Query and Predicate Emptiness in Description Logics

Franz Baader
TU Dresden, Germany
baader@inf.tu-dresden.de

Meghyn Bienvenu and **Carsten Lutz**
Universität Bremen, Germany
(meghyn|clu)@uni-bremen.de

Frank Wolter
University of Liverpool, UK
wolter@liverpool.ac.uk

Abstract

Ontologies can be used to provide an enriched vocabulary for the formulation of queries over instance data. We identify query emptiness and predicate emptiness as two central reasoning services in this context. Query emptiness asks whether a given query has an empty answer over all data sets formulated in a given signature. Predicate emptiness is defined analogously, but quantifies universally over all queries that contain a given predicate. In this paper, we determine the computational complexity of query emptiness and predicate emptiness in the $\mathcal{EL}$, DL-Lite, and $\mathcal{ALC}$-families of description logics, investigate the connection to ontology modules, and perform a practical case study to evaluate the new reasoning services.

Introduction

In recent years, the paradigm of ontology-based data access (OBDA) has gained increased popularity. The general idea is similar to querying in incomplete databases under constraints: an ontology is used to provide a conceptual model of the application domain; when querying instance data, an open-world semantics is adopted and the ontology is treated as a set of logical constraints that is used to derive additional answers. This approach to query answering has been taken up with particular verve in the context of ontologies formulated in a description logic (DL), see for example (Calvanese et al. 2009; Lutz, Toman, and Wolter 2009) and references therein. Since DLs are expressive logical languages, DL-based ontologies allow one to capture general constraints on the data, to provide additional vocabulary that can be used when formulating queries, and to translate between vocabularies in the case that the data vocabulary is different from the query vocabulary, which is common e.g. in a data integration context (Calvanese et al. 2007).

The OBDA approach is fueled by the recent availability of professional and comprehensive ontologies that aim at providing a 'standard vocabulary' for the targeted application domain. In particular, there are many popular "off-the-shelf" ontologies in the bio-medical domain such as SNOMED CT, NCI, and Galen, which are all formulated in a DL and allow an easy and inexpensive adoption of OBDA in bio-medical applications such as querying electronic medical records (Patel et al. 2007). Such ontologies typically have a very broad coverage and often contain tens or even hundreds of thousands of predicates that embrace various subject areas such as anatomy, diseases, medication, and even social context and geographic location. On the one hand, this broadness enables the use of these ontologies in a large number of applications. On the other hand, it also means that only a *small fragment of the vocabulary* described in the ontology will occur in the instance data of any given application. The remaining predicates, which occur only in the ontology but not in the data, can often still be used in queries in a meaningful way, thus enriching the vocabulary that is available for query formulation—in fact, this is a main aim of OBDA. Due to the size and complexity of the involved ontologies and vocabularies, however, it is often difficult to know whether and how a given such predicate can be used in a query. In particular, even basic properties are difficult to check by hand, such as whether a designed query can ever produce a non-empty answer and, closely related, whether a given predicate can meaningfully be used in *any* query (details below).

To assist with such problems, we distinguish between two scenarios: queries may be fixed once and forever during the design of the application (*fixed query case*), or they may be formulated freely at runtime of the application (*free query case*). In the fixed query case, the query is formulated at a point in time when only the vocabulary (set of predicates) Σ of the data is known, but no concrete data exists. Note that this is a standard scenario in the database world, where the application design involves producing a schema that fixes the vocabulary, but often also the design of queries that are to be executed during runtime. To identify mistakes in the query, it is therefore a central problem to decide *query emptiness*, i.e., whether a given query q provides an empty answer over all data sets formulated in a given vocabulary Σ. This problem is a standard one in many subareas of database theory such as XML, see e.g. (Benedikt, Fan, and Geerts 2008) and references therein. In a DL context, it has first been considered in (Lubyte and Tessaris 2008).

In the case of free queries, it is not possible to consider a fixed set of queries at design time of the application. However, to assist with the formulation of queries at run-time, it is possible to select already at design time the set of those predicates that can meaningfully be used in a query. Of course, we again assume that the data vocabulary is fixed at application design time, as is standard in database design. Formally, we want to decide *predicate emptiness*, i.e., whether for a given predicate p and data vocabulary Σ, it is the case that all queries q that involve p yield an empty answer over all data sets formulated in Σ. Predicate emptiness is loosely related to predicate emptiness in datalog queries as studied e.g. in (Vardi 1989; Levy 1993).

The purpose of this paper it to perform an in-depth study of both query emptiness and predicate emptiness in the context of DLs. We consider the two most common kinds of queries in DL-based OBDA, instance queries and conjunctive queries, and determine the (un)decidability and computational complexity of query emptiness and predicate emptiness for a broad range of DLs including members of the $\mathcal{EL}$, DL-Lite, and $\mathcal{ALC}$ families of DLs. The results range from PTIME for basic members of the $\mathcal{EL}$ and DL-Lite families via NEXPTIME for basic members of the $\mathcal{ALC}$ family to undecidable for $\mathcal{ALC}$ extended with functional roles ($\mathcal{ALCF}$). Note that because of the presence of a selected data vocabulary Σ and the quantification over all data sets formulated in Σ, query emptiness and predicate emptiness do not reduce to standard reasoning problems such as query entailment or query containment. Formally, this is witnessed by our undecidability result for $\mathcal{ALCF}$, which should be contrasted with the decidability of query entailment and containment in this DL, cf. (Calvanese, De Giacomo, and Lenzerini 1998).

We also introduce a new notion of an ontology module, called Σ-substitute, which is based on predicate emptiness. Σ-substitutes can be used instead of the (much larger) original ontology when answering queries over data sets that are formulated in the restricted vocabulary Σ. Finally, we carry out a case study to give an idea of the number of predicates that are not in Σ but still non-empty in practical cases. Note that it is these predicates by which the ontology enriches the vocabulary available for query formulation. We also compare Σ-substitutes to existing notions of modularity, both in theory and practice.

Some proofs are deferred to the appendix, available at http://www.informatik.uni-bremen.de/∼clu/papers/index.html.

Preliminaries

We use standard notation for the syntax and semantics of DLs, please see standard references such as (Baader et al. 2003) for full details. In particular, we use $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ to denote countably infinite sets of concept names, role names, and individual names, C, D to denote (potentially) composite concepts, A, B for concept names, r, s for role names, and a, b for individual names. We consider various DLs throughout the paper, the most basic ones being $\mathcal{EL}$, which offers the constructors $\top$, $C \sqcap D$ and $\exists r.C$; and $\mathcal{ALC}$, which offers $\neg C$, $C \sqcap D$, and $\exists r.C$. We use the usual calligraphic letters to denote extensions, in particular $\mathcal{I}$ to denote the extension with inverse roles, $\mathcal{F}$ to denote the extension with functional roles, and subscript $\cdot_\bot$ to denote the extension with the bottom concept. When working with functional roles, we assume that a countably infinite number of such roles is available, instead of adding a concept constructor $(\leq 1\, r)$. The semantics of DLs is based on interpretations $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ as usual.

Ontologies are formalized in terms of *TBoxes*, by which we mean a set of *concept inclusions* (CIs) $C \sqsubseteq D$. Data is stored in an *ABox*, i.e., a set of *concept assertions* $A(a)$ and $\neg A(a)$ and *role assertions* $r(a, b)$. To distinguish this kind of ABox from ABoxes that admit composite concepts in concept assertions, we sometimes use the term *literal ABox*. We use $\mathsf{Ind}(\mathcal{A})$ to denote the set of individual names used in the ABox $\mathcal{A}$. An interpretation is a *model* of a TBox $\mathcal{T}$ (resp. ABox $\mathcal{A}$) if it satisfies all concept inclusions in $\mathcal{T}$ (resp. assertions in $\mathcal{A}$), where satisfaction is defined in the standard way. An ABox $\mathcal{A}$ is *consistent* w.r.t. a TBox $\mathcal{T}$ if $\mathcal{A}$ and $\mathcal{T}$ have a common model.

Instance queries (IQs) take the form $A(v)$, and *conjunctive queries (CQs)* take the form $\exists \vec{v}.\varphi(\vec{v}, \vec{u})$ where φ is a conjunction of atoms of the form $A(t)$ and $r(t, t')$ with t, t' *terms*, i.e., individual names or variables taken from a set $\mathsf{N_V}$. We use $\mathsf{term}(q)$ to denote the set of terms used in the query q. Note that we disallow composite concepts in queries, which is a realistic assumption for many applications and required to enable a straightforward definition of predicate emptiness below. Also note that instance queries can only be used to query concept names, but not role names. This is the traditional definition, which is due to the fact that role assertions can only be implied by an ABox if they are explicitly contained in it (and thus querying is 'trivial'). From now on, we use $\mathcal{IQ}$ to refer to the set of all IQs and $\mathcal{CQ}$ to refer to the set of all CQs.

Let $\mathcal{I}$ be an interpretation and q an (instance or conjunctive) query q with k answer variables $v_1, \ldots, v_k$. For $\vec{a} = a_1, \ldots, a_n \in \mathsf{N_I}$, an $\vec{a}$*- match* for q in $\mathcal{I}$ is a mapping $\pi : \mathsf{term}(q) \to \Delta^\mathcal{I}$ such that $\pi(a) = a^\mathcal{I}$ for all $a \in \mathsf{term}(q) \cap \mathsf{N_I}$, $\pi(t) \in A^\mathcal{I}$ for all $A(t) \in q$, and $(\pi(t_1), \pi(t_2)) \in r^\mathcal{I}$ for all $r(t_1, t_2) \in q$. We write $\mathcal{I} \models q[a_1, \ldots, a_k]$ if there is an $(a_1, \ldots, a_k)$-match of q in $\mathcal{I}$. For a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$, we write $\mathcal{T}, \mathcal{A} \models q[a_1, \ldots, a_k]$ if $\mathcal{I} \models q[a_1, \ldots, a_k]$ for all models $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$. In this case, $(a_1, \ldots, a_k)$ is a *certain answer* to q w.r.t. $\mathcal{A}$ and $\mathcal{T}$. We use $\mathsf{cert}_{\mathcal{T},\mathcal{A}}(q)$ to denote the set of all certain answers to q w.r.t. $\mathcal{A}$ and $\mathcal{T}$.

We use the term *predicate* to refer to a concept name or role name and *signature* to refer to a set of predicates (in the introduction, we informally called a signature a vocabulary). Then $\mathsf{sig}(q)$ denotes the set of predicates used in the query q, and similarly $\mathsf{sig}(\mathcal{T})$ (resp. $\mathsf{sig}(\mathcal{A})$) refers to the signature of a TBox $\mathcal{T}$ (resp. ABox $\mathcal{A}$). Given a signature Σ, a Σ-*ABox* (resp. Σ*-concept*) is an ABox (resp. concept) using predicates from Σ only.

In the context of query answering in DLs, it is sometimes useful to adopt the unique name assumption (UNA), which requires that $a^\mathcal{I} \neq b^\mathcal{I}$ for all interpretations $\mathcal{I}$ and all $a, b \in \mathsf{N_I}$ with $a \neq b$. The results obtained in this paper do not depend on the UNA. The following well-known lemma shows that the UNA does not make a difference in $\mathcal{ALCI}$ (and all its fragments such as $\mathcal{EL}$ and $\mathcal{ALC}$) because the certain answers to queries do not change.

