Σ-*entailment*. Roughly, the Σ-difference between two ontologies is the set of 'formulas' over Σ that are derivable from one ontology but not from the other; and one ontology Σ-entails another if all Σ-formulas derivable from the latter are also derivable from the former.

A very important special case of Σ-entailment, namely various versions of the notion of *conservative extension*, has been intensively investigated in the past few years (Antoniou & Kehagias 2000; Ghilardi, Lutz, & Wolter 2006; Grau *et al.* 2007a; 2007b; Lutz, Walther, & Wolter 2007). In this case one ontology is included in the other and Σ is the vocabulary of the smaller one. The Σ-formulas considered in these papers were concept inclusions $C_1 \sqsubseteq C_2$,

and a number of complexity and decidability results were obtained.[1] Also, model conservativity (Lutz, Walther, & Wolter 2007) and sufficient syntactic conditions of conservativity, e.g., locality (Grau *et al.* 2007b), have been considered.

In this paper we also deal with concept inclusions, but more importantly, we analyse Σ-difference and entailment with respect to existential Σ-*queries*, where the reasoning task is to decide whether two ontologies give precisely the same answers to Σ-queries for *any* database (= ABox) over Σ, and perhaps *any* additional context ontology over Σ. The corresponding notions of Σ-query difference and entailment are of interest for any DL, but they are of particular importance to those DLs that were specifically designed in order to facilitate efficient query-answering over large data sets.

The idea of using ontologies as a conceptual view over data repositories goes back to (Borgida *et al.* 1989) and has recently been developed to a quite practical level (Acciarri *et al.* 2005; Calvanese *et al.* 2007) with promising applications in such areas as data integration and P2P data management. The *DL-Lite* family of description logics has been largely designed with this application in mind (Calvanese *et al.* 2005; 2006). The data complexity of query answering is within LOGSPACE for most members of the family, and moreover, queries over *DL-Lite* ontologies can be rewritten as SQL queries so that standard database query engines can be used. *DL-Lite* is part of the OWL 1.1 Web Ontology Language, which is a W3C Member Submission.

In this paper, we investigate four notions of Σ-difference and Σ-entailment for two members of the *DL-Lite* family: *DL-Lite*$_{bool}$, the most expressive language of the family, basically covering all others, and *DL-Lite*$_{horn}$, the Horn subset of *DL-Lite*$_{bool}$. The four notions of Σ-difference and entailment are obtained by distinguishing between differences visible among concept inclusions, answers to queries over ABoxes, and by taking into account additional context ontologies. We compare these notions, study their meta-properties, and determine the computational complexity of the corresponding reasoning tasks. Moreover, we show that the reasoning services discussed above can be implemented by means of encoding into satisfiability of quantified Boolean formulas (QBF). We report on our first experiments with general purpose off-the-shelf QBF solvers for deciding Σ-entailment between 'typical' *DL-Lite* ontologies. Finally, we show that Σ-difference and entailment are closely related to forgetting and uniform interpolation as considered in (Lin & Reiter 1994; Pitts 1992). More precisely, we show that uniform interpolants always exist for *DL-Lite*$_{bool}$ and *DL-Lite*$_{horn}$ TBoxes and that these can be used to decide Σ-entailment, in some cases.

The *DL-Lite* Family

We remind the reader of the syntax and semantics of the DLs *DL-Lite*$_{bool}$ and *DL-Lite*$_{horn}$ introduced and investigated in (Calvanese *et al.* 2005; 2006; Artale *et al.* 2007). The language of *DL-Lite*$_{bool}$ has *object names* $a_1, a_2, \ldots$, *concept names* $A_1, A_2, \ldots$, and *role names* $P_1, P_2, \ldots$. Complex *roles* R and *DL-Lite*$_{bool}$ *concepts* C are defined as follows:

$$\begin{array}{lcl} R & ::= & P_i \mid P_i^-, \\ B & ::= & \bot \mid \top \mid A_i \mid \geq q\, R, \\ C & ::= & B \mid \neg C \mid C_1 \sqcap C_2, \end{array}$$

where $q \geq 1$. The concepts of the form B above are called *basic*. A *concept inclusion* in *DL-Lite*$_{bool}$ is of the form $C_1 \sqsubseteq C_2$, where C_1 and C_2 are *DL-Lite*$_{bool}$ concepts. (Other concept constructs like $\exists R$, $\leq q\, R$ and $C_1 \sqcup C_2$ will be used as standard abbreviations.) A *TBox in DL-Lite*$_{bool}$, $\mathcal{T}$, is a finite set of concept inclusions in *DL-Lite*$_{bool}$.

In the *Horn* fragment *DL-Lite*$_{horn}$ of *DL-Lite*$_{bool}$, *concept inclusions* are restricted to the form $\bigsqcap_k B_k \sqsubseteq B$, where B and the B_k are basic concepts. In this context, basic concepts will also be called *DL-Lite*$_{horn}$ *concepts*. Note that the inclusions $\bigsqcap_k B_k \sqsubseteq \bot$ and $\top \sqsubseteq B$ are legal in *DL-Lite*$_{horn}$. A *DL-Lite*$_{horn}$ *TBox* is a finite set of *DL-Lite*$_{horn}$ concept inclusions. It is worth noting that in *DL-Lite*$_{horn}$ we can express both *global functionality* of a role and *local functionality* (i.e., functionality restricted to a (basic) concept B) by means of the axioms $\geq 2\, R \sqsubseteq \bot$ and $B \sqcap \geq 2\, R \sqsubseteq \bot$.

Let $\mathcal{L}$ be either *DL-Lite*$_{bool}$ or *DL-Lite*$_{horn}$. An *ABox in* $\mathcal{L}$, $\mathcal{A}$, is a set of assertions of the form $C(a_i)$, $R(a_i, a_j)$, where C is an $\mathcal{L}$-concept, R a role, and a_i, a_j are object names. A *knowledge base in* $\mathcal{L}$ (*KB*, for short) is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ with a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ both in $\mathcal{L}$.

An *interpretation* $\mathcal{I}$ is a structure of the form $(\Delta^{\mathcal{I}}, A_1^{\mathcal{I}}, \ldots, P_1^{\mathcal{I}}, \ldots, a_1^{\mathcal{I}}, \ldots)$, where $\Delta^{\mathcal{I}}$ is a nonempty set, $A_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, $P_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and $a_i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ with $a_i^{\mathcal{I}} \neq a_j^{\mathcal{I}}$, for $a_i \neq a_j$ (i.e., we adopt the *unique name assumption*). The *extension* $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ of a concept C is defined as usual, e.g.,

$$d \in (\geq q\, R)^{\mathcal{I}} \quad \text{iff} \quad |\{d' \in \Delta^{\mathcal{I}} \mid (d, d') \in R^{\mathcal{I}}\}| \geq q.$$

A concept inclusion $C_1 \sqsubseteq C_2$ is *satisfied* in $\mathcal{I}$ if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$; in this case we write $\mathcal{I} \models C_1 \sqsubseteq C_2$. $\mathcal{I}$ is a *model for* a TBox $\mathcal{T}$ if all concept inclusions from $\mathcal{T}$ are satisfied in $\mathcal{I}$. An ABox assertion $C(a)$ ($R(a_i, a_j)$) is satisfied in $\mathcal{I}$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ ($(a_i^{\mathcal{I}}, a_j^{\mathcal{I}}) \in R^{\mathcal{I}}$). A concept inclusion $C_1 \sqsubseteq C_2$ *follows from* $\mathcal{T}$, $\mathcal{T} \models C_1 \sqsubseteq C_2$ in symbols, if every model for $\mathcal{T}$ satisfies $C_1 \sqsubseteq C_2$. A concept C is $\mathcal{T}$*-satisfiable* if there exists a model $\mathcal{I}$ for $\mathcal{T}$ with $C^{\mathcal{I}} \neq \emptyset$. We say that $\mathcal{I}$ is a *model* for a KB $(\mathcal{T}, \mathcal{A})$ if $\mathcal{I}$ is a model for $\mathcal{T}$ and every assertion of $\mathcal{A}$ is satisfied in $\mathcal{I}$.

An (*essentially positive*) *existential query in* $\mathcal{L}$ (or simply a *query*, if $\mathcal{L}$ is understood) is a first-order formula

$$q(x_1, \ldots, x_n) \;=\; \exists y_1 \ldots \exists y_m \varphi(x_1, \ldots, x_n, y_1, \ldots, y_m),$$

where φ is constructed, using only $\wedge$ and $\vee$, from atoms of the form $C(t)$ and $R(t_1, t_2)$, with C being an $\mathcal{L}$-concept, R a role, and t_i being either an object name or a variable from the list $x_1, \ldots, x_n, y_1, \ldots, y_m$. Given a KB $\mathcal{K}$ and a query $q(\vec{x})$, $\vec{x} = x_1, \ldots, x_n$, we say that an n-tuple $\vec{a}$ of object names is a *certain answer* to $q(\vec{x})$ w.r.t. $\mathcal{K}$ and write $\mathcal{K} \models q(\vec{a})$ if, for every model $\mathcal{I}$ for $\mathcal{K}$, we have $\mathcal{I} \models q(\vec{a})$.

[1] The complexity and decidability results for conservative extensions obtained in (Ghilardi, Lutz, & Wolter 2006; Lutz, Walther, & Wolter 2007) can also be generalised to Σ-entailment.

The subsumption problem '$\mathcal{T} \models C_1 \sqsubseteq C_2$?' is coNP-complete in *DL-Lite*$_{bool}$ and P-complete in *DL-Lite*$_{horn}$; the data complexity of the query answering problem for *DL-Lite*$_{horn}$ KBs is in LogSpace, while for *DL-Lite*$_{bool}$ it is coNP-complete (Artale *et al.* 2007).

What is the Difference?

As we saw in the introduction, the notions of difference and entailment between ontologies are restricted to some *signature*, i.e., a finite set of concept and role names.[2] Given a concept, role, concept inclusion, TBox, ABox, or query E, we denote by $sig(E)$ the *signature* of E, that is, the set of concept and role names that occur in E. It is to be noted that $\bot$ and $\top$ are regarded as logical symbols, and so $sig(\bot) = sig(\top) = \emptyset$. A concept (role, concept inclusion, TBox, ABox, query) E is called a Σ-*concept* (*role*, *concept inclusion*, *TBox*, *ABox*, *query*, respectively) if $sig(E) \subseteq \Sigma$. Thus, P^- is a Σ-role iff $P \in \Sigma$.

Definition 1 Let $\mathcal{L} \in \{$*DL-Lite*$_{bool}$, *DL-Lite*$_{horn}\}$ and let $\mathcal{T}_1$ and $\mathcal{T}_2$ be TBoxes in $\mathcal{L}$ and Σ a signature.