Lemma 1. *Let $\mathcal{T}$ be an $\mathcal{ALCI}$-TBox, $\mathcal{A}$ an ABox, and $q \in \mathcal{L}$. Then $\mathsf{cert}_{\mathcal{T},\mathcal{A}}(q)$ is identical with and without the UNA.*

An analogous statement fails for $\mathcal{ALCF}$, e.g. because the ABox $\mathcal{A} = \{f(a,b), f(a,b')\}$, f a functional role, is consistent w.r.t. the empty TBox without the UNA (and thus $\mathsf{cert}_{\emptyset,\mathcal{A}}(A(v)) = \emptyset$), but inconsistent with the UNA (and thus $\mathsf{cert}_{\emptyset,\mathcal{A}}(A(v)) = \mathsf{N_I}$).

Query and Predicate Emptiness

The following definition introduces the central notions studied in this paper.[1]

Definition 2. Let $\mathcal{T}$ be a TBox, Σ a signature, and $\mathcal{L} \in \{\mathcal{IQ}, \mathcal{CQ}\}$ a query language. Then we call

- an $\mathcal{L}$-query q *empty for* Σ *given* $\mathcal{T}$ if for all Σ-ABoxes $\mathcal{A}$ that are consistent w.r.t. $\mathcal{T}$, we have $\mathsf{cert}_{\mathcal{T},\mathcal{A}}(q) = \emptyset$.
- a predicate S *$\mathcal{L}$-empty for* Σ *given* $\mathcal{T}$ if all $\mathcal{L}$-queries q with $S \in \mathsf{sig}(q)$ are empty for Σ given $\mathcal{T}$.

We quantify over *all* ABoxes that are formulated in the ABox to address typical database applications in which the instance data changes frequently, and thus deciding emptiness based on a concrete ABox is not of much interest. As an example, assume that ABoxes are formulated in the signature

$$\Sigma = \{\mathsf{Person}, \mathsf{hasDisease}, \mathsf{DiseaseA}, \mathsf{DiseaseB}\}$$

where here and in the following, all upper case words are concept names and all lower case ones are role names. This signature is fixed in the application design phase, similar to schema design in databases. For the TBox, we take

$$\begin{array}{rcll} \mathcal{T} & = & \{\mathsf{Person} & \sqsubseteq \exists \mathsf{hasFather}.(\mathsf{Person} \sqcap \mathsf{Male}), \\ & & \mathsf{DiseaseA} & \sqsubseteq \mathsf{InfectiousDisease}\}, \end{array}$$

Then both the IQ $\mathsf{InfectiousDisease}(v)$ and the CQ $\exists v.\mathsf{hasFather}(u,v)$ are non-empty for Σ given $\mathcal{T}$ despite using predicates that cannot occur in the data, as witnessed by the Σ-ABoxes $\{\mathsf{DiseaseA}(a)\}$ and $\{\mathsf{Person}(a)\}$, respectively. This illustrates how the TBox $\mathcal{T}$ enriches the vocabulary that is available for query formulation. By contrast, the CQ

$$\begin{array}{l} \exists v, v'.(\mathsf{hasFather}(u,v) \wedge \\ \qquad \mathsf{hasDisease}(v,v') \wedge \mathsf{InfectiousDisease}(v')), \end{array}$$

which uses the same predicates plus an additional one from the data signature, is empty for Σ given $\mathcal{T}$.

Regarding predicate emptiness, it is interesting to observe that the choice of the query language is important. For example, the predicate Male is $\mathcal{IQ}$-empty for Σ given $\mathcal{T}$, but not $\mathcal{CQ}$-empty as witnessed by the Σ-ABox $\{\mathsf{Person}(a)\}$ and the CQ $\exists v.\mathsf{Male}(v)$. It thus makes no sense to use Male in instance queries over Σ-ABoxes given $\mathcal{T}$, whereas it can be meaningfully used in conjunctive queries.

As every IQ is also a CQ, a predicate that is $\mathcal{CQ}$-empty must also be $\mathcal{IQ}$-empty. As illustrated by the above example, the converse does not hold. Also note that all role names are $\mathcal{IQ}$-empty for Σ given any $\mathcal{T}$ since a role name cannot occur in an instance query. By contrast, $\mathsf{hasFather}$ is clearly not $\mathcal{CQ}$-empty in the above example.

[1]In the workshop paper (Baader et al. 2009), the complement of "$\mathcal{L}$-predicate emptiness" was called "$\mathcal{L}$-relevance".

$$\begin{tikzcd} \text{$\mathcal{IQ}$-query emptiness} \arrow[r, phantom, "="] \arrow[d, "\text{Trivial}"'] & \text{$\mathcal{IQ}$-predicate emptiness} \\ \text{$\mathcal{CQ}$-query emptiness} & \text{$\mathcal{CQ}$-predicate emptiness} \arrow[l, "\text{Lemma 5}"] \arrow[u, "\text{Lemma 6}"'] \end{tikzcd}$$

Figure 1: Polytime reductions between emptiness notions.

It follows from Lemma 1 that, in $\mathcal{ALCI}$ and its fragments, query emptiness and predicate emptiness are oblivious as to whether or not the UNA is made, both for $\mathcal{IQ}$ and $\mathcal{CQ}$. As established by the following lemma, this is also true in $\mathcal{ALCF}$—despite the fact that the certain answers to queries can differ with and without the UNA. A proof is in the appendix.

Lemma 3. *Let $\mathcal{T}$ be an $\mathcal{ALCF}$-TBox. Then each CQ q is empty for Σ given $\mathcal{T}$ with the UNA iff it is empty for Σ given $\mathcal{T}$ without the UNA.*

Since all DLs considered in this paper are fragments of $\mathcal{ALCI}$ or $\mathcal{ALCF}$, we are thus free to adopt the UNA or not. In the remainder of this paper, we generally make the UNA unless explicitly noted otherwise.

Definition 2 gives rise to four natural decision problems.

Definition 4. Let $\mathcal{L} \in \{\mathcal{IQ}, \mathcal{CQ}\}$. Then

- *$\mathcal{L}$-query emptiness* is the problem of deciding, given a TBox $\mathcal{T}$, a signature Σ, and an $\mathcal{L}$-query q, whether q is empty for Σ given $\mathcal{T}$;
- *$\mathcal{L}$-predicate emptiness* means to decide, given a TBox $\mathcal{T}$, a signature Σ, and a predicate S, whether S is $\mathcal{L}$-empty for Σ given $\mathcal{T}$.

Clearly, these four problems are intimately related. In particular, $\mathcal{IQ}$-query emptiness and $\mathcal{IQ}$-predicate emptiness are effectively the same problem since an instance query consists only of a single predicate. For this reason, we will from now on disregard $\mathcal{IQ}$-predicate emptiness and only speak of $\mathcal{IQ}$-query emptiness. In the $\mathcal{CQ}$ case, things are different. Indeed, the following lemma shows that $\mathcal{CQ}$-predicate emptiness corresponds to $\mathcal{CQ}$-query emptiness where CQs are restricted to a very simple form. It is an easy consequence of the fact that, since composite concepts in queries are disallowed, CQs are purely positive, existential, and conjunctive.

Lemma 5. *$A \in \mathsf{N_C}$ (resp. $r \in \mathsf{N_R}$) is $\mathcal{CQ}$-predicate empty for Σ given $\mathcal{T}$ iff the conjunctive query $\exists v.A(v)$ (resp. $\exists v, v'.r(v,v')$) is empty for Σ given $\mathcal{T}$.*

Lemma 5 allows us to consider only queries of the form $\exists v.A(v)$ and $\exists v, v'.r(v,v')$ when dealing with $\mathcal{CQ}$-predicate emptiness. From now on, we do this without further notice.

Trivially, $\mathcal{IQ}$-query emptiness is a special case of $\mathcal{CQ}$-query emptiness. The following observation is less obvious.

Lemma 6. *In any DL that includes the constructor $\exists r.C$, $\mathcal{CQ}$-predicate emptiness can be polynomially reduced to $\mathcal{IQ}$-query emptiness.*

Proof. Let $\mathcal{T}$ be a TBox, Σ a signature, B a concept name that does not occur in $\mathcal{T}$ and Σ, and s a role name that does not occur in $\mathcal{T}$ and Σ. We prove that

1. A is $\mathcal{CQ}$-empty for Σ given $\mathcal{T}$ iff the IQ $B(v)$ is empty for $\Sigma \cup \{s\}$ given the TBox $\mathcal{T}' = \mathcal{T} \cup \mathcal{T}_B \cup \{A \sqsubseteq B\}$, where $\mathcal{T}_B = \{\exists r.B \sqsubseteq B \mid r = s \text{ or } r \text{ occurs in } \mathcal{T}\}$;
2. r is $\mathcal{CQ}$-empty for Σ given $\mathcal{T}$ iff the IQ $B(v)$ is empty for $\Sigma \cup \{s\}$ given the TBox $\mathcal{T}' = \mathcal{T} \cup \mathcal{T}_B \cup \{\exists r.\top \sqsubseteq B\}$, where $\mathcal{T}_B$ is as above.

The proofs of Points 1 and 2 are similar and we concentrate on Point 1. First suppose that A is $\mathcal{CQ}$ predicate non-empty for Σ given $\mathcal{T}$. Then there is a Σ-ABox $\mathcal{A}$ such that $\mathcal{T}, \mathcal{A} \models \exists v.A(v)$. Choose an $a_0 \in \mathsf{Ind}(\mathcal{A})$ and set $\mathcal{A}' := \mathcal{A} \cup \{s(a_0, b) \mid b \in \mathsf{Ind}(\mathcal{A})\}$. Using the fact that $\mathcal{T}, \mathcal{A} \models \exists v.A(v)$ and the definition of $\mathcal{A}'$ and $\mathcal{T}'$, it can be shown that $\mathcal{T}', \mathcal{A}' \models B(a_0)$. For the converse direction, suppose that B is $\mathcal{IQ}$ query non-empty for $\Sigma \cup \{s\}$ given $\mathcal{T}'$. Then there is a $\Sigma \cup \{s\}$-ABox $\mathcal{A}'$ such that $\mathcal{T}', \mathcal{A}' \models B(a)$ for some $a \in \mathsf{Ind}(\mathcal{A}')$. Let $\mathcal{A}$ be obtained from $\mathcal{A}'$ by removing all assertions $s(a, b)$. Using the fact that $\mathcal{T}', \mathcal{A}' \models B(a)$ and the definition of $\mathcal{A}'$ and $\mathcal{T}'$, it can be shown that $\mathcal{T}, \mathcal{A} \models \exists v.A(v)$. □

Figure 1 gives an overview of the available polytime reductions between our four (rather: three) problems. In terms of computational complexity, $\mathcal{CQ}$-query emptiness is thus (potentially) the hardest problem, while $\mathcal{CQ}$-predicate emptiness is the simplest.

We remark that it is the use of the signature Σ that makes our approach technically interesting. Indeed, deciding query and predicate emptiness is simple whenever Σ contains all symbols used in the TBox. By the described reductions, it suffices to consider CQ-query emptiness. For a CQ $q = \exists \vec{v}.\varphi(\vec{v}, \vec{u})$, we set $\mathcal{A}_q = \{A(a_t) \mid A(t) \text{ is a conjunct in } \varphi\} \cup \{r(a_t, a_{t'}) \mid r(t, t') \text{ is a conjunct in } \varphi\}$ where $a_t = t$ if $t \in \mathsf{N_I}$.

Theorem 7. *Let $\mathcal{T}$ be an $\mathcal{ALCFI}$-TBox, Σ a signature with $\mathsf{sig}(\mathcal{T}) \subseteq \Sigma$, and q a CQ. Then q is empty for Σ given $\mathcal{T}$ iff $\mathsf{sig}(q) \not\subseteq \Sigma$ or $\mathcal{A}_q$ is inconsistent w.r.t. $\mathcal{T}$.*

Proof. ("If") Assume that q is non-empty for Σ given $\mathcal{T}$. Then there is a Σ-ABox $\mathcal{A}$ that is consistent w.r.t. $\mathcal{T}$ and such that $\mathsf{cert}_{\mathcal{T},\mathcal{A}}(q) \neq \emptyset$. This clearly implies $\mathsf{sig}(q) \subseteq \Sigma$ since otherwise there is a predicate in $\mathsf{sig}(q) \setminus \Sigma$ and we can find a model of $\mathcal{A}$ and $\mathcal{T}$ in which this predicate is interpreted as the empty set, which would mean $\mathsf{cert}_{\mathcal{T},\mathcal{A}}(q) = \emptyset$. It thus remains to show that $\mathcal{A}_q$ is consistent w.r.t. $\mathcal{T}$. To this end, let $\mathcal{I}$ be a model of $\mathcal{A}$ and $\mathcal{T}$, $(a_1, \ldots, a_n) \in \mathsf{cert}_{\mathcal{T},\mathcal{A}}(q)$, and π an $(a_1, \ldots, a_n)$-match for q in $\mathcal{I}$. Modify $\mathcal{I}$ by setting $a_t^{\mathcal{I}} = \pi(t)$ for all terms t used in q. It is readily checked that the modified $\mathcal{I}$ is a model of $\mathcal{A}_q$ and $\mathcal{T}$, thus $\mathcal{A}_q$ is consistent w.r.t. $\mathcal{T}$ as required.

("Only if") Assume that $\mathsf{sig}(q) \subseteq \Sigma$ and $\mathcal{A}_q$ is consistent w.r.t. $\mathcal{T}$. Then $\mathsf{sig}(\mathcal{A}_q) \subseteq \Sigma$. Since clearly $\mathsf{cert}_{\mathcal{T},\mathcal{A}_q}(q) \neq \emptyset$, this means that q is non-empty for Σ given $\mathcal{T}$. □

The $\mathcal{EL}$ Family

We study query and predicate emptiness in the $\mathcal{EL}$ family of lightweight DLs (Baader, Brandt, and Lutz 2005). In particular, we show that all three problems can be decided in polynomial time in plain $\mathcal{EL}$, whereas already $\mathcal{CQ}$-predicate emptiness is ExpTime-complete in $\mathcal{ELI}$ and $\mathcal{EL}_\bot$. It is interesting to contrast these results with the complexity of subsumption and instance checking, which can be decided in polynomial time in the case of $\mathcal{EL}$ and $\mathcal{EL}_\bot$ and are ExpTime-complete in $\mathcal{ELI}$ (Baader, Brandt, and Lutz 2005; 2008).

Throughout this section, we assume that the UNA is not imposed (cf. Lemma 1). Since DLs of the $\mathcal{EL}$ family do not offer negation, it may seem more natural to define emptiness based on *positive* ABoxes, i.e., ABoxes in which all concept assertions are of the form $A(a)$ with A a concept name. The following lemma shows that this does not make a difference, which allows us to henceforth restrict our attention to positive ABoxes.

Lemma 8. *For every $\mathcal{ELI}_\bot$-TBox $\mathcal{T}$, literal ABox $\mathcal{A}$ consistent w.r.t. $\mathcal{T}$, and conjunctive query q, we have $\mathsf{cert}_{\mathcal{T},\mathcal{A}}(q) = \mathsf{cert}_{\mathcal{T},\mathcal{A}^-}(q)$, where $\mathcal{A}^-$ is the restriction of $\mathcal{A}$ to assertions of the form $A(a)$ and $r(a, b)$.*

The proof of Lemma 8 and subsequent results relies on canonical models, whose definition we recall here.

Let $\mathcal{T}$ be an $\mathcal{ELI}_\bot$-TBox and $\mathcal{A}$ a positive ABox that is consistent w.r.t. $\mathcal{T}$. For $a \in \mathsf{Ind}(\mathcal{A})$, a *path* for $\mathcal{A}$ and $\mathcal{T}$ is a finite sequence $a\, r_1\, C_1\, r_2 C_2 \cdots r_n\, C_n$, $n \geq 0$, where the C_i are concepts that occur in $\mathcal{T}$ (potentially as a subconcept) and the r_i are roles such that the following conditions are satisfied:

- $a \in \mathsf{Ind}(\mathcal{A})$,
- $\mathcal{T}, \mathcal{A} \models \exists r_1.C_1(a)$ if $n \geq 1$,
- $\mathcal{T} \models C_i \sqsubseteq \exists r_{i+1}.C_{i+1}$ for $1 \leq i < n$.

The domain $\Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ of the *canonical model* $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ for $\mathcal{T}$ and $\mathcal{A}$ is the set of all paths for $\mathcal{A}$ and $\mathcal{T}$. If $p \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \setminus \mathsf{Ind}(\mathcal{A})$, then $\mathsf{tail}(p)$ denotes the last concept C_n in p. Set

$$\begin{array}{rcl} A^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} & := & \{a \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{T}, \mathcal{A} \models A(a)\} \cup \\ & & \{p \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \setminus \mathsf{Ind}(\mathcal{A}) \mid \mathcal{T} \models \mathsf{tail}(p) \sqsubseteq A\} \\ r^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} & := & \{(a, b) \mid r(a, b) \in \mathcal{A}\} \cup \\ & & \{(p, q) \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \times \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \mid \\ & & \qquad q = p \cdot r\, C \text{ for some concept } C\} \\ a^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} & := & a \text{ for all } a \in \mathsf{Ind}(\mathcal{A}) \end{array}$$

It is standard to prove the following.

Lemma 9. *$\mathcal{I}_{\mathcal{T},\mathcal{A}}$ is a model of $\mathcal{T}$ and $\mathcal{A}$ such that:*

1. *for any $a \in \mathsf{Ind}(\mathcal{A})$ and $\mathcal{ELI}_\bot$-concept C, $a^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \in C^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ iff $\mathcal{T}, \mathcal{A} \models C(a)$;*
2. *for any k-ary conjunctive query q and $(a_1, \ldots, a_k) \in \mathsf{N_I}^k$, $\mathcal{I}_{\mathcal{T},\mathcal{A}} \models q[a_1, \ldots, a_k]$ iff $(a_1, \ldots, a_k) \in \mathsf{cert}_{\mathcal{T},\mathcal{A}}(q)$.*

We now prove Lemma 8.

Proof of Lemma 8. As "$\supseteq$" is trivial, we concentrate on "$\subseteq$". Suppose $(a_1, \ldots, a_k) \notin \mathsf{cert}_{\mathcal{T},\mathcal{A}^-}(q)$. Then there is a model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}^-$ such that $\mathcal{I} \not\models q[a_1, \ldots, a_k]$. By

Point 2 of Lemma 9, $\mathcal{I}_{\mathcal{T},\mathcal{A}^-} \not\models q[a_1,\ldots,a_k]$. To prove that $(a_1,\ldots,a_k) \notin \mathsf{cert}_{\mathcal{T},\mathcal{A}}(q)$, it thus suffices to show that $\mathcal{I}_{\mathcal{T},\mathcal{A}^-}$ satisfies all negative concept assertions in $\mathcal{A}$. Let $\neg A(a) \in \mathcal{A}$. Since $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$, we get $\mathcal{T},\mathcal{A} \not\models A(a)$, hence $\mathcal{T},\mathcal{A}^- \not\models A(a)$. By Point 1 of Lemma 9, $a^{\mathcal{I}_{\mathcal{T},\mathcal{A}^-}} \notin A^{\mathcal{I}_{\mathcal{T},\mathcal{A}^-}}$, so we are done. □

As another preliminary, we show that in the $\mathcal{EL}$ family, a converse of Lemma 6 can be established. We have found no such reduction for DL-Lite and expressive DLs. Note that, by the example given after Definition 2, the two emptiness notions considered in Lemma 10 do not coincide even in $\mathcal{EL}$.

Lemma 10. *In $\mathcal{ELI}_\bot$, $\mathcal{IQ}$-query emptiness can be polynomially reduced to $\mathcal{CQ}$-predicate emptiness.*

Proof. We claim that the instance query $A(v)$ is empty for Σ given $\mathcal{T}$ iff B is $\mathcal{CQ}$-empty for $\Sigma \cup \{X\}$ given the TBox $\mathcal{T}' = \mathcal{T} \cup \{A \sqcap X \sqsubseteq B\}$, where B and X are concept names that do not occur in $\mathcal{T}$.

For the "if" direction, assume that A is $\mathcal{IQ}$ non-empty for Σ given $\mathcal{T}$ and let $\mathcal{A}$ be a positive Σ-ABox such that $\mathcal{T},\mathcal{A} \models A(a)$ for some $a \in \mathsf{Ind}(\mathcal{A})$. Set $\mathcal{A}' := \mathcal{A} \cup \{X(a)\}$. It is easy to see that $\mathcal{T}',\mathcal{A}' \models \exists v.B(v)$ and thus B is $\mathcal{CQ}$-predicate non-empty for $\Sigma \cup \{X\}$ given $\mathcal{T}'$.

For the "only if" direction, assume that B is $\mathcal{CQ}$ non-empty for $\Sigma \cup \{X\}$ given $\mathcal{T}'$ and let $\mathcal{A}'$ be a positive $\Sigma \cup \{X\}$-ABox which is consistent with $\mathcal{T}'$ and such that $\mathcal{T}',\mathcal{A}' \models \exists v.B(v)$. By Point 2 of Lemma 9, $\mathcal{I}_{\mathcal{T}',\mathcal{A}'} \models \exists v.B(v)$. We want to show that there is an $a \in \mathsf{Ind}(\mathcal{A}')$ with $a^{\mathcal{I}_{\mathcal{T}',\mathcal{A}'}} \in B^{\mathcal{I}_{\mathcal{T}',\mathcal{A}'}}$. Assume to the contrary that there is no such a. Let $\mathcal{I}$ be obtained from $\mathcal{I}_{\mathcal{T}',\mathcal{A}'}$ by setting

$$\begin{array}{rcl} X^{\mathcal{I}} & := & \{a^{\mathcal{I}_{\mathcal{T}',\mathcal{A}'}} \mid a \in \mathsf{Ind}(\mathcal{A}')\} \\ B^{\mathcal{I}} & := & B^{\mathcal{I}_{\mathcal{T}',\mathcal{A}'}} \cap X^{\mathcal{I}} \end{array}$$

It is easy to see that $\mathcal{I}$ is still a model of $\mathcal{T}'$ and $\mathcal{A}'$. By our assumption that there is no $a \in \mathsf{Ind}(\mathcal{A}')$ with $a^{\mathcal{I}_{\mathcal{T}',\mathcal{A}'}} \in B^{\mathcal{I}_{\mathcal{T}',\mathcal{A}'}}$, we have $B^{\mathcal{I}} = \emptyset$, in contradiction to $\mathcal{T}',\mathcal{A}' \models \exists v.B(v)$. Thus, the desired $a \in \mathsf{Ind}(\mathcal{A}')$ exists. By Point 1 of Lemma 9, $a^{\mathcal{I}_{\mathcal{T}',\mathcal{A}'}} \in B^{\mathcal{I}_{\mathcal{T}',\mathcal{A}'}}$ implies that $\mathcal{T}',\mathcal{A}' \models B(a)$. By definition of $\mathcal{T}'$, this implies $\mathcal{T}',\mathcal{A}' \models A(a)$. Again by definition of $\mathcal{T}'$, this clearly implies $\mathcal{T},\mathcal{A} \models A(a)$, where $\mathcal{A}$ is obtained from $\mathcal{A}'$ by dropping all concept assertions of the form $X(b)$. Since $\mathcal{A}$ is a Σ-ABox and consistent w.r.t. $\mathcal{T}$ (since $\mathcal{A}'$ is consistent w.r.t. $\mathcal{T}$'), it witnesses that $A(v)$ is non-empty for Σ given $\mathcal{T}$. □

We now show that there is a very simple way to decide $\mathcal{IQ}$-query emptiness and $\mathcal{CQ}$-predicate emptiness in $\mathcal{EL}$, based on the following lemma. Take an individual name a_Σ and define the *total Σ-ABox* as $\mathcal{A}_\Sigma := \{A(a_\Sigma) \mid A \in \Sigma\} \cup \{r(a_\Sigma, a_\Sigma) \mid r \in \Sigma\}$.