- The Σ-*concept difference* between $\mathcal{T}_1$ and $\mathcal{T}_2$ is the set $\mathsf{cDiff}^{\mathcal{L}}_{\Sigma}(\mathcal{T}_1, \mathcal{T}_2)$ of all Σ-concept inclusions $C \sqsubseteq D$ in $\mathcal{L}$ such that $\mathcal{T}_2 \models C \sqsubseteq D$ and $\mathcal{T}_1 \not\models C \sqsubseteq D$. We say that $\mathcal{T}_1$ Σ-*concept entails* $\mathcal{T}_2$ *in* $\mathcal{L}$ if $\mathsf{cDiff}^{\mathcal{L}}_{\Sigma}(\mathcal{T}_1, \mathcal{T}_2) = \emptyset$.
- The Σ-*query difference* between $\mathcal{T}_1$ and $\mathcal{T}_2$ is the set $\mathsf{qDiff}^{\mathcal{L}}_{\Sigma}(\mathcal{T}_1, \mathcal{T}_2)$ of pairs $(\mathcal{A}, q(\vec{x}))$, where $\mathcal{A}$ is a Σ-ABox in $\mathcal{L}$ and $q(\vec{x})$ a Σ-query in $\mathcal{L}$ such that $(\mathcal{T}_1, \mathcal{A}) \not\models q(\vec{a})$ and $(\mathcal{T}_2, \mathcal{A}) \models q(\vec{a})$, for some tuple $\vec{a}$ of object names from $\mathcal{A}$. We say that $\mathcal{T}_1$ Σ-*query entails* $\mathcal{T}_2$ *in* $\mathcal{L}$ if $\mathsf{qDiff}^{\mathcal{L}}_{\Sigma}(\mathcal{T}_1, \mathcal{T}_2) = \emptyset$.
- The *strong* Σ-*concept difference* between $\mathcal{T}_1$ and $\mathcal{T}_2$ is the set $\mathsf{scDiff}^{\mathcal{L}}_{\Sigma}(\mathcal{T}_1, \mathcal{T}_2)$ of all pairs $(\mathcal{T}, C \sqsubseteq D)$ such that $\mathcal{T}$ is a Σ-TBox in $\mathcal{L}$ and $C \sqsubseteq D \in \mathsf{cDiff}^{\mathcal{L}}_{\Sigma}(\mathcal{T} \cup \mathcal{T}_1, \mathcal{T} \cup \mathcal{T}_2)$. $\mathcal{T}_1$ *strongly* Σ-*concept entails* $\mathcal{T}_2$ *in* $\mathcal{L}$ if $\mathsf{scDiff}^{\mathcal{L}}_{\Sigma}(\mathcal{T}_1, \mathcal{T}_2) = \emptyset$.
- The *strong* Σ-*query difference* between $\mathcal{T}_1$ and $\mathcal{T}_2$ is the set $\mathsf{sqDiff}^{\mathcal{L}}_{\Sigma}(\mathcal{T}_1, \mathcal{T}_2)$ of all triples $(\mathcal{T}, \mathcal{A}, q(\vec{x}))$ such that $\mathcal{T}$ is a Σ-TBox in $\mathcal{L}$ and $(\mathcal{A}, q(\vec{x})) \in \mathsf{qDiff}^{\mathcal{L}}_{\Sigma}(\mathcal{T} \cup \mathcal{T}_1, \mathcal{T} \cup \mathcal{T}_2)$. We also say that $\mathcal{T}_1$ *strongly* Σ-*query entails* $\mathcal{T}_2$ *in* $\mathcal{L}$ if $\mathsf{sqDiff}^{\mathcal{L}}_{\Sigma}(\mathcal{T}_1, \mathcal{T}_2) = \emptyset$.

As argued in the introduction, the notions of Σ-difference and Σ–entailment can play an important role in comparing ontologies, checking whether a refinement of an ontology has undesirable effects on a certain part of its signature, and in checking whether a re-used (imported) ontology changes when put into the environment of another ontology. In all those cases, Σ indicates the vocabulary over which the user wants to compare ontologies. For example, for two versions of a medical ontology, a user interested in anatomy might choose Σ to be the set of terms relevant to anatomy and then check whether the two ontologies differ w.r.t. these terms.

In the definition of Σ-query difference, we take into account *arbitrary* Σ-ABoxes in $\mathcal{L}$. The reason is that during the ontology design phase, the data repositories to which the ontology will be applied are often either unknown or are subject to more or less frequent changes. Thus, to assume that we have a fixed ABox is unrealistic when checking differences between ontologies, and that is why in our approach we regard ABoxes as 'black boxes.'

Observe that, in general, more differences are detected when we consider Σ-queries rather than Σ-concept inclusions. Indeed, let $\mathcal{L}$ be one of *DL-Lite*$_{bool}$ and *DL-Lite*$_{horn}$. To see that any difference detected by means of concept inclusions can also be detected by means of queries, suppose that we have $\mathcal{T}_1 \not\models C_1 \sqsubseteq C_2$ and $\mathcal{T}_2 \models C_1 \sqsubseteq C_2$, for some Σ-concept inclusion $C_1 \sqsubseteq C_2$ in $\mathcal{L}$. Consider the ABox $\mathcal{A} = \{C_1(a)\}$ and the query $q = C_2(a)$. Then $(\mathcal{T}_2, \mathcal{A}) \models q$, while $(\mathcal{T}_1, \mathcal{A}) \not\models q$. (Note that in *DL-Lite*$_{horn}$, $C_1 = B_1 \sqcap \cdots \sqcap B_k$ and $C_2 = B$, where $B, B_1, \ldots, B_k$ are basic concepts.) To show that the converse does not hold, namely that queries can detect more differences than concept inclusions, we consider the following example. (Most of the claims in the examples below can be verified directly or using the criteria of Theorem 11 below.)

Example 2 Take $\Sigma = \{\mathsf{Lecturer}, \mathsf{Course}\}$, $\mathcal{T}_1 = \emptyset$, and

$$\mathcal{T}_2 = \{\mathsf{Lecturer} \sqsubseteq \exists \mathsf{teaches},\ \exists \mathsf{teaches}^- \sqsubseteq \mathsf{Course}\}.$$

Intuitively, the only consequence of $\mathcal{T}_2$ over Σ is 'if there is a lecturer, then there is a course,' but it cannot be expressed as a Σ-concept inclusion. Thus, $\mathcal{T}_1$ Σ-concept entails $\mathcal{T}_2$ (in both *DL-Lite*$_{bool}$ and *DL-Lite*$_{horn}$). However, $\mathcal{T}_1$ does not Σ-query entail $\mathcal{T}_2$. Indeed, let $\mathcal{A} = \{\mathsf{Lecturer}(a)\}$ and $q = \exists y\, \mathsf{Course}(y)$. Then $(\mathcal{T}_1, \mathcal{A}) \not\models q$ but $(\mathcal{T}_2, \mathcal{A}) \models q$.

It is also of interest to observe that Σ-query entailment in *DL-Lite*$_{horn}$ does not imply Σ-query entailment in *DL-Lite*$_{bool}$ (the converse implication follows immediately from the fact that *DL-Lite*$_{horn}$ is a fragment of *DL-Lite*$_{bool}$).

Example 3 Let $\Sigma = \{\mathsf{Lecturer}\}$, $\mathcal{T}_1 = \emptyset$, and

$$\mathcal{T}_2 = \{\mathsf{Lecturer} \sqsubseteq \exists \mathsf{teaches},\ \mathsf{Lecturer} \sqcap \exists \mathsf{teaches}^- \sqsubseteq \bot\}$$

Then $\mathcal{T}_1$ does not Σ-query entail $\mathcal{T}_2$ in *DL-Lite*$_{bool}$: just take $\mathcal{A}$ as before and $q = \exists y\, \neg \mathsf{Lecturer}(y)$. But $\mathcal{T}_1$ Σ-query entails $\mathcal{T}_2$ in *DL-Lite*$_{horn}$.

The first two notions of difference in Definition 1 do not take into account any *context ontologies* in which $\mathcal{T}_1$ or $\mathcal{T}_2$ may be used, nor do they cover the situation where $\mathcal{T}_1$ and $\mathcal{T}_2$ are changed by adding new axioms. To accommodate for this, we have introduced *strong* versions of entailment and difference. If two versions of ontologies strongly Σ-entail each other for their shared signature Σ, then they can be safely *replaced* by each other within any ontology $\mathcal{T}$ which only uses symbols from Σ; after such a replacement no differences between the sets of derivable Σ-concept inclusions (or answers to Σ-queries) can be detected. To see that the 'weak' notions of entailment do not always have this replacement property, consider the following example.

Example 4 Let $\mathcal{T}_1 = \emptyset$ and $\mathcal{T}_2$ be the TBox from Example 3 saying that every lecturer teaches and that a lecturer is not something which is taught. Let, as before, $\Sigma = \{\mathsf{Lecturer}\}$. Then $\mathcal{T}_1$ and $\mathcal{T}_2$ Σ-concept entail each other in *DL-Lite*$_{bool}$.

[2] As *DL-Lite*$_{bool}$ TBoxes do not contain object names, we do not have to include them to signatures (unlike DLs with nominals).

But for $\mathcal{T} = \{\top \sqsubseteq \mathsf{Lecturer}\}$, we have $\mathcal{T}_1 \cup \mathcal{T} \not\models \top \sqsubseteq \bot$ and $\mathcal{T}_2 \cup \mathcal{T} \models \top \sqsubseteq \bot$, i.e., the former TBox is consistent while the latter is not. Thus, the difference between $\mathcal{T}_1$ and $\mathcal{T}_2$ becomes visible if we extend them with the ontology $\mathcal{T}$.

Example 4 shows also that Σ-query entailment in *DL-Lite*$_{horn}$ does not imply strong Σ-concept entailment in *DL-Lite*$_{horn}$. In the context of defining modules within ontologies, taking into account changes to ontologies and context ontologies has been strongly advocated in (Grau *et al.* 2007b), which inspired our definitions. The following example shows that strong Σ-concept entailment in *DL-Lite*$_{horn}$ does not imply strong Σ-concept entailment in *DL-Lite*$_{bool}$.

Example 5 Consider the *DL-Lite*$_{horn}$ TBoxes

$$\mathcal{T}_1 = \big\{\mathsf{Male} \sqcap \mathsf{Female} \sqsubseteq \bot,\ \top \sqsubseteq \exists \mathsf{father},\ \top \sqsubseteq \exists \mathsf{mother},\ \exists \mathsf{father}^- \sqsubseteq \mathsf{Male},\ \exists \mathsf{mother}^- \sqsubseteq \mathsf{Female}\big\},$$
$$\mathcal{T}_2 = \big\{\top \sqsubseteq \exists \mathsf{id},\ \mathsf{Male} \sqcap \exists \mathsf{id}^- \sqsubseteq \bot,\ \mathsf{Female} \sqcap \exists \mathsf{id}^- \sqsubseteq \bot\big\},$$

and let $\Sigma = \{\mathsf{Male}, \mathsf{Female}, \mathsf{father}, \mathsf{mother}\}$. $\mathcal{T}_2$ implies that $\top \not\sqsubseteq \mathsf{Male} \sqcup \mathsf{Female}$. Now, in *DL-Lite*$_{bool}$, $\mathcal{T}_1 \cup \mathcal{T}_2$ is not strongly Σ-concept entailed by $\mathcal{T}_1$: it is enough to take $\mathcal{T} = \{\top \sqsubseteq \mathsf{Male} \sqcup \mathsf{Female}\}$. However, $\mathcal{T}_1 \cup \mathcal{T}_2$ is strongly Σ-entailed by $\mathcal{T}_1$ in *DL-Lite*$_{horn}$.