Lemma 11. *For all conjunctive queries q, q is empty for Σ given $\mathcal{T}$ iff $\mathsf{cert}_{\mathcal{T},\mathcal{A}_\Sigma}(q) = \emptyset$.*

Proof. It is not hard to verify that q is empty for Σ given $\mathcal{T}$ iff the CQ obtained by replacing in q all individual names with answer variables is empty for Σ given $\mathcal{T}$. Thus, we can w.l.o.g. assume that q does not contain any individual names. The "only if" direction is trivial. For the "if" direction, we consider the contrapositive. Thus, let q be non-empty for Σ given $\mathcal{T}$. By Lemma 8, there is a positive Σ-ABox $\mathcal{A}$ consistent with $\mathcal{T}$ such that $\mathsf{cert}_{\mathcal{T},\mathcal{A}}(q) \neq \emptyset$. Every model $\mathcal{I}$ of $\mathcal{A}_\Sigma$ and $\mathcal{T}$ can be turned into a model $\mathcal{I}'$ of $\mathcal{A}$ and $\mathcal{T}$ such that

1. $\mathcal{I}$ and $\mathcal{I}'$ are identical modulo the interpretation of individual names and
2. if $d = a^{\mathcal{I}'}$ for some $a \in \mathsf{N_I}$ (and thus an answer variable can be mapped to a in $\mathcal{I}'$), then $d = b^{\mathcal{I}}$ for some $b \in \mathsf{N_I}$ (and thus an answer variable can be mapped to a in $\mathcal{I}$)

by simply setting $b^{\mathcal{I}'} := a_\Sigma^{\mathcal{I}}$ for all individual names b. Given that q does not comprise individual names and using Points 1 and 2, it can also be verified that any match of q in $\mathcal{I}'$ can be reproduced in $\mathcal{I}$. It follows that $\mathsf{cert}_{\mathcal{T},\mathcal{A}}(q) \subseteq \mathsf{cert}_{\mathcal{T},\mathcal{A}_\Sigma}(q)$, whence $\mathsf{cert}_{\mathcal{T},\mathcal{A}_\Sigma}(q) \neq \emptyset$ as required. □

Lemma 11 provides a polytime reduction of $\mathcal{CQ}$-predicate emptiness (and thus also $\mathcal{IQ}$-query emptiness) to CQ-query answering where CQs are of the form $\exists v.A(v)$ or $\exists v, v'.r(v, v')$. The latter problem can be trivially reduced to the instance problem in $\mathcal{EL}$ enriched with the universal role u, where $u^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for all interpretations $\mathcal{I}$. Since it is easy to extend the standard PTIME algorithm for the instance problem in $\mathcal{EL}$ (Baader, Brandt, and Lutz 2005) to this enriched version of $\mathcal{EL}$, we obtain the following theorem.

Theorem 12. *In $\mathcal{EL}$, $\mathcal{IQ}$-query emptiness and $\mathcal{CQ}$-predicate emptiness can be decided in* PTIME.

Note that we need very little for the proof of Theorem 12 to go through: it suffices that $\mathcal{A}_\Sigma$ is consistent with every TBox. It follows that for all DLs of this sort, deciding $\mathcal{IQ}$-query emptiness and $\mathcal{CQ}$-predicate emptiness has the same complexity (modulo complementation) as subsumption/instance checking in the DL enriched with the universal role. The upper bound is obtained as in the proof of Theorem 12, based on instance checking. For the lower bound, note that C is subsumed by D w.r.t. $\mathcal{T}$ iff $B(v)$ is non-empty for the signature $\{A\}$ given $\mathcal{T} \cup \{A \sqsubseteq C, D \sqsubseteq B\}$, where $A, B \notin \mathsf{sig}(C, D, \mathcal{T})$. We thus obtain the following result for the DL $\mathcal{ELI}$, for which subsumption and instance checking are EXPTIME-complete (Baader, Brandt, and Lutz 2008), even with the addition of the universal role.

Theorem 13. *In $\mathcal{ELI}$, $\mathcal{IQ}$-query emptiness and $\mathcal{CQ}$-predicate emptiness are* EXPTIME*-complete.*

The simplest extension of $\mathcal{EL}$ in which the total ABox $\mathcal{A}_\Sigma$ is not consistent w.r.t. every TBox is $\mathcal{EL}_\bot$. Here, deciding $\mathcal{IQ}$-query emptiness and $\mathcal{CQ}$-predicate emptiness is significantly harder than deciding subsumption/instance checking (which can be decided in polynomial time).

Theorem 14. *In $\mathcal{EL}_\bot$, $\mathcal{IQ}$-query emptiness and $\mathcal{CQ}$-predicate emptiness are* EXPTIME*-hard.*

Proof (idea). We consider $\mathcal{IQ}$-query emptiness. The two main ingredients to the proof are: first, a proof of the fact that if the IQ $B(v)$ is non-empty for a signature Σ given an

$\mathcal{EL}_\bot$-TBox $\mathcal{T}$, then there is a Σ-concept C which is satisfiable w.r.t. $\mathcal{T}$ with $\mathcal{T} \models C \sqsubseteq B$. And second, a careful analysis of the reduction underlying Theorem 36 in (Lutz and Wolter 2009) which shows that it is EXPTIME-hard to decide for a given $\mathcal{EL}_\bot$-TBox $\mathcal{T}$, signature Σ, and concept name B, whether there exists a Σ-concept C such that C is satisfiable w.r.t. $\mathcal{T}$ and $\mathcal{T} \models C \sqsubseteq B$. $\square$

We now provide a matching upper bound for Theorem 14.

Theorem 15. *In $\mathcal{EL}_\bot$, $\mathcal{IQ}$-query emptiness and $\mathcal{CQ}$-predicate emptiness are* EXPTIME*-complete.*

Proof (idea). We first show that if an ABox witnesses non-emptiness of an IQ $A(v)$, then there is a tree-shaped ABox that witnesses non-emptiness of $A(v)$. This enables a decision procedure based on automata on finite trees, which is detailed in the appendix. $\square$

We now take a glimpse at $\mathcal{CQ}$-query emptiness showing that, in $\mathcal{EL}$, this problem is not harder than $\mathcal{IQ}$-query emptiness.

Theorem 16. *In $\mathcal{EL}$, $\mathcal{CQ}$-query emptiness can be decided in* PTIME.

Proof (idea). The proof utilizes Lemma 11 and proceeds by showing that in the case of the total Σ-ABox $\mathcal{A}_\Sigma$, emptiness of $\mathsf{cert}_{\mathcal{T},\mathcal{A}_\Sigma}(q)$ for a CQ q can be decided in polytime. $\square$

We do not pinpoint the exact complexity of $\mathcal{CQ}$-query emptiness in $\mathcal{ELI}$ and $\mathcal{EL}_\bot$. Note though that an EXPTIME lower bound is obtained from Theorems 13 and 14, and a 2-EXPTIME upper bound from Theorem 27 established later.

The DL-Lite Family

We study query and predicate emptiness in the DL-Lite family of description logics (Calvanese et al. 2007). In particular, we show that $\mathcal{IQ}$-query emptiness and $\mathcal{CQ}$-predicate emptiness are typically decidable in polynomial time for members of the DL-Lite family without conjunctions on the left-hand side of CIs such as DL-Lite$_{core}$, DL-Lite$_\mathcal{R}$, and DL-Lite$_\mathcal{F}$ (Calvanese et al. 2007). In contrast, $\mathcal{CQ}$-query emptiness turns out to be coNP-complete in these DLs. The situation changes for DL-Lite dialects in which conjunctions are admitted on the left-hand side of CIs such as in DL-Lite$_{horn}$ (Artale et al. 2009): in this case, all three problems are coNP-complete.

We start by proving the coNP lower bounds which hold already in the respective DL-Lite fragments without role names (i.e., the corresponding fragments of propositional logic).

Let $\mathcal{L}_{core}$ be the DL that admits only CIs $A \sqsubseteq B$ and $A \sqcap B \sqsubseteq \bot$ and let $\mathcal{L}_{horn}$ be the DL that admits only CIs $A \sqcap A' \sqsubseteq B$ and $A \sqcap B \sqsubseteq \bot$, where A, A', and B are concept names.

Theorem 17. *In $\mathcal{L}_{horn}$, $\mathcal{IQ}$-query emptiness, $\mathcal{CQ}$-query emptiness, and $\mathcal{CQ}$-predicate emptiness are coNP-hard. In $\mathcal{L}_{core}$, $\mathcal{CQ}$-query emptiness is coNP-hard.*

Proof. The proof for $\mathcal{L}_{horn}$ is by reduction from the SAT problem for propositional formulas in conjunctive normal form (CNF). Let $\varphi = \psi_0 \vee \cdots \vee \psi_k$ be a CNF formula, $v_0, \ldots, v_n$ the variables used in φ, $A_{\psi_1}, \ldots, A_{\psi_k}$ concept names for representing clauses, and $A_{v_1}, A_{\neg v_1}, \ldots, A_{v_n}, A_{\neg v_n}$ concept names for representing literals. We first define an $\mathcal{L}_{horn}$ TBox $\mathcal{T}$ as follows:

- $A_{v_j} \sqcap A_{\neg v_j} \sqsubseteq \bot$ for all $j \leq n$;
- $A_{\ell_j} \sqsubseteq A_{\psi_i}$ for all $i \leq k$ and each $\ell_j = (\neg)v_j$ that is a disjunct of ψ_i;
- $A_{\psi_1} \sqcap \cdots \sqcap A_{\psi_k} \sqsubseteq B$.

It is straighforward to show that $B(v)$ is empty for Σ given $\mathcal{T}$ iff $\exists v.B(v)$ is empty for Σ given $\mathcal{T}$ iff φ is unsatisfiable. For the $\mathcal{L}_{\mathsf{horn}}$ result, we drop the last CI from $\mathcal{T}$ and use the CQ $A_{\psi_1}(v) \wedge \cdots \wedge A_{\psi_k}(v)$. $\square$

We now prove matching upper complexity bounds, considering the logic DL-Lite$_{core}$ and leaving the straighforward extensions to more expressive DL-Lite dialects to the reader. DL-Lite$_{core}$ is the fragment of $\mathcal{ELI}_\bot$ with existential restrictions $\exists R.C$ (R of the form r or r^-) replaced with $\exists R$ and with conjunctions on the left-hand side of CIs allowed only if the right-hand side is $\bot$. Thus, a CI in DL-Lite$_{core}$ is of the form

$$B_1 \sqsubseteq B_2, \quad B_1 \sqcap B_2 \sqsubseteq \bot$$

where B_1 and B_2 are concepts of the form $\exists r$, $\exists r^-$, $\top$, $\bot$, or A (for $A \in \mathsf{N_C}$). As DL-Lite$_{core}$ is a fragment of $\mathcal{ELI}_\bot$, Lemma 8 holds, and so we can restrict our attention to positive ABoxes. However, we need a slightly modified version of the canonical model constructed in the proof of Lemma 9, presented in the following.