Semantic Criteria of Σ-Entailment

Now we compare the notions of Σ-difference and Σ-entailment in a systematic way using model-theoretic characterisations. Our first observation generalises the well-known result from propositional logic according to which two propositional Horn theories entail the same Horn formulas if, and only if, these theories have the same consequences in the class of all propositional formulas.

Theorem 6 *For any DL-Lite*$_{horn}$ *TBoxes* $\mathcal{T}_1$, $\mathcal{T}_2$ *and any signature* Σ, *the following two conditions are equivalent*:

- $\mathcal{T}_1$ Σ*-concept entails* $\mathcal{T}_2$ *in DL-Lite*$_{bool}$;
- $\mathcal{T}_1$ Σ*-concept entails* $\mathcal{T}_2$ *in DL-Lite*$_{horn}$.

Examples 3 and 5 show that this theorem does not hold for the stronger notions of Σ-entailment. Moreover, for neither *DL-Lite*$_{horn}$ nor *DL-Lite*$_{bool}$ any of the stronger notions is equivalent to Σ-concept entailment. Our second theorem summarises the classification of the remaining notions and shows that in all those cases where we have not provided counterexamples our notions of Σ-entailment are equivalent.

Theorem 7 *Let* $\mathcal{L}$ *be DL-Lite*$_{bool}$ *or DL-Lite*$_{horn}$, $\mathcal{T}_1$ *and* $\mathcal{T}_2$ *TBoxes in* $\mathcal{L}$, *and* Σ *a signature. For* $\mathcal{L}$ = *DL-Lite*$_{bool}$, *the following conditions are equivalent*:

(1) $\mathcal{T}_1$ Σ*-query entails* $\mathcal{T}_2$ *in* $\mathcal{L}$;

(2) $\mathcal{T}_1$ *strongly* Σ*-concept entails* $\mathcal{T}_2$ *in* $\mathcal{L}$;

(3) $\mathcal{T}_1$ *strongly* Σ*-query entails* $\mathcal{T}_2$ *in* $\mathcal{L}$.

For $\mathcal{L}$ = *DL-Lite*$_{horn}$, *conditions* (2) *and* (3) *are equivalent, while* (1) *is strictly weaker than each of them.*

Thus, the full comparison table looks as follows:

DL-Lite$_{horn}$
Σ-concept $\lneqq$ Σ-query $\lneqq$ strong Σ-concept$\equiv$strong Σ-query
DL-Lite$_{bool}$
Σ-concept $\lneqq$ Σ-query $\equiv$ strong Σ-concept$\equiv$strong Σ-query

The equivalence results of Theorem 7 follow from the model-theoretic characterisations of the notions of Σ-entailment to be presented below. In this paper, our characterisations will have a somewhat syntactic flavour in the sense that they are formulated in terms of *types*—syntactic abstractions of domain elements—realised in models, rather than in model-theoretic terms. The advantage of such characterisations is that they can be used directly for designing decision algorithms, despite the fact that the underlying models are often infinite as neither *DL-Lite*$_{bool}$ nor *DL-Lite*$_{horn}$ has the finite model property (Calvanese *et al.* 2005). Needless to say, however, that the correctness of the type-based characterisations presented below require model constructions (see the Appendix on proof sketches).

Let Σ be a signature and Q a set of positive natural numbers containing 1. By a ΣQ*-concept* we mean any concept of the form $\bot$, $\top$, A_i, $\geq q\, R$, or its negation, for some $A_i \in \Sigma$, Σ-role R and $q \in Q$. A ΣQ*-type* is a set $\boldsymbol{t}$ of ΣQ-concepts containing $\top$ such that the following conditions hold:

- for every ΣQ-concept C, either $C \in \boldsymbol{t}$ or $\neg C \in \boldsymbol{t}$,
- if $q < q'$ are both in Q and $\geq q'\, R \in \boldsymbol{t}$ then $\geq q\, R \in \boldsymbol{t}$.

Clearly, for each ΣQ-type $\boldsymbol{t}$ with $\bot \notin \boldsymbol{t}$, there is an interpretation $\mathcal{I}$ and a point x in it such that $x \in C^{\mathcal{I}}$, for all $C \in \boldsymbol{t}$. In this case we say that $\boldsymbol{t}$ is *realised* (at x) in $\mathcal{I}$.

Definition 8 For a TBox $\mathcal{T}$, a ΣQ-type $\boldsymbol{t}$ is called $\mathcal{T}$*-realisable* if $\boldsymbol{t}$ is realised in a model for $\mathcal{T}$. A set Ξ of ΣQ-types is said to be $\mathcal{T}$*-realisable* if there is a model for $\mathcal{T}$ realising *all* types from Ξ. We also say that Ξ is *precisely* $\mathcal{T}$*-realisable* if there is a model $\mathcal{I}$ for $\mathcal{T}$ such that $\mathcal{I}$ realises all types in Ξ, and every ΣQ-type realised in $\mathcal{I}$ is in Ξ.

Given a TBox $\mathcal{T}$, let $Q_{\mathcal{T}}$ denote the set of numerical parameters occurring in $\mathcal{T}$ together with 1. The following conditions are equivalent:

- $\mathcal{T}_1$ Σ-concept entails $\mathcal{T}_2$ in *DL-Lite*$_{bool}$;
- every $\mathcal{T}_1$-realisable $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-type is $\mathcal{T}_2$-realisable.

This equivalence is trivial if one considers $\Sigma\mathbb{N}$-types instead of $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types. Thus, the message here is that it is sufficient to consider only parameters from $Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$.

For Σ-query entailment in *DL-Lite*$_{bool}$ (and the two other equivalent notions), the following conditions are equivalent:

- $\mathcal{T}_1$ Σ-query entails $\mathcal{T}_2$ in *DL-Lite*$_{bool}$;
- every precisely $\mathcal{T}_1$-realisable set Ξ of $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types is precisely $\mathcal{T}_2$-realisable.

Intuitively, while Σ-concept entailment is a 'local' form of entailment referring to one point in a model, Σ-query entailment and strong Σ-concept/query entailment are 'global' in the sense that all points of models have to be considered.

Example 9 In Example 2, to compute the $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types, we do not require numerical parameters, as Σ contains

no role names. There are four $\mathcal{T}_1$-realisable Σ-types $\{(\neg)\mathsf{Lecturer}, (\neg)\mathsf{Course}\}$. All of these are $\mathcal{T}_2$-realisable as well. However, the singleton set $\{\{\mathsf{Lecturer}, \neg\mathsf{Course}\}\}$ is precisely $\mathcal{T}_1$-realisable but not precisely $\mathcal{T}_2$-realisable.

In the case of *DL-Lite*$_{horn}$ more definitions are required. Given a ΣQ-type $\boldsymbol{t}$, let $\boldsymbol{t}^+ = \{B \in \boldsymbol{t} \mid B \text{ a basic concept}\}$ (i.e., positive part of the type). Say that a ΣQ-type $\boldsymbol{t}_1$ is *h-contained* in a ΣQ-type $\boldsymbol{t}_2$ if $\boldsymbol{t}_1^+ \subseteq \boldsymbol{t}_2^+$. The following two notions characterise Σ-entailment for *DL-Lite*$_{horn}$:

Definition 10 A set Ξ of ΣQ-types is said to be *sub-precisely $\mathcal{T}$-realisable* if there is a model $\mathcal{I}$ for $\mathcal{T}$ such that $\mathcal{I}$ realises all types from Ξ, and every ΣQ-type realised in $\mathcal{I}$ is h-contained in a type from Ξ. We also say that Ξ is *meet-precisely $\mathcal{T}$-realisable* if there is a model $\mathcal{I}$ for $\mathcal{T}$ such that, for every ΣQ-type $\boldsymbol{t}$ realised in $\mathcal{I}$, $\Xi_{\boldsymbol{t}} \neq \emptyset$ and $\boldsymbol{t}^+ = \bigcap_{\boldsymbol{t}_i \in \Xi_{\boldsymbol{t}}} \boldsymbol{t}_i^+$, where $\Xi_{\boldsymbol{t}} = \{\boldsymbol{t}_i \in \Xi \mid \boldsymbol{t}^+ \subseteq \boldsymbol{t}_i^+\}$. (It follows that $\boldsymbol{t}^+ \subseteq \boldsymbol{t}_i^+$, for all $\boldsymbol{t}_i \in \Xi_{\boldsymbol{t}}$, and thus, Ξ is sub-precisely $\mathcal{T}$-realisable.)

Theorem 11 *Let $\mathcal{L} \in \{$DL-Lite$_{bool}$, DL-Lite$_{horn}\}$ and 'Σ-entails' be one of the four notions of Σ-entailment. For a signature Σ and TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ in $\mathcal{L}$, the following are equivalent*:

- *$\mathcal{T}_1$ Σ-entails $\mathcal{T}_2$ in $\mathcal{L}$;*
- *every precisely $\mathcal{T}_1$-realisable set of $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$ types satisfies the corresponding property from the following table*:

Σ-*entailment*	*language* $\mathcal{L}$	
	DL-Lite$_{horn}$	DL-Lite$_{bool}$
Σ-*concept*[3]	$\mathcal{T}_2$-*realisable*	$\mathcal{T}_2$-*realisable*
Σ-*query*	*sub-precisely* $\mathcal{T}_2$-*realisable*	*precisely* $\mathcal{T}_2$-*realisable*
strong Σ-*concept*	*meet-precisely* $\mathcal{T}_2$-*realisable*	
strong Σ-*query*		

Example 12 Consider the TBoxes from Example 3. Again, we do not require numerical parameters because Σ does not contain role names. The $\mathcal{T}_1$-realisable Σ-types are $\{\neg\mathsf{Lecturer}\}$ and $\{\mathsf{Lecturer}\}$, and both are $\mathcal{T}_2$-realisable. Hence $\mathcal{T}_1$ Σ-concept entails $\mathcal{T}_2$ in *DL-Lite*$_{bool}$ (and, therefore, in *DL-Lite*$_{horn}$). The singleton set $\{\{\mathsf{Lecturer}\}\}$ is precisely $\mathcal{T}_1$-realisable, but not precisely $\mathcal{T}_2$-realisable. Hence $\mathcal{T}_1$ does not Σ-query entail $\mathcal{T}_2$ in *DL-Lite*$_{bool}$. However, $\{\{\mathsf{Lecturer}\}\}$ is sub-precisely $\mathcal{T}_1$-realisable and, therefore $\mathcal{T}_1$ Σ-query entails $\mathcal{T}_2$ in *DL-Lite*$_{horn}$. On the other hand, $\{\{\mathsf{Lecturer}\}\}$ is not meet-precisely $\mathcal{T}_2$-realisable, and so $\mathcal{T}_1$ does not strongly Σ-concept entail $\mathcal{T}_2$ in *DL-Lite*$_{horn}$.