Let $\mathcal{T}$ be a DL-Lite$_{core}$-TBox and $\mathcal{A}$ a positive ABox. We construct the canonical model $\mathcal{U}_\mathcal{K}$ for $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ as follows. Take an x_R for every role R of the form r, r^- such that r occurs in $\mathcal{K}$. A *$\mathcal{K}$-path* is a finite sequence $ax_{R_1} \cdots x_{R_n}$, $n \geq 0$, such that a occurs in $\mathcal{A}$ and

- $\mathcal{T} \models \top \sqsubseteq \exists R_1$ or there exists $B(a) \in \mathcal{A}$ such that $\mathcal{T} \models B \sqsubseteq \exists R_1$ or there exists $r(a,b) \in \mathcal{A}$ such that $\mathcal{T} \models \exists r \sqsubseteq \exists R_1$ or there exists $r(b,a) \in \mathcal{A}$ such that $\mathcal{T} \models \exists r^- \sqsubseteq \exists R_1$;
- $\mathcal{T} \models \exists R_i^- \sqsubseteq \exists R_{i+1}$ for all $i < n$.

Now define $\mathcal{U}_\mathcal{K} = (\Delta^\mathcal{U}, \cdot^\mathcal{U})$ by taking as $\Delta^\mathcal{U}$ the set of all $\mathcal{K}$-paths and constructing $\cdot^{\mathcal{U}_\mathcal{K}}$ as follows:

- $a^{\mathcal{U}_\mathcal{K}} = a$ for all $a \in \mathsf{Ind}(\mathcal{A})$;
- for all $a \in \mathsf{Ind}(\mathcal{A})$ and concept names A, $a \in A^{\mathcal{U}_\mathcal{K}}$ iff $\mathcal{T} \models \top \sqsubseteq A$ or there exists $B(a) \in \mathcal{A}$ such that $\mathcal{T} \models B \sqsubseteq A$ or there exists $r(a,b) \in \mathcal{A}$ such that $\mathcal{T} \models \exists r \sqsubseteq A$ or there exists $r(b,a) \in \mathcal{A}$ such that $\mathcal{T} \models \exists r^- \sqsubseteq A$;
- for all $ax_{R_1} \cdots x_{R_n} \in \Delta^{\mathcal{U}_\mathcal{K}}$ such that $n \geq 1$ and all concept names A, $ax_{R_1} \cdots x_{R_n} \in A^{\mathcal{U}_\mathcal{K}}$ iff $\mathcal{T} \models \exists R_n^- \sqsubseteq A$;
- for all $d_1, d_2 \in \Delta^{\mathcal{U}_\mathcal{K}}$, if $d_1, d_1 \in \mathsf{Ind}(\mathcal{A})$, then $(d_1, d_2) \in r^{\mathcal{U}_\mathcal{K}}$ iff $r(d_1, d_2) \in \mathcal{A}$. Otherwise $(d_1, d_2) \in r^{\mathcal{U}_\mathcal{K}}$ iff $d_2 = d_1 \cdot x_r$ or $d_1 = d_2 \cdot x_{r^-}$ for some r.

It is not difficult to show the following:

Lemma 18. *Let $\mathcal{K}$ be consistent. For any k-ary conjunctive query q and $(a_1,\ldots,a_k) \in \mathsf{N}_\mathsf{I}^k$, $\mathcal{U}_\mathcal{K} \models q[a_1,\ldots,a_k]$ iff $(a_1,\ldots,a_k) \in \mathsf{cert}_\mathcal{K}(q)$.*

We are now ready to establish the announced PTIME result.

Theorem 19. *In DL-Lite$_{core}$, $\mathcal{IQ}$-query emptiness and $\mathcal{CQ}$-predicate emptiness can be decided in* PTIME.

Proof. We first consider $\mathcal{IQ}$-query emptiness. Let $\mathcal{T}$ be a DL-Lite$_{core}$-TBox and Σ a signature.

Claim. For all concept names A, $A(v)$ is not empty for Σ given $\mathcal{T}$ if, and only if, $(\mathcal{T},\mathcal{A}) \models A(a)$ for some ABox $\mathcal{A}$ from the list

- $\{\top(a)\}$,
- $\{B(a)\}$, $B \in \Sigma$,
- $\{r(a,b)\}$, $r \in \Sigma$,
- $\{r(b,a)\}$, $r \in \Sigma$,

such that $(\mathcal{T},\mathcal{A})$ is consistent.

Since the instance problem '$(\mathcal{T},\mathcal{A}) \models A(a)$' can be solved in polynomial time in DL-Lite$_{core}$ (Calvanese et al. 2007), it follows immediately from this claim that $\mathcal{IQ}$-query emptiness can be decided in PTIME.

The "if" direction of the above claim is trivial. For the "only if" direction, assume that $A(v)$ is not empty for Σ given $\mathcal{T}$. By Lemma 8, there is a positive Σ-ABox $\mathcal{A}$ such that $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ and $(\mathcal{T},\mathcal{A}) \models A(a_0)$ for some $a_0 \in \mathsf{Ind}(\mathcal{A})$. By Lemma 18, this implies $\mathcal{U}_{\mathcal{T},\mathcal{A}} \models A[a_0]$. By inspecting the construction of $\mathcal{U}_{\mathcal{T},\mathcal{A}}$, it is readily checked that this implies that one of the following conditions holds:

- $(\mathcal{T},\{\top(a_0)\}) \models A(a_0)$;
- there exists $B(a_0) \in \mathcal{A}$ with $(\mathcal{T},\{B(a_0)\}) \models A(a_0)$;
- there exists $r(a_0,b) \in \mathcal{A}$ with $(\mathcal{T},\{r(a_0,b)\}) \models A(a_0)$;
- there exists $r(b,a_0) \in \mathcal{A}$ with $(\mathcal{T},\{r(b,a_0)\}) \models A(a_0)$.

This observation proves the "only if" direction.

The polynomial time algorithm for $\mathcal{CQ}$-predicate emptiness is similar. In this case, one can easily show using Lemma 18 that $\exists v.A(v)$ is not $\mathcal{CQ}$-empty for Σ given $\mathcal{T}$ if, and only if, $(\mathcal{T},\mathcal{A}) \models \exists v.A(v)$ for some ABox $\mathcal{A}$ from the list $\{\top(a)\}$, $\{B(a)\}$, $B \in \Sigma$, $\{r(a,b)\}$, $r \in \Sigma$, $\{r(b,a)\}$, $r \in \Sigma$, such that $(\mathcal{T},\mathcal{A})$ is satisfiable. The same characterization holds for queries of the form $\exists v,v'.r(v,v')$. Since answering Boolean conjunctive queries in DL-Lite$_{core}$ is in PTIME, it follows that $\mathcal{CQ}$-predicate emptiness can be decided in PTIME. □

Theorem 20. *In DL-Lite$_{core}$, $\mathcal{CQ}$-query emptiness can be decided in coNP.*

Proof (idea). One can show that if a CQ q is non-empty for Σ given a DL-Lite$_{core}$ TBox $\mathcal{T}$, then there exists a witness Σ-ABox of size bounded by the size of Σ times the length of q. An NP-algorithm checking non-emptiness is obtained by guessing such a Σ-ABox together with a match of q in some appropriately defined minimal model $\mathcal{I}$ of $(\mathcal{T},\mathcal{A})$, and then checking in polynomial time that the match and the model are as required. □

The proofs of Theorems 19 and 20 are easily extended to DLs such as DL-Lite$_\mathcal{F}$, DL-Lite$_\mathcal{R}$, and DL-Lite$_{horn}$.

Expressive DLs

We consider the $\mathcal{ALC}$ family of expressive DLs, establishing decidability results for $\mathcal{ALC}$ and $\mathcal{ALCI}$, and undecidability results for $\mathcal{ALCF}$. We start with the former.

Our aim is to show that $\mathcal{IQ}$-query emptiness and $\mathcal{CQ}$-predicate emptiness in $\mathcal{ALCI}$ are decidable in NEXPTIME. As in the $\mathcal{EL}$ case, we first show that it is possible to concentrate on a single ABox $\mathcal{A}_\Sigma$ instead of considering all Σ-ABoxes. This ABox is defined as follows.

Definition 21. Let $\mathcal{T}$ be an $\mathcal{ALCI}$-TBox and Σ a signature. The *closure* $\mathsf{cl}(\mathcal{T},\Sigma)$ is the smallest set that contains $\Sigma \cap \mathsf{N}_\mathsf{C}$ as well as all concepts that occur (potentially as a subconcept) in $\mathcal{T}$ and is closed under single negations. A *type* for $\mathcal{T}$ and Σ is a set $t \subseteq \mathsf{cl}(\mathcal{T},\Sigma)$ such that for some model $\mathcal{I}$ of $\mathcal{T}$ and some $d \in \Delta^\mathcal{I}$, we have $t = \{C \in \mathsf{cl}(\mathcal{T},\Sigma) \mid d \in C^\mathcal{I}\}$. Let $\mathfrak{T}_{\mathcal{T},\Sigma}$ denote the set of all types for $\mathcal{T}$ and Σ. We use $\mathcal{I}_{\mathcal{T},\Sigma}$ to denote the *canonical* Σ*-model of* $\mathcal{T}$, defined as:

$$\begin{array}{rcl}\Delta^{\mathcal{I}_{\mathcal{T},\Sigma}} & = & \mathfrak{T}_{\mathcal{T},\Sigma} \\ A^{\mathcal{I}_{\mathcal{T},\Sigma}} & = & \{t \in \mathfrak{T}_{\mathcal{T},\Sigma} \mid A \in t\} \\ r^{\mathcal{I}_{\mathcal{T},\Sigma}} & = & \{(t,t') \in \mathfrak{T}_{\mathcal{T},\Sigma} \times \mathfrak{T}_{\mathcal{T},\Sigma} \mid \\ & & \text{for all } \exists r.C \in \mathsf{cl}(\mathcal{T},\Sigma) : C \in t' \Rightarrow \exists r.C \in t\}\end{array}$$

The canonical Σ-ABox $\mathcal{A}_{\mathcal{T},\Sigma}$ for $\mathcal{T}$ is defined as follows:

$$\begin{array}{rcl}\mathcal{A}_{\mathcal{T},\Sigma} & = & \{A(a_t) \mid t \in A^{\mathcal{I}_{\mathcal{T},\Sigma}} \wedge A \in \Sigma\} \cup \\ & & \{\neg A(a_t) \mid t \notin A^{\mathcal{I}_{\mathcal{T},\Sigma}} \wedge A \in \Sigma\} \cup \\ & & \{r(a_t,a_{t'}) \mid (t,t') \in r^{\mathcal{I}_{\mathcal{T},\Sigma}} \wedge r \in \Sigma\}\end{array}$$

It is easy to see that the cardinality of $\mathfrak{T}_{\mathcal{T},\Sigma}$ is at most exponential in the size of $\mathcal{T}$ and the cardinality of Σ, and that the set $\mathfrak{T}_{\mathcal{T},\Sigma}$ can be computed in exponential time by making use of well-known EXPTIME procedures for concept satisfiability w.r.t. TBoxes in $\mathcal{ALCI}$. Thus, $\mathcal{A}_{\mathcal{T},\Sigma}$ is of exponential size and can be computed in exponential time.

The following theorem provides the basis for our decision procedure.