Robustness Properties

Results regarding Σ-difference and Σ-entailment can be easily misinterpreted and are of limited use if these notions do not enjoy certain robustness properties. To start with, recall that in the definition of essentially positive existential queries for *DL-Lite*$_{bool}$, we allow *negated* concepts in queries and ABoxes. An alternative approach would be to allow only *positive* concepts. These two types of queries give rise to different notions of query entailment: under the second definition, the TBox $\mathcal{T}_2$ from Example 3 is Σ-query entailed by $\mathcal{T}_1 = \emptyset$, even in *DL-Lite*$_{bool}$. We argue, however, that it is the *essentially positive* queries that should be considered in the context of this investigation. The reason is that, with only positive queries allowed, the addition of the *definition* $B \equiv \neg\mathsf{Lecturer}$ to $\mathcal{T}_2$ and B to Σ would result in a TBox which is not Σ-query entailed by $\mathcal{T}_1$ in *DL-Lite*$_{bool}$ any longer. This kind of non-robust behaviour of the notion of Σ-entailment is clearly undesirable. Obviously, the formulations we gave are robust under the addition of definitions to TBoxes. We now consider two other robustness conditions.

Theorem 13 *Let $\mathcal{L} \in \{$DL-Lite$_{horn}$, DL-Lite$_{bool}\}$ and 'Σ-entails' be one of the four notions of Σ-entailment given in Definition 1.*

- *The relation 'Σ-entails' is robust under vocabulary extensions for $\mathcal{L}$: for all $\mathcal{L}$-TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$, if $\mathcal{T}_1$ Σ-entails $\mathcal{T}_2$, then $\mathcal{T}_1$ Σ'-entails $\mathcal{T}_2$, for every Σ' such that $\Sigma' \cap sig(\mathcal{T}_2) \subseteq \Sigma$.*
- *The relation 'Σ-entails' is robust under joins for $\mathcal{L}$: for all $\mathcal{L}$-TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$, if $\mathcal{T}$ and $\mathcal{T}_i$ Σ-entail each other, for $i = 1, 2$, and $sig(\mathcal{T}_1) \cap sig(\mathcal{T}_2) \subseteq \Sigma$, then $\mathcal{T}$ Σ-entails $\mathcal{T}_1 \cup \mathcal{T}_2$.*

Robustness under vocabulary extensions is of particular importance for query Σ-entailment and the strong versions of Σ-entailment. For example, it implies that if $\mathcal{T}_1$ strongly Σ-query entails $\mathcal{T}_2$ then, for any ABox $\mathcal{A}$, TBox $\mathcal{T}$ and query q containing, besides Σ, *arbitrary* symbols not occurring in $\mathcal{T}_2$, we have $(\mathcal{T}_1 \cup \mathcal{T}, \mathcal{A}) \models q(\vec{a})$ whenever $(\mathcal{T}_2 \cup \mathcal{T}, A) \models q(\vec{a})$. This property is critical for applications, as it is hardly possible to restrict ABoxes and context ontologies to a fixed signature Σ and not permit the use of any fresh symbols.

Robustness under joins is of interest for collaborative ontology development. This property means that if two (or more) ontology developers extend a given ontology $\mathcal{T}$ independently and do not use common symbols with the exception of those in a certain signature Σ then they can safely form the union of $\mathcal{T}$ and all their additional axioms provided that their individual extensions are safe for Σ.

Both robustness conditions are closely related to the well-known *Robinson consistency lemma* and *interpolation* (see e.g., (Chang & Keisler 1990)), which have been investigated in the context of modular software specification (Diaconescu, Goguen, & Stefaneas 1993) as well. They typically fail for description logics with nominals and/or role hierarchies (Areces & ten Cate 2006; Konev *et al.* 2007). Observe that, for robustness under joins, *mutual* Σ-entailment of $\mathcal{T}$ and $\mathcal{T}_i$ is required:

Example 14 Let $\mathcal{T}_1 = \{A \sqsubseteq \exists R, \exists R^- \sqsubseteq B\}$, $\mathcal{T}_2 = \mathcal{T}$, $\mathcal{T} = \{\top \sqsubseteq \neg B\}$, and $\Sigma = \{A, B\}$. Then $\mathcal{T}$ Σ-concept entails $\mathcal{T}_i$, $i = 1, 2$, but $\mathcal{T}_1 \cup \mathcal{T}_2 \models \top \sqsubseteq \neg A$, and so it is not Σ-concept entailed by $\mathcal{T}$.

Complexity and Algorithms

We first determine the complexity of deciding Σ-entailment and then consider the problem of computing Σ-differences.

[3] Every $\mathcal{T}_1$-realisable type is always contained in a precisely $\mathcal{T}_1$-realisable set.

Theorem 15 *For all notions of Σ-entailment introduced in Definition 1, deciding Σ-entailment is Π_2^p-complete in DL-Lite$_{bool}$ and* coNP*-complete in DL-Lite$_{horn}$.*

The lower bounds follow immediately from the corresponding lower bounds for propositional logic and its Horn fragment. The upper bound for Σ-concept entailment in *DL-Lite$_{bool}$* and *DL-Lite$_{horn}$* is rather straightforward: by the characterisation of Theorem 11, it is sufficient to check that every $\mathcal{T}_1$-realisable $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-type is $\mathcal{T}_2$-realisable. Thus, to check non Σ-concept entailment in *DL-Lite$_{bool}$*, the algorithm guesses a $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-type and checks, using an NP-oracle, that it is $\mathcal{T}_1$-realisable but not $\mathcal{T}_2$-realisable. For *DL-Lite$_{horn}$*, this latter check can be done in deterministic polynomial time.

Proving the upper bounds for the remaining decision problems is harder: the criteria of Theorem 11 do not make any claim regarding the cardinality of the sets of $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types one has to consider (there are exponentially many $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types). Our upper bound proof shows that it suffices to consider sets Ξ of $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types the size of which is bounded by a *linear* function in the size of the TBoxes. Then, for *DL-Lite$_{bool}$* TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$, one can decide whether $\mathcal{T}_1$ does *not* Σ-entail $\mathcal{T}_2$ by guessing a set Ξ of linearly many $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types and checking that it is precisely $\mathcal{T}_1$-realisable and *not* precisely $\mathcal{T}_2$-realisable. The Appendix provides an NP algorithm deciding whether a given set of ΣQ-types is precisely $\mathcal{T}$-realisable. This gives the Π_2^p upper bound for Σ-query entailment in *DL-Lite$_{bool}$*. The procedures for *DL-Lite$_{horn}$* are similar: given *DL-Lite$_{horn}$* TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$, one can decide whether $\mathcal{T}_1$ does not (strongly) Σ-query entail $\mathcal{T}_2$ by guessing Ξ and checking that it is precisely $\mathcal{T}_1$-realisable and *not* sub-precisely (respectively, meet-precisely) $\mathcal{T}_2$-realisable. The Appendix provides polynomial *deterministic* algorithms deciding whether a set of ΣQ-types is precisely, sub-precisely and meet-precisely $\mathcal{T}$-realisable, for a *DL-Lite$_{horn}$* TBox $\mathcal{T}$. This gives coNP upper bounds for Σ-query and strong Σ-query entailment.

Observe that deciding Σ-entailment and conservativity is much harder for most DLs: it is ExpTime-complete for $\mathcal{EL}$ (Lutz & Wolter 2007), 2ExpTime-complete for $\mathcal{ALC}$ and $\mathcal{ALCQI}$, and undecidable for $\mathcal{ALCQIO}$ (Ghilardi, Lutz, & Wolter 2006; Lutz, Walther, & Wolter 2007).

In applications, it is not enough just to decide whether two ontologies differ w.r.t. a signature. If the ontologies are different, the ontology engineer needs an informative list of differences. Observe that the set of Σ-differences as defined in Definition 1 is either infinite or empty. Thus, only approximations of these sets can be computed. By the criteria of Theorem 11, for Σ-concept difference the ΣQ-types which are $\mathcal{T}_1$-realisable but not $\mathcal{T}_2$-realisable are obvious candidates to include in such a set. Such a type contains, for each concept name $A \in \Sigma$ and $(\geq qR)$ with $q \in Q$, $R \in \Sigma$, either the concept itself, or its negation. If there are too many Σ-differences (remember, there are exponentially many types) and the resulting list is incomprehensible, the user can step-by-step decrease the size of Σ (e.g., by removing elements X from Σ such that two types which coincide except for X are in the Σ-difference) until the set of types in the Σ-difference can be analysed. Moreover, as a second step the user might consider applying pinpointing algorithms (Schlobach & Cornet 2003) which exhibit the axioms in the ontology from which the Σ-differences are derivable. For stronger versions of Σ-difference, it appears to be unavoidable to consider precisely $\mathcal{T}_1$-realisable sets of ΣQ-types, which are (in one of the three ways described in Theorem 11) not precisely $\mathcal{T}_2$-realisable. We leave a systematic study of the problem of constructing or approximating the query and strong versions of difference for future research.

Experimental Results

To see whether the algorithms of the previous section can be used in practice, we refined the criteria for Σ-concept and Σ-query entailment in *DL-Lite$_{bool}$* and encoded them by means of $\forall\exists$ QBFs (the validity problem for which is Π_2^p-complete). The reader can find the encodings in the full version available at `www.dcs.bbk.ac.uk/~roman/qbf2`. We used these QBF translations to check Σ-concept/query entailment for *DL-Lite$_{bool}$* ontologies with the help of three *off-the-shelf* QBF solvers: sKizzo (Benedetti 2005), 2clsQ (Samulowitz & Bacchus 2006) and Quaffle (Zhang & Malik 2002b; 2002a). As our benchmarks, we considered three series of instances of the form $(\mathcal{T}_1, \mathcal{T}_2, \Sigma)$. In the *NN-series*, $\mathcal{T}_1$ does not Σ-concept entail $\mathcal{T}_2$; in the *YN-series*, $\mathcal{T}_1$ Σ-concept but not Σ-query entails $\mathcal{T}_2$; and in the *YY-series*, $\mathcal{T}_1$ Σ-query entails $\mathcal{T}_2$. The sizes of the instances are uniformly distributed over the intervals given in the table below.

series	no. of instances	axioms		basic concepts		
		$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_1$	$\mathcal{T}_2$	Σ
NN	420	59–154	74–198	47–121	49–146	5–52
YN	252	56–151	77–191	44–119	58–145	6–45
YY	156	54–88	62–110	43–79	47–94	6–32

It is to be noted that our ontologies were *not* randomly generated. On the contrary, we used 'typical' *DL-Lite* ontologies available on the Web: extensions of *DL-Lite$_{bool}$* fragments of the standard 'department ontology' as well as *DL-Lite$_{bool}$* representations of the ER diagrams used in the QuOnto system (`www.dis.uniroma1.it/~quonto/`).