Theorem 22. *A conjunctive query q is empty for Σ given $\mathcal{T}$ iff $\mathsf{cert}_{\mathcal{T},\mathcal{A}_{\mathcal{T},\Sigma}}(q) = \emptyset$.*

To prove Theorem 22, we start by establishing a series of helpful lemmas.

Lemma 23. *$\mathcal{I}_{\mathcal{T},\Sigma}$ is a model of $\mathcal{T}$ and $\mathcal{A}_{\mathcal{T},\Sigma}$.*

Proof. It is straightforward to prove by induction on the structure of C that for all $C \in \mathsf{cl}(\mathcal{T},\Sigma)$, we have $C \in t$ iff $t \in C^{\mathcal{I}_{\mathcal{T},\Sigma}}$. By definition of types, $C \sqsubseteq D \in \mathcal{T}$ and $C \in t$ implies $D \in t$. Thus, $\mathcal{I}_{\mathcal{T},\Sigma}$ is clearly a model of $\mathcal{T}$. It is an immediate consequence of the definition of $\mathcal{A}_{\mathcal{T},\Sigma}$ that $\mathcal{I}_{\mathcal{T},\Sigma}$ is also a model of $\mathcal{A}_{\mathcal{T},\Sigma}$. □

Definition 24. Let $\mathcal{A}$ and $\mathcal{A}'$ be literal ABoxes. An *ABox homomorphism* from $\mathcal{A}$ to $\mathcal{A}'$ is a total map $h : \mathsf{Ind}(\mathcal{A}) \to \mathsf{Ind}(\mathcal{A}')$ such that the following conditions are satisfied:

- $A(a) \in \mathcal{A}$ implies $A(h(a)) \in \mathcal{A}'$;
- $\neg A(a) \in \mathcal{A}$ implies $\neg A(h(a)) \in \mathcal{A}'$;
- $r(a, b) \in \mathcal{A}$ implies $r(h(a), h(b)) \in \mathcal{A}'$.

Lemma 25. *If $\mathcal{T}$ is an $\mathcal{ALCI}$-TBox, q a CQ, $\mathcal{T}, \mathcal{A} \models q[a_1, \ldots, a_n]$, and h is an ABox homomorphism from $\mathcal{A}$ to $\mathcal{A}'$, then $\mathcal{T}, \mathcal{A}' \models q[a_1, \ldots, a_n]$.*

Proof. We prove the contrapositive. Thus assume that $\mathcal{T}, \mathcal{A}' \not\models q[a_1, \ldots, a_n]$. Then there is a model $\mathcal{I}'$ of $\mathcal{T}$ and $\mathcal{A}'$ such that $\mathcal{I}' \not\models q[a_1, \ldots, a_n]$. Define a model $\mathcal{I}$ by starting with $\mathcal{I}'$ and reinterpreting the individual names in $\mathsf{Ind}(\mathcal{A})$ by setting $a^{\mathcal{I}} = h(a)^{\mathcal{I}'}$ for each $a \in \mathsf{Ind}(\mathcal{A})$. Since individual names do not occur in $\mathcal{T}$, $\mathcal{I}$ is clearly a model of $\mathcal{T}$. It is also a model of $\mathcal{A}$: if $A(a) \in \mathcal{A}$, then $A(h(a)) \in \mathcal{A}'$ by definition of ABox homomorphisms. Since $\mathcal{I}'$ is a model of $\mathcal{A}'$ and by definition of $\mathcal{I}$, it follows that $a^{\mathcal{I}} \in A^{\mathcal{I}}$. The cases $\neg A(a) \in \mathcal{A}$ and $r(a, b) \in \mathcal{A}$ are analogous. Finally, $\mathcal{I}' \not\models q[a_1, \ldots, a_n]$ and the definition of $\mathcal{I}$ yield $\mathcal{I} \not\models q[a_1, \ldots, a_n]$. We have thus shown that $\mathcal{T}, \mathcal{A} \not\models q[a_1, \ldots, a_n]$. □

Lemma 26. *Let $\mathcal{T}$ be an $\mathcal{ALCI}$-TBox and $\mathcal{A}$ a literal Σ-ABox that is consistent w.r.t. $\mathcal{T}$. Then there is an ABox homomorphism from $\mathcal{A}$ to $\mathcal{A}_{\mathcal{T},\Sigma}$.*

Proof. Let $\mathcal{I}$ be a model of $\mathcal{T}$ and $\mathcal{A}$ and for each $d \in \Delta^{\mathcal{I}}$, define $t_d^{\mathcal{I}} = \{C \in \mathsf{cl}(\mathcal{T}, \Sigma) \mid d \in C^{\mathcal{I}}\}$. Define h by setting $h(a) = a_t$ with $t = t_{a^{\mathcal{I}}}^{\mathcal{I}}$ for all $a \in \mathsf{Ind}(\mathcal{A})$. Using the definition of $\mathcal{A}_{\mathcal{T},\mathcal{A}}$, it is easy to see that h is indeed an ABox homomorphism. □

We are now ready to prove Theorem 22.

Proof of Theorem 22. The "only if" direction is trivial. For the "if" direction, let $\mathsf{cert}_{\mathcal{T},\mathcal{A}_{\mathcal{T},\Sigma}}(q) = \emptyset$. To show that q is empty for Σ given $\mathcal{T}$, take a Σ-ABox $\mathcal{A}$ that is consistent with $\mathcal{T}$. By Lemmas 25 and 26, $\mathsf{cert}_{\mathcal{T},\mathcal{A}_{\mathcal{T},\Sigma}}(q) = \emptyset$ implies $\mathsf{cert}_{\mathcal{T},\mathcal{A}}(q) = \emptyset$ as required. □

Theorem 22 is the key to a NExpTime algorithm for $\mathcal{IQ}$-query emptiness. This refutes our own conjecture of NExpTime$^{\text{NP}}$ -hardness from the workshop paper (Baader et al. 2009) (unless NExpTime = NExpTime$^{\text{NP}}$).

Theorem 27. *In $\mathcal{ALCI}$, $\mathcal{IQ}$-query emptiness and $\mathcal{CQ}$-predicate emptiness are in* NExpTime.

Proof. By Lemma 6, it suffices to consider $\mathcal{IQ}$-query emptiness. Thus, let $\mathcal{T}$ be an ABox, Σ a signature, and $A(v)$ an IQ for which emptiness for Σ given $\mathcal{T}$ is to be decided. The algorithm first computes the canonical ABox $\mathcal{A}_{\mathcal{T},\Sigma}$ (in exponential time) and then verifies in the following way that for each $a \in \mathsf{Ind}(\mathcal{A}_{\mathcal{T},\Sigma})$, we have $\mathcal{T}, \mathcal{A}_{\mathcal{T},\Sigma} \not\models A(a)$: guess a map $\pi : \mathsf{Ind}(\mathcal{A}_{\mathcal{T},\Sigma}) \to \mathfrak{T}_{\mathcal{T},\Sigma}$ with $\neg A \in \pi(a)$ and such that (i) $B(c) \in \mathcal{A}_{\mathcal{T},\Sigma}$ implies $B \in \pi(c)$, and (ii) $r(b, c) \in \mathcal{A}_{\mathcal{T},\Sigma}$, $C \in \pi(c)$, and $\exists r.C \in \mathsf{cl}(\mathcal{T}, \Sigma)$ implies $\exists r.C \in \pi(b)$; then check for each $b \in \mathsf{Ind}(\mathcal{A}_{\mathcal{T},\Sigma})$ that $\bigsqcap_{C \in \pi(b)} C$ is satisfiable w.r.t. $\mathcal{T}$ (this takes at most single-exponential time in $|\mathcal{T}|$ and $|\Sigma|$). The non-deterministic algorithm accepts if all satisfiability checks succeed (for each $a \in \mathsf{Ind}(\mathcal{A}_{\mathcal{T},\Sigma})$), and rejects otherwise.

By Theorem 22, it suffices to show that

Claim. The algorithm returns "yes" iff $\mathcal{A}_{\mathcal{T},\Sigma} \not\models A(a)$ for all $a \in \mathsf{Ind}(\mathcal{A}_{\mathcal{T},\Sigma})$.

Proof of claim. For the first direction, suppose the algorithm returns "yes". Then for each $a \in \mathsf{Ind}(\mathcal{A}_{\mathcal{T},\Sigma})$, there is a mapping $\pi : \mathsf{Ind}(\mathcal{A}_{\mathcal{T},\Sigma}) \to \mathfrak{T}_{\mathcal{T},\Sigma}$ with $\neg A \in \pi(a)$ which satisfies conditions (i) and (ii) above and is such that $\bigsqcap_{C \in \pi(b)} C$ is satisfiable w.r.t. $\mathcal{T}$ for every $b \in \mathsf{Ind}(\mathcal{A}_{\mathcal{T},\Sigma})$. But that means we can find a model $\mathcal{I}_a$ of $\mathcal{T}$ and $\mathcal{A}_{\mathcal{T},\Sigma}$ such that $a \notin A^{\mathcal{I}_a}$. In other words, for all $a \in \mathsf{Ind}(\mathcal{A}_{\mathcal{T},\Sigma})$, we have $\mathcal{T}, \mathcal{A}_{\mathcal{T},\Sigma} \not\models A(a)$.

For the second direction, suppose $\mathcal{A}_{\mathcal{T},\Sigma} \not\models A(a)$ for all $a \in \mathsf{Ind}(\mathcal{A}_{\mathcal{T},\Sigma})$. Then for each $a \in \mathsf{Ind}(\mathcal{A}_{\mathcal{T},\Sigma})$, we can find some model $\mathcal{I}_a$ of $\mathcal{T}$ and $\mathcal{A}_{\mathcal{T},\Sigma}$ such that $a \notin A^{\mathcal{I}_a}$. So when it is time to check $a \in \mathsf{Ind}(\mathcal{A}_{\mathcal{T},\Sigma})$, we guess the mapping π such that $\pi(b)$ is the type of $b^{\mathcal{I}_a}$ in the model $\mathcal{I}_a$, i.e. $\pi(b) = \{C \mid b \in C^{\mathcal{I}_a} \text{ and } C \in \mathsf{cl}(\mathcal{T}, \Sigma)\}$. By construction, π will satisfy the required conditions, and each concept $\bigsqcap_{C \in \pi(b)} C$ will be satisfiable w.r.t. $\mathcal{T}$. So all of the satisfiability tests will succeed, which means the algorithm returns "yes". □

The best known lower bound for the problems considered in Theorem 27 is ExpTime. It stems from an easy reduction of satisfiability in $\mathcal{ALC}$: a concept C is satisfiable w.r.t. $\mathcal{T}$ iff A is $\mathcal{CQ}$-predicate empty for $\Sigma = \emptyset$ and $\mathcal{T} = \{\neg C \sqsubseteq A\}$. For $\mathcal{CQ}$-query emptiness, we can easily derive the following results.