The next table illustrates the size of the QBF translations of our instances for both Σ-concept and Σ-query entailment.

series	Σ-concept entailment QBF		Σ-query entailment QBF	
	variables	clauses	variables	clauses
NN	1,469–11,752	2,391–18,277	1,715–15,174	5,763–163,936
YN	1,460–11,318	2,352–17,424	1,755–14,723	7,006–151,452
YY	1,526–4,146	2,200–6,079	1,510–4,946	5,121–29,120

The large difference between the size of the QBF translations for Σ-concept and Σ-query entailment (say, 18,277 v. 163,936 clauses in the same instance) reflects the difference between simple and precise realisability of sets of types (cf. Theorem 11): roughly, the QBF encoding of the latter requires *quadratical* number of clauses (in the number of role names) whereas the former needs only linearly many.

A brief summary of the tests, conducted on a 3GHz P4 machine with 2GB RAM, is given in Fig. 1, where the graphs in the upper (lower) row show the percentage of solved instances for Σ-concept (respectively, Σ-query) entailment; for details and more charts see

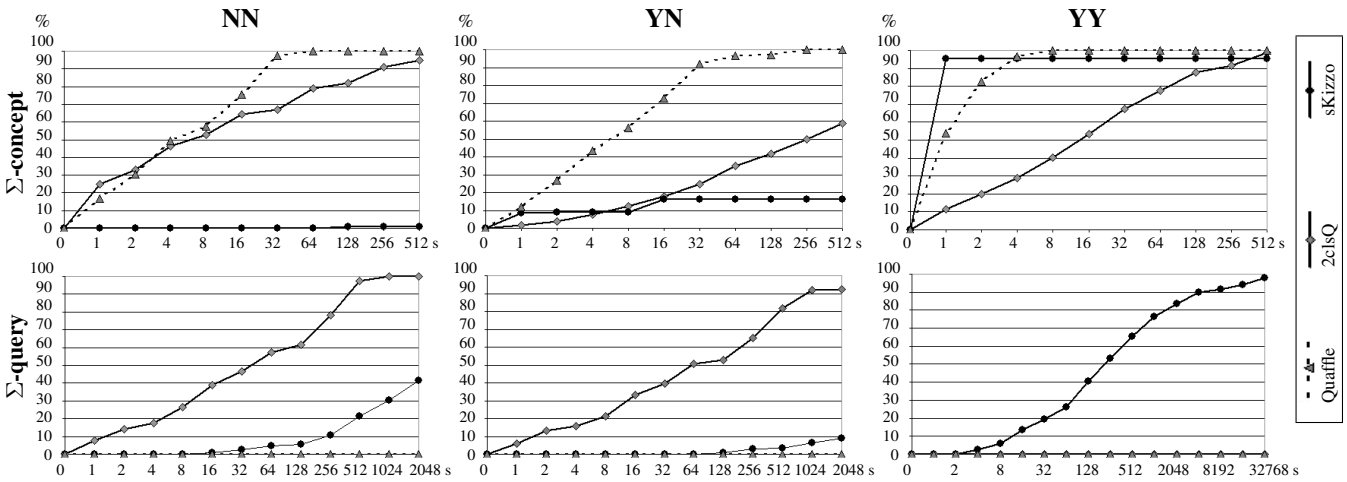

Figure 1: Percentage of instances solved (Y axis) for each timeout (X axis).

www.dcs.bbk.ac.uk/~roman/qbf2.

The main conclusion of the tests is that automated checking of Σ-entailment between *DL-Lite*$_{bool}$ ontologies[4] is indeed possible, even with *off-the-shelf general purpose software* (let alone dedicated reasoners). Although of the same worst-case complexity, in practice Σ-concept entailment turns out to be much easier to check than Σ-query entailment. Quaffle solved all of our 828 Σ-concept instances. However, none of the solvers could cope with all Σ-query instances, with those of the YY series being especially hard. All in all, we have solved more than 95% of the Σ-query instances. Another interesting observation is that, for Σ-concept entailment, bigger signatures usually meant harder instances, whereas the impact of the size of Σ on Σ-query entailment was rather limited. Finally, our tests showed that none of the three solvers was better than the others when checking Σ-entailment: Quaffle was the best for Σ-concept entailment, 2clsQ for Σ-query entailment with the answer 'NO,' and sKizzo for Σ-query entailment with the answer 'YES.' On the other hand, having tried various quantifier orderings in the prenex QBF translations (www.dcs.bbk.ac.uk/~roman/qbf2), we have identified a number of strategies that could dramatically improve performance of QBF solvers when checking Σ-query entailment.

Forgetting

Now we briefly discuss a different approach to deciding Σ-entailment, which is based on the notion of *forgetting* (or *uniform interpolation*). Assume that we want to re-use the information provided by a TBox $\mathcal{T}$ about a certain signature Σ. If we are not interested at all in what $\mathcal{T}$ says about the symbols that do not belong to Σ, then it would be useful to have a new TBox $\mathcal{T}_\Sigma$ *containing only symbols from* Σ and having the same consequences over Σ as $\mathcal{T}$ in the sense that $\mathcal{T}$ and $\mathcal{T}_\Sigma$ Σ-entail each other. Of course, the question whether such an ontology $\mathcal{T}_\Sigma$ exists and can be constructed depends on the language constructors, the signature Σ, and the type of Σ-entailment we are interested in. A similar problem (in different contexts) has been studied by different research communities which used different names such as *forgetting* (Lang, Liberatore, & Marquis 2003; Lin & Reiter 1994; Wang *et al.* 2008), *uniform interpolation* (Pitts 1992; Visser 1996; Ghilardi, Lutz, & Wolter 2006; Konev *et al.* 2007), and *variable elimination*. For the case of Σ-concept entailment in *DL-Lite*, the notion of forgetting or uniform interpolation can be defined as follows.

Definition 16 Let $\mathcal{L} \in \{\textit{DL-Lite}_{horn}, \textit{DL-Lite}_{bool}\}$. We say that $\mathcal{L}$ *admits forgetting* (or *has uniform interpolation*) if, for every TBox $\mathcal{T}$ in $\mathcal{L}$ and every finite signature Σ, there exists a TBox $\mathcal{T}_\Sigma$ in $\mathcal{L}$ with $\mathsf{sig}(\mathcal{T}_\Sigma) \subseteq \Sigma$ such that $\mathcal{T}$ and $\mathcal{T}_\Sigma$ Σ-concept entail each other in $\mathcal{L}$. In this case, $\mathcal{T}_\Sigma$ is called a *uniform interpolant* of $\mathcal{T}$ w.r.t. Σ in $\mathcal{L}$.

Example 17 Consider the TBox

$$\mathcal{T} = \{\mathsf{Hand} \sqsubseteq \mathsf{BodyPart},\ \mathsf{BodyPart} \sqsubseteq \mathsf{PhysicalObject}\}$$

in $\mathcal{L}$, and let $\Sigma = \{\mathsf{Hand}, \mathsf{PhysicalObject}\}$. Then the TBox $\mathcal{T}_\Sigma = \{\mathsf{Hand} \sqsubseteq \mathsf{PhysicalObject}\}$ is clearly a uniform interpolant of $\mathcal{T}$ w.r.t. Σ in $\mathcal{L}$.

Observe that if $\mathcal{L}$ has uniform interpolation, then computing uniform interpolants is an alternative way of deciding Σ-concept entailment in $\mathcal{L}$. Indeed, suppose that two TBoxes $\mathcal{T}$ and $\mathcal{T}'$ in $\mathcal{L}$ are given and that we want to decide whether $\mathcal{T}$ Σ-concept entails $\mathcal{T}'$ in $\mathcal{L}$. Compute a uniform interpolant $\mathcal{T}'_\Sigma$ of $\mathcal{T}'$ w.r.t. Σ in $\mathcal{L}$. Then $\mathcal{T}$ Σ-concept entails $\mathcal{T}'$ if $\mathcal{T} \models C_1 \sqsubseteq C_2$ for all $(C_1 \sqsubseteq C_2) \in \mathcal{T}'_\Sigma$. Thus, Σ-concept entailment can be reduced to computing uniform interpolants and checking subsumption in $\mathcal{L}$. The following theorem states that *DL-Lite*$_{bool}$ and *DL-Lite*$_{horn}$ do enjoy uniform interpolation.

Theorem 18 *Let* $\mathcal{L} \in \{\textit{DL-Lite}_{bool}, \textit{DL-Lite}_{horn}\}$. *Then* $\mathcal{L}$ *has uniform interpolation, and a uniform interpolant of a*

[4]The main application area of the *DL-Lite* family of logics is conceptual data modelling and data integration, where typical *DL-Lite* ontologies do not contain more than a few hundred axioms.

TBox $\mathcal{T}$ w.r.t. Σ in $\mathcal{L}$ can be constructed in exponential time in the size of $\mathcal{T}$ (that is, the number of occurrences of symbols in $\mathcal{T}$).

It would be interesting to conduct experiments on deciding Σ-concept entailment using Theorem 18 and compare the performance of this approach with the one based on the QBF encoding we discussed above.

We remark that forgetting of concepts (but not roles) has been studied in (Wang *et al.* 2008). However, the method presented in (Wang *et al.* 2008) cannot be generalised to forgetting roles, which should be allowed according to the definition of forgetting. It is also worth mentioning that, for many standard DLs such as $\mathcal{ALC}$ and even $\mathcal{EL}$, uniform interpolants do not always exist (Ghilardi, Lutz, & Wolter 2006; Konev *et al.* 2007).

The notion of uniform interpolation introduced above is of little help if we want to decide stronger versions of Σ-entailment from Definition 1.

Example 19 Let $\mathcal{L} \in \{\textit{DL-Lite}_{bool}, \textit{DL-Lite}_{horn}\}$. Consider the TBox

$$\mathcal{T} = \{\mathsf{Lecturer} \sqsubseteq \exists \mathsf{teaches},\ \exists \mathsf{teaches}^- \sqsubseteq \mathsf{Course}\}$$

from Example 2, and let $\Sigma = \{\mathsf{Lecturer}, \mathsf{Course}\}$. Then $\mathcal{T}_\Sigma = \emptyset$ is a uniform interpolant of $\mathcal{T}$ w.r.t. Σ in $\mathcal{L}$ because, as we have already seen, the empty TBox Σ-concept entails $\mathcal{T}$ in $\mathcal{L}$. We also know that the empty TBox does not Σ-query entail $\mathcal{T}$ in $\mathcal{L}$. Thus, in contrast to Σ-concept entailment, computing a uniform interpolant of $\mathcal{T}$ w.r.t. Σ in $\mathcal{L}$ does not really help us decide Σ-query entailment in $\mathcal{L}$.

Thus, we are facing the problem of finding a modification of the notion of uniform interpolation that is capable of deciding Σ-query entailment (and other stronger notions of Σ-entailment from Definition 1). Here we briefly sketch a solution for *DL-Lite*$_{bool}$.

Denote by *DL-Lite*$^u_{bool}$ the extension of *DL-Lite*$_{bool}$ by the *universal role* u, with the *DL-Lite*$^u_{bool}$ *concepts* defined as for *DL-Lite*$_{bool}$ except that now we have the additional clause

$$C \quad ::= \quad \ldots \ \mid \ \exists u.C' \ \mid \ \ldots,$$

where C' is a *DL-Lite*$_{bool}$ concept. (Note that *DL-Lite*$^u_{bool}$ concepts contain no nested occurrences of $\exists u$.) Given an interpretation $\mathcal{I}$, we set $(\exists u.C)^\mathcal{I} = \Delta^\mathcal{I}$ if $C^\mathcal{I} \neq \emptyset$ and $(\exists u.C)^\mathcal{I} = \emptyset$ otherwise. The remaining model-theoretic notions are defined exactly as for *DL-Lite*$_{bool}$. Using the construction from (Artale *et al.* 2007) one can show that the subsumption problem '$\mathcal{T} \models C_1 \sqsubseteq C_2$?' is still coNP-complete for TBoxes $\mathcal{T}$ in *DL-Lite*$^u_{bool}$ and concept inclusions $C_1 \sqsubseteq C_2$ in *DL-Lite*$^u_{bool}$. It is important that we regard u as a *logical symbol*, so that $\mathsf{sig}(\exists u.C) = \mathsf{sig}(C)$.

Definition 20 Let $\mathcal{T}$ be a TBox in *DL-Lite*$_{bool}$ and Σ a signature. A TBox $\mathcal{T}_\Sigma$ in *DL-Lite*$^u_{bool}$ is called a *uniform interpolant* of $\mathcal{T}$ w.r.t. Σ in *DL-Lite*$^u_{bool}$ if $\mathsf{sig}(\mathcal{T}_\Sigma) \subseteq \Sigma$ and, for every concept inclusion $C_1 \sqsubseteq C_2$ in *DL-Lite*$^u_{bool}$ with $\mathsf{sig}(C_1 \sqsubseteq C_2) \subseteq \Sigma$, we have

$$\mathcal{T} \models C_1 \sqsubseteq C_2 \qquad \text{iff} \qquad \mathcal{T}_\Sigma \models C_1 \sqsubseteq C_2.$$

The language *DL-Lite*$^u_{bool}$ captures exactly the remaining notions of Σ-entailment for *DL-Lite*$_{bool}$, which, by Theorem 7, are all equivalent to Σ-query entailment:

Theorem 21 *Let $\mathcal{T}$ and $\mathcal{T}'$ be TBoxes in DL-Lite$_{bool}$ and Σ a signature. And let $\mathcal{T}'_\Sigma$ be a uniform interpolant of $\mathcal{T}'$ w.r.t. Σ in DL-Lite$^u_{bool}$. Then $\mathcal{T}$ Σ-query entails $\mathcal{T}'$ if, and only if, $\mathcal{T} \models C_1 \sqsubseteq C_2$, for every $(C_1 \sqsubseteq C_2) \in \mathcal{T}'_\Sigma$.*

The following theorem shows that *DL-Lite*$_{bool}$ has uniform interpolation w.r.t. *DL-Lite*$^u_{bool}$.

Theorem 22 *For every TBox $\mathcal{T}$ in DL-Lite$_{bool}$ and every signature Σ, one can constructed in exponential time in the size of $\mathcal{T}$ a uniform interpolant $\mathcal{T}_\Sigma$ of $\mathcal{T}$ w.r.t. Σ in DL-Lite$^u_{bool}$.*

Example 23 Consider again the TBox

$$\mathcal{T} = \{\mathsf{Lecturer} \sqsubseteq \exists \mathsf{teaches},\ \exists \mathsf{teaches}^- \sqsubseteq \mathsf{Course}\}$$

and $\Sigma = \{\mathsf{Lecturer}, \mathsf{Course}\}$. Then

$$\mathcal{T}_\Sigma = \{\mathsf{Lecturer} \sqsubseteq \exists u.\mathsf{Course}\}$$

is a uniform interpolant of $\mathcal{T}$ w.r.t. Σ in *DL-Lite*$^u_{bool}$.

Conclusion

We have analysed the relation between various notions of difference and entailment w.r.t. a signature in description logics *DL-Lite*$_{bool}$ and *DL-Lite*$_{horn}$, and proved that the corresponding reasoning problems are not harder (at least theoretically) than similar problems in propositional logic. We also demonstrated that an efficient reasoning service for checking Σ-entailment between *DL-Lite*$_{bool}$ ontologies can be implemented, even using general purpose off-the-shelf QBF solvers.

Future research problems include the following:

- The algorithms presented for Σ-entailment provide a basis for developing *module extraction algorithms* for *DL-Lite* ontologies. Such an algorithm should output, given an ontology and a signature Σ, a minimal sub-ontology which Σ-entails the full ontology; see (Grau *et al.* 2007a) for an overview. It remains to develop the details of such a procedure for *DL-Lite*.
- We have only provided a sketch of how an approximation of the differences between two versions of an ontology can be computed. Further experimental results are required to evaluate the feasibility of this approach.
- It would be of interest to implement an algorithm computing uniform interpolants and conduct experiments to evaluate how large uniform interpolants can be for real-world ontologies and signatures. Note that, in the worst case, minimal uniform interpolants are of exponential size in the size of the original TBox. Moreover, the computed uniform interpolants can then be used to decide Σ-entailment for *DL-Lite*$_{bool}$ and it would be of interest to compare this approach to deciding Σ-entailment with the QBF encoding method developed in this paper.

- We have shown that the extension *DL-Lite*$^{\mathcal{U}}_{bool}$ of *DL-Lite*$_{bool}$ with the universal role is sufficiently expressive to capture Σ-query entailment using uniform interpolants. It would be interesting to design corresponding languages for Σ-query entailment and strong Σ-concept entailment for *DL-Lite*$_{horn}$.

Acknowledgements

The work on this paper was partially supported by the U.K. EPSRC research grants EP/E034942/1 and EP/E065279/1.

Appendix: Proof Sketches

In the appendix, we give proof sketches of some of the results for *DL-Lite*$_{bool}$. Full proofs of all results of this paper, including those for the case of *DL-Lite*$_{horn}$, are available at `www.dcs.bbk.ac.uk/~roman`.

First, we prove an important amalgamation property of *DL-Lite*$_{bool}$ models. Given a signature Σ, we say that two interpretations $\mathcal{I}$ and $\mathcal{J}$ are Σ-*isomorphic* and write $\mathcal{I} \sim_\Sigma \mathcal{J}$ if there is a bijection $f\colon \Delta^{\mathcal{I}} \to \Delta^{\mathcal{J}}$ such that (i) $f(a^{\mathcal{I}}) = a^{\mathcal{J}}$, for every object name a, (ii) $x \in A^{\mathcal{I}}$ iff $f(x) \in A^{\mathcal{J}}$, for every concept name A in Σ, and (iii) $(x, y) \in P^{\mathcal{I}}$ iff $(f(x), f(y)) \in P^{\mathcal{J}}$, for every role name P in Σ. Clearly, Σ-isomorphic interpretations cannot be distinguished by Σ-TBoxes, Σ-ABoxes or Σ-queries. Given a set $\mathcal{I}_i$, $i \in I$, of interpretations with $0 \in I$, define the interpretation

$$\mathcal{J} \;=\; \bigoplus_{i \in I} \mathcal{I}_i,$$

where $\Delta^{\mathcal{J}} = \{(i, w) \mid i \in I, w \in \Delta_i\}$, $a^{\mathcal{J}} = (0, a^{\mathcal{I}_1})$, for an object name a, $A^{\mathcal{J}} = \{(i, w) \mid w \in A^{\mathcal{I}_i}\}$, for a concept name A, and $P^{\mathcal{J}} = \{((i, w_1), (i, w_2)) \mid (w_1, w_2) \in P^{\mathcal{I}_i}\}$, for a role name P. $\mathcal{J}$ can be regarded as a *disjoint union* of the $\mathcal{I}_i$. Given an interpretation $\mathcal{I}$, we set

$$\mathcal{I}^\omega \;=\; \bigoplus_{i \in \omega} \mathcal{I}_i,$$

where $\mathcal{I}_i = \mathcal{I}$ for $i \in \omega$. It should be clear that Σ-TBoxes, Σ-ABoxes or Σ-queries (for *any* signature Σ) cannot distinguish between $\mathcal{I}$ and $\mathcal{I}^\omega$.

Lemma A.1 *Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be at most countable models for TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively, and let Σ be a signature such that sig$(\mathcal{T}_1) \cap$ sig$(\mathcal{T}_2) \subseteq \Sigma$. If interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ realise precisely the same $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types, then there is an interpretation $\mathcal{I}^*$ such that:*

- $\mathcal{I}^* \models \mathcal{T}_1 \cup \mathcal{T}_2$,
- $\mathcal{I}^* \sim_\Sigma \mathcal{I}_1^\omega$, *and*
- $\mathcal{I}^*$, $\mathcal{I}_1$ *and* $\mathcal{I}_2$ *realise the same set of* $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$*-types.*

Proof. Let Ξ be the set of $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types realised in $\mathcal{I}_1$ (and $\mathcal{I}_2$). We show that $\mathcal{I}_1^\omega$ can be expanded to a model $\mathcal{I}^*$ for $\mathcal{T}_1 \cup \mathcal{T}_2$. As both $\mathcal{I}_1^\omega$ and $\mathcal{I}_2^\omega$ realise each $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-type from Ξ by countably infinitely many points, there is a bijection $f\colon \Delta^{\mathcal{I}_2^\omega} \to \Delta^{\mathcal{I}_1^\omega}$ which is *invariant under* $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types. Now, we set $\Delta^{\mathcal{I}^*} = \Delta^{\mathcal{I}_1^\omega}$ and, for all object names a, concept names A, and role names P,

$$a^{\mathcal{I}^*} = a^{\mathcal{I}_1^\omega},$$

$$A^{\mathcal{I}^*} = \begin{cases} A^{\mathcal{I}_1^\omega}, & \text{if } A \in \Sigma \cup \mathit{sig}(\mathcal{T}_1), \\ \{f(x) \mid x \in A^{\mathcal{I}_2^\omega}\}, & \text{otherwise,} \end{cases}$$

$$P^{\mathcal{I}^*} = \begin{cases} P^{\mathcal{I}_1^\omega}, & \text{if } P \in \Sigma \cup \mathit{sig}(\mathcal{T}_1), \\ \{(f(x), f(y)) \mid (x, y) \in P^{\mathcal{I}_2^\omega}\}, & \text{otherwise.} \end{cases}$$

One can show that $\mathcal{I}^*$ is as required. ❑

We need the following immediate consequence of Lemma A.1.

Lemma A.2 *Let $\mathcal{J}$ be an (at most countable) model for $\mathcal{T}_1$ and Σ a signature with $\Sigma \subseteq$ sig$(\mathcal{T}_1)$. Suppose that there is a model for $\mathcal{T}_2$ realising exactly the same $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types as $\mathcal{J}$. Then there is a model $\mathcal{I}^*$ for $\mathcal{T}_2$ such that $\mathcal{I}^* \sim_\Sigma \mathcal{J}^\omega$.*

In particular, $\mathcal{I}^* \models \mathcal{A}$ iff $\mathcal{J} \models \mathcal{A}$, for all Σ-ABoxes $\mathcal{A}$, $\mathcal{I}^* \models \mathcal{T}$ iff $\mathcal{J} \models \mathcal{T}$, for all Σ-TBoxes $\mathcal{T}$, and $\mathcal{I}^* \models q(\vec{a})$ iff $\mathcal{J} \models q(\vec{a})$, for all Σ-queries $q(\vec{a})$.