Theorem 28. *In $\mathcal{ALC}$ and $\mathcal{ALCI}$, $\mathcal{CQ}$-query emptiness is in 2-*ExpTime*. In $\mathcal{ALCI}$, it is 2-*ExpTime*-complete.*

Proof. The upper bound for $\mathcal{CQ}$-query emptiness in $\mathcal{ALCI}$ (and thus $\mathcal{ALC}$) is obtained by simply computing the canonical ABox $\mathcal{A}_{\mathcal{T},\Sigma}$ and $\mathsf{cert}_{\mathcal{T},\mathcal{A}_{\mathcal{T},\Sigma}}(q)$, and then checking whether the latter is empty. This can be done in 2-ExpTime since it is shown in (Calvanese, De Giacomo, and Lenzerini 1998) that for all $\mathcal{T}$, $\mathcal{A}$, and q, the set $\mathsf{cert}_{\mathcal{T},\mathcal{A}}(q)$ can be computed in time $2^{p(m) \cdot 2^{p(n)}}$ with p a polynomial, m the size of $\mathcal{T} \cup \mathcal{A}$, and n the size of q. The lower bound stems from the facts that (i) CQ entailment in $\mathcal{ALCI}$ is 2-ExpTime-hard already for empty ABoxes (Lutz 2008) and (ii) a Boolean CQ q is entailed by $\mathcal{T}$ and the empty ABox iff q is non-empty for $\Sigma = \emptyset$ and $\mathcal{T}$. □

The best known lower bound for $\mathcal{CQ}$-query emptiness in $\mathcal{ALC}$ is ExpTime. We conjecture that the upper bound can be improved from 2-ExpTime to NExpTime by adapting the proof of Theorem 27 to the more intricate case of CQs.

We now show that the simple addition of functional roles to $\mathcal{ALC}$ leads to undecidability of $\mathcal{CQ}$-predicate emptiness, thus also of $\mathcal{IQ}$-query emptiness and $\mathcal{CQ}$-query emptiness. The proof is by reduction from a tiling problem that asks for a tiling of a rectangle of finite size (but the size is neither fixed nor bounded). The reduction involves a variety of technical tricks such as the treatment of concept names that are not in Σ as universally quantified second-order variables, which allows one to enforce a grid structure by standard frame axioms from modal logic. In a similar way, inverse roles are simulated by normal role names. When conjunctive queries are considered and not instance queries, the correctness proof of the reduction is surprisingly subtle and easily the most intricate proof in this paper. Of course, undecidability carries over to variants of $\mathcal{ALCF}$ that use a concept constuctor $(\leq 1\, r)$ instead of functional roles as an additional sort, and to all DLs with qualified or unqualified number restrictions.

Theorem 29. *In $\mathcal{ALCF}$, $\mathcal{CQ}$-query emptiness, $\mathcal{IQ}$-query emptiness, and $\mathcal{IQ}$-predicate emptiness are undecidable.*

An instance of the aforementioned tiling problem is given by a triple $(\mathfrak{T}, H, V)$ with $\mathfrak{T}$ a non-empty, finite set of *tile types* including an *initial tile* T_{init} to be placed on the lower left corner and a *final tile* T_{final} to be placed on the upper right corner, $H \subseteq \mathfrak{T} \times \mathfrak{T}$ a *horizontal matching relation*, and $V \subseteq \mathfrak{T} \times \mathfrak{T}$ a *vertical matching relation*. A *tiling* for $(\mathfrak{T}, H, V)$ is a map $f : \{0, \dots, n\} \times \{0, \dots, m\} \to \mathfrak{T}$ such that $n, m \geq 0$, $f(0,0) = T_{\mathsf{init}}$, $f(n,m) = T_{\mathsf{final}}$, $(f(i,j), f(i+1,j)) \in H$ for all $i < n$, and $(f(i,j), f(i,j+1)) \in v$ for all $i < m$. It is undecidable whether an instance of the tiling problem has a tiling.

For the reduction, let $(\mathfrak{T}, H, V)$ be an instance of the tiling problem with $\mathfrak{T} = \{T_1, \dots, T_p\}$. We construct a signature Σ and a TBox $\mathcal{T}$ such that $(\mathfrak{T}, H, V)$ has a solution if and only if a selected concept name A is $\mathcal{CQ}$-predicate non-empty for Σ given $\mathcal{T}$.

The ABox signature is $\Sigma = \{T_1, \dots, T_p, x, y, x^-, y^-\}$ where $T_1, \dots, T_p$ are used as concept names, and x, y, x^-, and y^- are *functional* role names. We use the role names x and y to represent horizontal and vertical adjacency of points in the rectangle, and the role names x^- and y^- to simulate the inverses of x and y. In $\mathcal{T}$, we use the concept names U, R, A, Y, I_x, I_y, C, where U and R mark the upper and right border of the rectangle, A is the concept name used in the conjunctive query, and Y, I_x, I_y, and C are used for technical purposes explained below. We also require 6 auxiliary concept names $Z_{c,1}$, $Z_{c,2}$, $Z_{x,1}$, $Z_{x,2}$, $Z_{y,1}$, and $Z_{y,2}$. In the following, for $e \in \{c, x, y\}$, we let $\mathcal{B}_e$ range over all Boolean combinations of the concept names $Z_{e,1}$ and $Z_{e,2}$, i.e., over all concepts $L_1 \sqcap L_2$ where L_i is a literal over $Z_{e,i}$, for $i \in \{1, 2\}$.

The TBox $\mathcal{T}$ is defined as the union of the following CIs, for all $(T_i, T_j) \in H$ and $(T_i, T_\ell) \in V$:

$$\begin{array}{rcl}
T_{\mathsf{final}} & \sqsubseteq & Y \sqcap U \sqcap R \\
\exists x.(U \sqcap Y \sqcap T_j) \sqcap I_x \sqcap T_i & \sqsubseteq & U \sqcap Y \\
\exists y.(R \sqcap Y \sqcap T_\ell) \sqcap I_y \sqcap T_i & \sqsubseteq & R \sqcap Y \\
\exists x.(T_j \sqcap Y \sqcap \exists y.Y) & & \\
\sqcap \exists y.(T_\ell \sqcap Y \sqcap \exists x.Y) & & \\
\sqcap I_x \sqcap I_y \sqcap C \sqcap T_i & \sqsubseteq & Y \\
Y \sqcap T_{\mathsf{init}} & \sqsubseteq & A \\
\mathcal{B}_x \sqcap \exists x.\exists x^-.\mathcal{B}_x & \sqsubseteq & I_x \\
\mathcal{B}_y \sqcap \exists y.\exists y^-.\mathcal{B}_y & \sqsubseteq & I_y \\
\exists x.\exists y.\mathcal{B}_c \sqcap \exists y.\exists x.\mathcal{B}_c & \sqsubseteq & C \\
U & \sqsubseteq & \forall y.\bot \\
R & \sqsubseteq & \forall x.\bot \\
U & \sqsubseteq & \forall x.U \\
R & \sqsubseteq & \forall y.R \\
\bigsqcup_{1 \leq s < t \leq p} T_s \sqcap T_t & \sqsubseteq & \bot
\end{array}$$

Observe that the concept name A used in the conjunctive query occurs only once in the TBox, on the right-hand side of a CI. Taken together, the upper part of $\mathcal{T}$ ensures the existence of a tiled $n \times m$-rectangle in a witness ABox. The concept name Y is entailed at every individual name in such an ABox that is part of the rectangle. Observe that the CIs for Y enforce the horizontal and vertical matching conditions. The CI for C enforces confluence, i.e., C is entailed at an individual name a if there is an individual b that is both an x-y-successor and a y-x-successor of a. This is so because, intuitively, $\mathcal{B}_c$ is universally quantified: if confluence fails, we can interpret $Z_{c,1}$ and $Z_{c,2}$ in a way such that neither of the two conjuncts in the precondition of the CI for C is satisfied. In a similar manner, the CI for I_x (resp. I_y) is used to ensure that x^- (resp. y^-) acts as the inverse of x (resp. y) at all points in the rectangle, which means that x (resp. y) is inverse functional within the rectangle.

To establish Theorem 29, it suffices to prove the following lemma (see the appendix for details).

Lemma 30. *$(\mathfrak{T}, H, V)$ admits a tiling iff there is a Σ-ABox $\mathcal{A}$ that is consistent with $\mathcal{T}$ and such that $\mathcal{T}, \mathcal{A} \models \exists v.A(v)$.*

Σ-substitutes of a TBox

Apart from being a fundamental reasoning service in ontology-based data access, predicate emptiness can also be used to extract a module from a TBox to speed up query answering. The idea is to exploit the information about empty predicates for Σ given $\mathcal{T}$ to compute a subset $\mathcal{T}'$ of $\mathcal{T}$ that can be used instead of $\mathcal{T}$ to query Σ-ABoxes without affecting the certain answers. If $\mathcal{T}'$ is significantly smaller than $\mathcal{T}$, then using $\mathcal{T}'$ instead of $\mathcal{T}$ to answer queries over Σ-ABoxes should significantly speed up querying processing. This idea is formalized by Σ-substitutes.

Definition 31. Let $\mathcal{T}' \subseteq \mathcal{T}$ and $\mathcal{L} \in \{\mathcal{IQ}, \mathcal{CQ}\}$. Then $\mathcal{T}'$ is a *Σ-substitute for $\mathcal{T}$ w.r.t. $\mathcal{L}$* if for all Σ-ABoxes $\mathcal{A}$ and all $q \in \mathcal{L}$, we have that $\mathsf{cert}_{\mathcal{T}',\mathcal{A}}(q) = \mathsf{cert}_{\mathcal{T},\mathcal{A}}(q)$.

Modules and module extraction have been studied extensively in recent years (Stuckenschmidt, Parent, and Spaccapietra 2009), and we briefly discuss the relationship between Σ-substitutes and existing notions of a module from the literature. Most logic-based approaches to module extraction demand that a Σ-module $\mathcal{M}$ of a TBox $\mathcal{T}$ is a subset of $\mathcal{T}$ that gives the same answers to queries that use *only symbols from* Σ (Grau et al. 2008; Konev et al. 2009; 2008; Kontchakov, Wolter, and Zakharyaschev 2010). Thus, the main conceptual difference between Σ-substitutes and Σ-modules from the literature is that Σ-substitutes give the same answers to *all queries regardless of their signature*, and so the restriction to Σ-symbols applies to the ABox only. It follows that minimal Σ-modules as defined in (Konev et al. 2008; Kontchakov, Wolter, and Zakharyaschev 2010) cannot in general be used as Σ-substitutes.

As it is beyond the scope of this paper to investigate Σ-substitutes in depth, we confine ourselves to a couple of important observations. Firstly, in $\mathcal{ELI}$ (and, therefore, various $\mathcal{EL}$ and DL-Lite dialects) one can use $\mathcal{CQ}$-emptiness in a straighforward way to compute a Σ-substitute w.r.t. $\mathcal{CQ}$. For a TBox $\mathcal{T}$ in $\mathcal{ELI}$ and a signature Σ, we denote by $\mathcal{T}_\Sigma^{\mathcal{CQ}}$ the set of all concept inclusions $\alpha \in \mathcal{T}$ such that no $X \in \mathsf{sig}(\alpha)$ is $\mathcal{CQ}$-empty for Σ given $\mathcal{T}$.

Theorem 32. *In $\mathcal{ELI}$, $\mathcal{T}_\Sigma^{\mathcal{CQ}}$ is a Σ-substitute for $\mathcal{T}$ w.r.t. $\mathcal{CQ}$ (and thus also w.r.t. $\mathcal{IQ}$).*

Note that by Theorem 16, $\mathcal{T}_\Sigma^{\mathcal{CQ}}$ can be computed in polynomial time if $\mathcal{T}$ is an $\mathcal{EL}$-TBox. It can be seen that a subset $\mathcal{T}_\Sigma^{\mathcal{IQ}}$ defined in analogy to $\mathcal{T}_\Sigma^{\mathcal{CQ}}$ but based on $\mathcal{IQ}$-emptiness instead of $\mathcal{CQ}$-emptiness cannot serve as a Σ-substitute w.r.t. $\mathcal{IQ}$ even when $\mathcal{T}$ is formulated in $\mathcal{EL}$ or DL-Lite$_{\mathsf{core}}$.

Currently, no designated algorithms for computing Σ-substitutes in more expressive DLs are available. Interestingly, however, and in contrast to the Σ-modules discussed above, semantic and syntactic $\bot$-modules as introduced in (Grau et al. 2008) turn out to be examples of Σ-substitutes. To define $\bot$-modules, let Σ be a signature. Two interpretations $\mathcal{I}$ and $\mathcal{I}'$ *coincide* w.r.t. Σ if $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}'}$ and $X^{\mathcal{I}} = X^{\mathcal{I}'}$ for all $X \in \Sigma$. A subset $\mathcal{T}'$ of a TBox $\mathcal{T}$ is called a *semantic $\bot$-module of $\mathcal{T}$ w.r.t. Σ* if for every interpretation $\mathcal{I}$ the interpretation $\mathcal{I}'$ that coincides with $\mathcal{I}$ w.r.t. $\Sigma \cup \mathsf{sig}(\mathcal{T}')$ and in which $X^{\mathcal{I}} = \emptyset$ for all $X \notin \Sigma \cup \mathsf{sig}(\mathcal{T}')$ is a model of $\mathcal{T} \setminus \mathcal{T}'$. In (Grau et al. 2008), it is shown that extracting a minimal semantic $\bot$-module is of the same complexity as standard reasoning. In addition, it is shown that a syntactic approximation called the *syntactic $\bot$-module* can be computed in polynomial time. The following lemma establishes the relationship between $\bot$-modules and Σ-substitutes.

Lemma 33. *Let $\mathcal{T}$ be a TBox in any of the DLs introduced in this paper and let $\mathcal{T}'$ be a semantic $\bot$-module of $\mathcal{T}$ w.r.t. Σ. Then $\Sigma \cup \mathsf{sig}(\mathcal{T}')$ contains all predicates that are not $\mathcal{CQ}$-empty for Σ given $\mathcal{T}$ and $\mathcal{T}'$ is a Σ-substitute of $\mathcal{T}$ w.r.t. $\mathcal{CQ}$.*

Thus, one can use the algorithms from (Grau et al. 2008) for computing semantic or syntactic $\bot$-modules in a large variety of DLs to obtain Σ-substitutes. In general, however, $\bot$-modules can be much larger than a minimal Σ-substitute. The following example shows that this can be the case already for acyclic $\mathcal{EL}$-TBoxes and for the Σ-substitutes considered in Theorem 32. Further, empirical evidence is provided in the subsequent section.

Example 34. Let $\mathcal{T} = \{A \sqsubseteq \exists s_1.\exists r_1.\top \sqcap \exists s_2.\exists r_2.\top, B \equiv \exists r_1.\top \sqcap \exists r_2.\top\}$ and $\Sigma = \{A\}$. The predicates that are not $\mathcal{CQ}$-empty for Σ given $\mathcal{T}$ and $\mathcal{T}_\Sigma^{\mathcal{CQ}}$ are A, s_1, s_2, r_1, r_2 and $\mathcal{T}_\Sigma^{\mathcal{CQ}}$ comprises only the first CI of $\mathcal{T}$. However, $\mathcal{T}$ has no non-trivial $\bot$-modules w.r.t. Σ.

Case Study

The aim of this section is to evaluate predicate emptiness and Σ-substitutes in a real-world application, demonstrating the usefulness of these notions. Our application is from the medical domain: ABoxes are used to store clinical patient data using a suitable signature Σ that stems from real-world medical records, and the well-known ontology SNOMED CT is used to provide additional vocabulary. To show that our results are not specific to the chosen signature and since additional signatures from real-world applications are difficult to obtain, we also consider a number of randomly generated signatures.

concepts	roles	$\mathcal{IQ}$ non-empty	$\mathcal{CQ}$ non-empty	axioms $\bot$-mod.	axioms $\mathcal{CQ}$-subst.
500	16	3557	4631	8910	4597
500	31	3654	4734	8911	4696
1000	16	5827	7385	14110	7349
1000	31	6242	7762	14147	7731
5000	16	18330	21451	33469	21427
5000	31	18469	21557	33616	21532
10000	16	29519	33493	47044	33489
10000	31	30643	34645	47256	34637

Figure 2: Experimental Results

We have carried out two kinds of experiment. The first one aims at understanding how many additional predicates for query formulation are provided by SNOMED CT. We consider $\mathcal{IQ}$-query emptiness and $\mathcal{CQ}$-predicate emptiness, counting the number of symbols that are not in the input signature Σ but still non-empty for Σ given $\mathcal{T}$. In the second experiment, we analyze the size of Σ-substitutes $\mathcal{T}_\Sigma^{\mathcal{CQ}}$ of Theorem 32 and compare it to the size of the original ontology and to $\bot$-modules, which can be used as an alternative to Σ-substitutes as discussed in the previous section.

The real-world signature was obtained by analyzing clinical notes of the emergency department and the intensive care unit of two Australian hospitals, using natural language processing methods to detect SNOMED CT concepts and roles[2]. SNOMED CT contains about 370,000 concepts and 62 roles, but in the analyzed clinical notes only 8,858 concepts and 16 roles were detected. For this signature Σ, 16,212 $\mathcal{IQ}$-non-empty predicates and 17,339 $\mathcal{CQ}$-non-empty predicates were computed. Thus, SNOMED CT provides a substantial number of additional predicates for query formulation, roughly identical to the number of predicates in the ABox signature. These numbers also show that the majority of predicates in SNOMED CT cannot meaningfully be used in queries over Σ-ABoxes, and thus identifying the relevant ones via predicate emptiness is rather helpful. Somewhat surprisingly, the number of $\mathcal{CQ}$-non-empty predicates is only about 10% higher than the number of $\mathcal{IQ}$-non-empty symbols.

We also computed the Σ-substitute w.r.t. $\mathcal{CQ}$ of Theorem 32, which contains 17,322 axioms. Thus, the Σ-substitute is of about 5% the size of the original ontology and can be expected to significantly speed up query processing when used instead of the whole SNOMED CT. The $\bot$-module w.r.t. Σ turns out to be significantly larger than the

[2]See “Current Projects” at http://www.it.usyd.edu.au/∼hitru.

	$\mathcal{IQ}$-query	$\mathcal{CQ}$-predicate	$\mathcal{CQ}$-query
$\mathcal{EL}$	in PTime	in PTime	in PTime
$\mathcal{EL}_\bot$	ExpTime-c.	ExpTime-c.	in 2-ExpTime ExpTime-h.
$\mathcal{ELI}$	ExpTime-c.	ExpTime-c.	
DL-Lite$_{core}$	in PTime	in PTime	coNP-c.
DL-Lite$_{horn}$	coNP-c.	coNP-c.	coNP-c.
$\mathcal{ALC}$	in NExpTime ExpTime-h.	in NExpTime ExpTime-h.	in 2-Exp, Exp-h.
$\mathcal{ALCI}$			2-ExpTime-c.
$\mathcal{ALCF}$	undec.	undec.	undec.

Figure 3: Complexity Results

computed Σ-substitute: it contains 27,383 axioms.

We have additionally analyzed randomly generated signatures that contain 500, 1,000, 5,000, and 10,000 concept names and 16 or 31 role names. Every signature contained the special role name role-group, which is used in SNOMED CT to implement a certain modeling pattern and should be present also in ABoxes to allow the same pattern there. For each number of concept and role names, we generated 10 signatures. Figure 2 shows the results, where the numbers are averages for the 10 experiments for each size. These additional experiments confirm the findings for our real-world signature Σ: in each case, a substantial number of additional predicates becomes available for query formulation and Σ-substitutes are much smaller than the original ontology and than $\bot$-modules.

Conclusion

We have established a relatively complete picture of the complexity of $\mathcal{IQ}$-query emptiness and $\mathcal{CQ}$-predicate emptiness in the $\mathcal{EL}$, DL-Lite and $\mathcal{ALC}$ families of DLs, with complexities ranging from PTime to undecidable. In the case of the $\mathcal{EL}$ and DL-Lite family, the described algorithms are rather simple and easily implemented (for DL-Lite, one could e.g. use a SAT checker). First experiments show that the computed signatures are typically of manageable size, and that the resulting ontology modules are significantly smaller than modules based on other popular notions of modularity. We have also given some first results concerning the complexity of $\mathcal{CQ}$-query emptiness. Figure 3 gives a summary of what we have achieved. Relevant open problems include the exact complexity of $\mathcal{IQ}$-query emptiness in $\mathcal{ALCI}$ and of $\mathcal{CQ}$-query emptiness in $\mathcal{EL}_\bot$ and $\mathcal{ALC}$.

Acknowledgements. We are grateful to Julian Mendez and Dirk Walther for helping with the case study.

References

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

Baader, F.; McGuiness, D. L.; Nardi, D.; and Patel-Schneider, P., eds. 2003. *The Description Logic Handbook*. Cambridge University Press.

Baader, F.; Bienvenu, M.; Lutz, C.; and Wolter, F. 2009. Query answering over DL aboxes: How to pick the relevant symbols. In *Proc. of DL workshop*.

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the $\mathcal{EL}$ envelope. In *Proc. of IJCAI*, 364–369.

Baader, F.; Brandt, S.; and Lutz, C. 2008. Pushing the $\mathcal{EL}$ envelope further. In *Proc. of OWLED workshop*.

Benedikt, M.; Fan, W.; and Geerts, F. 2008. XPath satisfiability in the presence of DTDs. *J. of the ACM* 55(2):1–79.

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.

Calvanese, D.; Giacomo, G. D.; Lembo, D.; Lenzerini, M.; Poggi, A.; Rodriguez-Muro, M.; and Rosati, R. 2009. Ontologies and databases: The DL-Lite approach. In *Reasoning Web*, volume 5689 of *LNCS*, 255–356.

Calvanese, D.; De Giacomo, G.; and Lenzerini, M. 1998. On the decidability of query containment under constraints. In *Proc. of PODS*, 149–158.

Grau, B. C.; Horrocks, I.; Kazakov, Y.; and Sattler, U. 2008. Modular reuse of ontologies: Theory and practice. *J. of Artifical Intelligence Research* 31:273–318.

Konev, B.; Lutz, C.; Walther, D.; and Wolter, F. 2008. Semantic modularity and module extraction in description logics. In *Proc. of ECAI*, 55–59.

Konev, B.; Lutz, C.; Walther, D.; and Wolter, F. 2009. Formal properties of modularisation. In *Modular Ontologies*, volume 5445 of *LNCS*. Springer. 25–66.

Kontchakov, R.; Wolter, F.; and Zakharyaschev, M. 2010. Logic-based ontology comparison and module extraction with an application to DL-Lite. J. of Artificial Intelligence.

Levy, A. 1993. *Irrelevance Reasoning in Knowledge Based Systems*. Ph.D. Dissertation, Stanford University.

Lubyte, L., and Tessaris, S. 2008. Supporting the design of ontologies for data access. In *Proc. of DL workshop*.

Lutz, C., and Wolter, F. 2009. Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. *J. of Symbolic Computation* 45(2):194–228.

Lutz, C.; Toman, D.; and Wolter, F. 2009. Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In *Proc. of IJCAI*, 2070–2075.

Lutz, C. 2008. The complexity of CQ answering in expressive description logics. In *Proc. of IJCAR*, 179–193.

Patel, C.; Cimino, J. J.; Dolby, J.; Fokoue, A.; Kalyanpur, A.; Kershenbaum, A.; Ma, L.; Schonberg, E.; and Srinivas, K. 2007. Matching patient records to clinical trials using ontologies. In *Proc. of ISWC/ASWC*, 816–829.

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

Vardi, M. Y. 1989. Automata theory for database theoreticans. In *Proc. of PODS*, 83–92.