Now, the *DL-Lite*$_{bool}$ part of Theorems 7 and 11 is a consequence of the following lemma:

Lemma A.3 *Let $\mathcal{T}_1$, $\mathcal{T}_2$ be TBoxes in DL-Lite$_{bool}$ and Σ a signature.*
(i) *The following two conditions are equivalent*:

(ce$_\mathbf{b}$) *$\mathcal{T}_1$ Σ-concept entails $\mathcal{T}_2$ in DL-Lite$_{bool}$;*
(r) *every $\mathcal{T}_1$-realisable $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-type is $\mathcal{T}_2$-realisable.*

(ii) *The following four conditions are equivalent*:

(sce$_\mathbf{b}$) *$\mathcal{T}_1$ strongly Σ-concept entails $\mathcal{T}_2$;*
(qe$_\mathbf{b}$) *$\mathcal{T}_1$ Σ-query entails $\mathcal{T}_2$;*
(sqe$_\mathbf{b}$) *$\mathcal{T}_1$ strongly Σ-query entails $\mathcal{T}_2$;*
(pr) *if a set of $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types is precisely $\mathcal{T}_1$-realisable, then it is precisely $\mathcal{T}_2$-realisable.*

Proof. **(ce$_\mathbf{b}$) ⇒ (r)** Suppose $\boldsymbol{t}$ is a $\mathcal{T}_1$-realisable $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-type which is not $\mathcal{T}_2$-realisable. Then $\mathcal{T}_2 \models D \sqsubseteq \bot$ but $\mathcal{T}_1 \not\models D \sqsubseteq \bot$, for $D = \bigsqcap_{C \in \boldsymbol{t}} C$, contrary to $\mathcal{T}_2$ being Σ-entailed by $\mathcal{T}_1$.

(r) ⇒ (ce$_\mathbf{b}$) Assume $\mathcal{T}_1 \not\models C_1 \sqsubseteq C_2$ and $\mathit{sig}(C_1 \sqsubseteq C_2) \subseteq \Sigma$. Take a model $\mathcal{J}$ for $\mathcal{T}_1$ with $\mathcal{J} \not\models C_1 \sqsubseteq C_2$. By **(r)**, we find a model $\mathcal{I}_2$ for $\mathcal{T}_2$ realising all $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types which are realised in $\mathcal{J}$. Let $\mathcal{I}_1$ be the disjoint union of $\mathcal{J}$ and $\mathcal{I}_2$. Then $\mathcal{I}_1 \not\models C_1 \sqsubseteq C_2$ and $\mathcal{I}_1$ and $\mathcal{I}_2$ realise precisely the same $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types. By Lemma A.1 (applied to $\mathcal{T}_2$ and the empty TBox), we find a model $\mathcal{I}$ for $\mathcal{T}_2$ with $\mathcal{I} \not\models C_1 \sqsubseteq C_2$. Hence $\mathcal{T}_2 \not\models C_1 \sqsubseteq C_2$.

(pr) ⇒ (sqe$_\mathbf{b}$) Suppose that there are a Σ-TBox $\mathcal{T}$, a Σ-ABox $\mathcal{A}$ and a Σ-query $q(\vec{x})$ such that $(\mathcal{T}_2 \cup \mathcal{T}, \mathcal{A}) \models q(\vec{a})$ but $(\mathcal{T}_1 \cup \mathcal{T}, \mathcal{A}) \not\models q(\vec{a})$, for some object names $\vec{a}$. Take a model $\mathcal{J}$ for $(\mathcal{T}_1 \cup \mathcal{T}, \mathcal{A})$ such that $\mathcal{J} \not\models q(\vec{a})$ and let Ξ be the set of $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types realised in $\mathcal{J}$. Then there is a model for $\mathcal{T}_2$ realising exactly the types in Ξ and, by Lemma A.2, there exists a model $\mathcal{I}^*$ for $\mathcal{T}_2$ such that $\mathcal{I}^* \models (\mathcal{T}, \mathcal{A})$ and $\mathcal{I}^* \not\models q(\vec{a})$, which is a contradiction.

(qe$_\mathbf{b}$) ⇒ (pr) Suppose there is a set Ξ of $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types that is precisely $\mathcal{T}_1$-realisable but not precisely $\mathcal{T}_2$-realisable. Consider the ABox $\mathcal{A}_\Xi = \{C(a_{\boldsymbol{t}}) \mid C \in \boldsymbol{t}, \boldsymbol{t} \in \Xi\}$, where

$a_{\boldsymbol{t}}$ is a fresh object name, for each type $\boldsymbol{t} \in \Xi$. Let Θ be the set of all $\mathcal{T}_2$-realisable $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types that are not in Ξ. We have $\Theta \neq \emptyset$. Now consider the query

$$q \;=\; \exists x \bigvee_{\boldsymbol{t} \in \Theta} \bigwedge_{C \in \boldsymbol{t}} C(x).$$

Then $(\mathcal{T}_2, \mathcal{A}_\Xi) \models q$ but $(\mathcal{T}_1, \mathcal{A}_\Xi) \not\models q$, which is again a contradiction.

(sce$_\mathbf{b}$) $\Rightarrow$ (pr) Let Ξ be a set of precisely $\mathcal{T}_1$-realisable $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types. Consider

$$\mathcal{T}_\Xi = \Big\{ \top \sqsubseteq \bigsqcup_{\boldsymbol{t} \in \Xi} \prod_{C \in \boldsymbol{t}} C \Big\}.$$

Clearly, $\mathcal{T}_1 \cup \mathcal{T}_\Xi \not\models \prod_{C \in \boldsymbol{t}} C \sqsubseteq \bot$, for every $\boldsymbol{t} \in \Xi$. Then, by **(sce$_\mathbf{b}$)**, $\mathcal{T}_2 \cup \mathcal{T}_\Xi \not\models \prod_{C \in \boldsymbol{t}} C \sqsubseteq \bot$ and thus, there is a model $\mathcal{I}_{\boldsymbol{t}}$ for $\mathcal{T}_2 \cup \mathcal{T}_\Xi$ realising $\boldsymbol{t}$. Take the disjoint union $\mathcal{I}$ of all these models $\mathcal{I}_{\boldsymbol{t}}$, for $\boldsymbol{t} \in \Xi$. It is easy to see that $\mathcal{I}$ is a model for $\mathcal{T}_2$ realising precisely the types in Ξ. ❑

We now prove the complexity results for deciding Σ-query entailment for *DL-Lite$_{bool}$* as stated in Theorem 15. The proof also indicates the encoding into QBF used in our experiments. We reduce precise $\mathcal{T}$-realisability of a set of types (as stated in criterion **(pr)** of Lemma A.3) to a satisfiability problem in propositional logic. Let Σ be a signature and Q a set of positive natural numbers containing 1. With every basic concept B of the form A or $\geq q\, R$ we associate a fresh *propositional variable* B^*, and, for a concept C in *DL-Lite$_{bool}$*, denote by C^* the result of replacing each B in it with B^* (and $\sqcap$, $\sqcup$ with $\wedge$, $\vee$, respectively), for a ΣQ-type $\boldsymbol{t}$, denote by $\boldsymbol{t}^*$ the set $\{C^* \mid C \in \boldsymbol{t}\}$, and, for a TBox $\mathcal{T}$, denote by $\mathcal{T}^*$ the set $\{C_1^* \to C_2^* \mid C_1 \sqsubseteq C_2 \in \mathcal{T}\}$. Thus, C^* is a formula and $\boldsymbol{t}^*$, $\mathcal{T}^*$ are sets of formulas of *propositional logic*.

The following result follows immediately from (Artale *et al.* 2007):

Lemma A.4 *Let $\mathcal{T}$ be a TBox in DL-Lite$_{bool}$, $Q \supseteq Q_\mathcal{T}$, and Ω be a set of roles closed under inverse and containing all sig($\mathcal{T}$)-roles. Then a set Ξ of ΣQ-types is precisely $\mathcal{T}$-realisable iff there is a set $\Omega_0 \subseteq \Omega$ closed under inverse and such that*:

(t) *for each $\boldsymbol{t} \in \Xi$, $\boldsymbol{t}^* \cup \mathsf{Ax}(\mathcal{T}, \Omega_0)$ is satisfiable*;

(pw) *for each $R \in \Omega_0$, there is a type $\boldsymbol{t}_R$ in Ξ such that $\boldsymbol{t}_R^* \cup \{(\geq 1\, R)^*\} \cup \mathsf{Ax}(\mathcal{T}, \Omega_0)$ is satisfiable,*

where

$$\mathsf{Ax}(\mathcal{T}, \Omega_0) \;=\; \mathcal{T}^* \;\cup\; \big\{\neg(\geq 1\, R)^* \mid R \in \Omega \setminus \Omega_0\big\} \;\cup\; \big\{(\geq q\, R)^* \to (\geq q'\, R)^* \mid R \in \Omega,\, q, q' \in Q,\, q > q'\big\}.$$

It follows, in particular, that, given $\mathcal{T}$ and a set Ξ of ΣQ-types, precise $\mathcal{T}$-realisability of Ξ is decidable in NP. To prove the Π_2^p-upper bound we show that sets Ξ of polynomial size in the size of $\mathcal{T}$ are enough.

Lemma A.5 *Suppose that a set Ξ of $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types is precisely $\mathcal{T}_1$-realisable but not precisely $\mathcal{T}_2$-realisable. Let Ω be the set of role names and their inverses that occur in $\mathcal{T}_1 \cup \mathcal{T}_2$. Then there is some $\Theta \subseteq \Xi$ with $|\Theta| \leq |\Omega| + 1$ such that Θ is precisely $\mathcal{T}_1$-realisable but not precisely $\mathcal{T}_2$-realisable.*

Proof. By Lemma A.4, for every $\boldsymbol{t} \in \Xi$, there is $\Omega_0 \subseteq \Omega$ such that the set $\Theta_{\boldsymbol{t}} = \{\boldsymbol{t}\} \cup \{\boldsymbol{t}_R \mid R \in \Omega_0\}$ is precisely $\mathcal{T}_1$-realisable. But then at least one of these $\Theta_{\boldsymbol{t}}$, for $\boldsymbol{t} \in \Xi$, is as required, for otherwise, if all of them turned out to be precisely $\mathcal{T}_2$-realisable, the disjoint union of models $\mathcal{I}_{\boldsymbol{t}}$ for $\mathcal{T}_2$ precisely realising $\Theta_{\boldsymbol{t}}$ would precisely realise the whole Ξ, which is impossible. ❑

Theorem A.6 *The Σ-query, strong Σ-concept and strong Σ-query entailment problems are all Π_2^p-complete for DL-Lite$_{bool}$.*

Proof. We check criterion **(pr)** of Lemma A.3. Let Σ be a signature and Ω the set of role names and their inverses that occur in $\mathcal{T}_1 \cup \mathcal{T}_2$. We may assume that $\Sigma \subseteq sig(\mathcal{T}_1 \cup \mathcal{T}_2)$. By Lemma A.4, for both $\mathcal{T} = \mathcal{T}_1$ and $\mathcal{T} = \mathcal{T}_2$, it is decidable in NP (in $|\mathcal{T}_1 \cup \mathcal{T}_2|$) whether a set Ξ of $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types of size $\leq |\Omega| + 1$ is precisely $\mathcal{T}$-realisable. The Σ_2^p algorithm deciding whether there exists a set of $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types that is precisely $\mathcal{T}_1$-realisable but not precisely $\mathcal{T}_2$-realisable is as follows:

1. Guess a set Ξ of $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$-types of size $\leq |\Omega| + 1$.
2. Check, using an NP-oracle, whether (i) Ξ is precisely $\mathcal{T}_1$-realisable, and whether (ii) Ξ is not precisely $\mathcal{T}_2$-realisable.
3. Return '$\mathcal{T}_1$ Σ-query entails $\mathcal{T}_2$' if the answers to (i) and (ii) are both positive.

By Lemmas A.3 and A.5, $\mathcal{T}_1$ Σ-query entails $\mathcal{T}_2$ if, and only if, the algorithm says so. ❑

Finally, we give proof sketches of Theorems 18, 22 and 21 on uniform interpolation, again restricting ourselves to the case of *DL-Lite$_{bool}$*.

Proof of Theorem 18. Let $\mathcal{T}$ be a TBox in *DL-Lite$_{bool}$* and Σ a signature. Define $\mathcal{T}_\Sigma$ to be the set of all concept inclusions of the form $\prod_{D \in \boldsymbol{t}} D \sqsubseteq \bot$, where $\boldsymbol{t}$ is a $\Sigma Q_\mathcal{T}$-type which is not $\mathcal{T}$-realisable. It follows immediately from Theorem 11 that $\mathcal{T}_\Sigma$ is a uniform interpolant of $\mathcal{T}$ w.r.t. Σ in *DL-Lite$_{bool}$*. As there are only exponentially many different $\Sigma Q_\mathcal{T}$-types and concept satisfiability in *DL-Lite$_{bool}$* is NP-complete, $\mathcal{T}_\Sigma$ can be constructed in exponential time. ❑

Proof of Theorem 21. Let $\mathcal{T}'_\Sigma$ be a uniform interpolant of $\mathcal{T}'$ w.r.t. Σ in *DL-Lite$^u_{bool}$*. Suppose that $\mathcal{T}$ Σ-query entails $\mathcal{T}'$ and $\mathcal{T} \not\models \varkappa$ for some $\varkappa \in \mathcal{T}'_\Sigma$. Let $\mathcal{I}$ be a model for $\mathcal{T}$ with $\mathcal{I} \not\models \varkappa$. Let Q be the set of numerical parameters from $\mathcal{T} \cup \mathcal{T}' \cup \{\varkappa\}$ and Ξ the set of ΣQ-types realised in $\mathcal{I}$. Then Ξ is $\mathcal{T}$-precisely realisable. Hence, by Theorem 11, Ξ is $\mathcal{T}'$-precisely realisable. Let $\mathcal{I}'$ be a model for $\mathcal{T}'$ which precisely realises Ξ. Then $\mathcal{I}' \not\models \varkappa$. It follows that $\mathcal{T}' \not\models \varkappa$, and so $\varkappa \notin \mathcal{T}'_\Sigma$, which is a contradiction.

Conversely, suppose $\mathcal{T}$ does not Σ-query entail $\mathcal{T}'$. By Theorem 11, there exists a set Ξ of $\Sigma Q_{\mathcal{T} \cup \mathcal{T}'}$ types which is precisely $\mathcal{T}$-realisable but not precisely $\mathcal{T}'$-realisable. Let

$$C \;=\; \Big(\prod_{\boldsymbol{t} \in \Xi} \exists u. \prod_{D \in \boldsymbol{t}} D\Big) \;\sqcap\; \Big(\forall u. \bigsqcup_{\boldsymbol{t} \in \Xi} \prod_{D \in \boldsymbol{t}} D\Big).$$

Then $\mathcal{T} \not\models C \sqsubseteq \bot$ but $\mathcal{T}' \models C \sqsubseteq \bot$. It follows that $\mathcal{T}'_\Sigma \models C \sqsubseteq \bot$. So there exists $\varkappa \in \mathcal{T}'_\Sigma$ such that $\mathcal{T} \not\models \varkappa$. ❑

Proof of Theorem 22. Let $\mathcal{T}$ be a TBox in *DL-Lite*$_{bool}$ and Σ a signature. Define $\mathcal{T}_\Sigma$ to be the set containing all concept inclusions of the form $\bigsqcap_{D \in \boldsymbol{t}} D \sqsubseteq \bot$, where $\boldsymbol{t}$ is a $\Sigma Q_\mathcal{T}$-type which is not $\mathcal{T}$-realisable, as well as all concept inclusions of the form

$$\bigsqcap_{D \in \boldsymbol{t}} D \sqsubseteq \bigsqcup_{\Xi \in \boldsymbol{\Omega}} \Big(\bigsqcap_{\boldsymbol{t}' \in \Xi} \exists u. \bigsqcap_{D \in \boldsymbol{t}'} D \Big),$$

where $\boldsymbol{t}$ is a $\mathcal{T}$-realisable $\Sigma Q_\mathcal{T}$-type and $\boldsymbol{\Omega}$ is the set of all minimal sets Ξ of $\Sigma Q_\mathcal{T}$-types such that $\{\boldsymbol{t}\} \cup \Xi$ is precisely $\mathcal{T}$-realisable. One can show (see Lemma A.5) that all Ξ contain at most $2n$ types, where n is the number of role names in $\mathcal{T}$. It follows that $\mathcal{T}_\Sigma$ can be constructed in exponential time in the size of $\mathcal{T}$. It remains to show that $\mathcal{T}_\Sigma$ is a uniform interpolant. Clearly, $\mathcal{T} \models \varkappa$, for all $\varkappa \in \mathcal{T}_\Sigma$. For the converse direction, it is sufficient to show that each precisely $\mathcal{T}_\Sigma$-realisable set of $\Sigma Q_\mathcal{T}$-types is precisely $\mathcal{T}$-realisable. Let Ξ_0 be such a set. By the definition of $\mathcal{T}_\Sigma$, for each $\boldsymbol{t} \in \Xi_0$ there exists $\Xi_{\boldsymbol{t}} \subseteq \Xi_0$ such that $\{\boldsymbol{t}\} \cup \Xi_{\boldsymbol{t}}$ is $\mathcal{T}$-precisely realisable. Take the disjoint union of models for $\mathcal{T}$ realising $\{\boldsymbol{t}\} \cup \Xi_{\boldsymbol{t}}$, for $\boldsymbol{t} \in \Xi_0$. It is readily seen that this is a model for $\mathcal{T}$ precisely realising Ξ_0. ❑

References

Acciarri, A.; Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; Palmieri, M.; and Rosati, R. 2005. QuOnto: Querying ontologies. In *Proc. of AAAI*, 1670–1671.

Antoniou, G., and Kehagias, A. 2000. A note on the refinement of ontologies. *Int. J. of Intelligent Systems* 15:623–632.

Areces, C., and ten Cate, B. 2006. Hybrid logics. In Blackburn, P.; van Benthem, J.; and Wolter, F., eds., *Handbook of Modal Logic*. Elsevier.

Artale, A.; Calvanese, D.; Kontchakov, R.; and Zakharyaschev, M. 2007. DL-Lite in the light of first-order logic. In *Proc. of AAAI*, 361–366.

Benedetti, M. 2005. sKizzo: A suite to evaluate and certify QBFs. In Nieuwenhuis, R., ed., *Proc. of CADE–20*, 369–376.

Borgida, A.; Brachman, R.; McGuinness, D.; and Resnick, L. A. 1989. Classic: A structural data model for objects. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, 58–67.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2005. DL-Lite: Tractable description logics for ontologies. In *Proc. of AAAI*, 602–607.

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.; Poggi, A.; and Rosati, R. 2007. Mastro-I: Efficient integration of relational data through DL ontologies. In *DL*, 1670–1671.

Chang, C., and Keisler, H. 1990. *Model Theory*. Elsevier.

Diaconescu, R.; Goguen, J.; and Stefaneas, P. 1993. Logical support for modularisation. In Huet, G., and Plotkin, G., eds., *Logical Environments*, 83–130. Cambridge University Press, New York.

Ghilardi, S.; Lutz, C.; and Wolter, F. 2006. Did I damage my ontology? A case for conservative extensions in description logic. In *Proc. of KR*, 187–197.

Grau, B. C.; Horrocks, I.; Kazakov, Y.; and Sattler, U. 2007a. Just the right amount: Extracting modules from ontologies. In *Proc. of WWW*, 717–726.

Grau, B. C.; Horrocks, I.; Kazakov, Y.; and Sattler, U. 2007b. A logical framework for modularity of ontologies. In *Proc. of IJCAI*, 298–303.

Konev, B.; Lutz, C.; Walther, D.; and Wolter, F. 2007. Formal properties of modularisation. In H. Stuckenschmidt and S. Spaccapietra, eds., *Ontology Modularization*, Springer, 2008.

Lang, J.; Liberatore, P.; and Marquis, P. 2003. Propositional independence: formula-variable independence and forgetting. *JAIR*, 18:391–443.

Lin, F., and Reiter, R. 1994. Forget it! In *Proceedings of AAAI Fall Symposium on Relevance*, 154–159.

Lutz, C., and Wolter, F. 2007. Conservative extensions in the lightweight description logic $\mathcal{EL}$. In *Proc. of CADE*, 84–99.

Lutz, C.; Walther, D.; and Wolter, F. 2007. Conservative extensions in expressive description logics. In *Proc. of IJCAI*, 453–458.

Noy, N., and Musen, M. 2002. PromptDiff: a fixed-point algorithm for comparing ontology versions. In *Proc. of AAAI*, 744–750.

Pitts, A. 1992. On an interpretation of second-order quantification in first-order intuitionistic propositional logic. *J. of Symbolic Logic*, 57:33–52.

Samulowitz, H., and Bacchus, F. 2006. Binary clause reasoning in QBF. In *Proc. of SAT*, volume 4121 of *LNCS*, 353–367.

Schlobach, S., and Cornet, R. 2003. Non-standard reasoning services for debugging of description logic terminologies. In *Proc. of IJCAI*, 355–362.

Visser, A. 1996. Uniform interpolation and layered bisimulation. In *Gödel '96 (Brno, 1996)*, volume 6 of *Lecture Notes in Logic*. Springer, 139–164.

Wang, Z.; Wang, K.; Topor, R.; Pan, J. 2008. Forgetting Concepts in DL-Lite. In *Proc. of ESWC*, 245–257.

Zhang, L., and Malik, S. 2002a. Conflict driven learning in a quantified Boolean satisfiability solver. In *Proc. of ICCAD*, 442–449.

Zhang, L., and Malik, S. 2002b. Towards a symmetric treatment of satisfaction and conflicts in quantified Boolean formula evaluation. In *Proc. of CP*, volume 2470 of *LNCS*, 200–215.