Noname manuscript No.
(will be inserted by the editor)

Forgetting for Knowledge Bases in DL-Lite

Zhe Wang · Kewen Wang · Rodney Topor · Jeff Z. Pan

Received: date / Accepted: date

Abstract To support the reuse and combination of ontologies in Semantic Web applications, it is often necessary to obtain smaller ontologies from existing larger ontologies. In particular, applications may require the omission of certain terms, *e.g.*, concept names and role names, from an ontology. However, the task of omitting terms from an ontology is challenging because the omission of some terms may affect the relationships between the remaining terms in complex ways. We present the first solution to the problem of omitting concepts and roles from knowledge bases of description logics (DLs) by adapting the technique of forgetting, previously used in other domains. Specifically, we first introduce a model-theoretic definition of forgetting for knowledge bases (both TBoxes and ABoxes) in DL-Lite$^{\mathcal{N}}_{bool}$, which is a non-trivial adaption of the standard definition for classical logic, and show that our model-based forgetting satisfies all major criteria of forgetting, which in turn verifies the suitability of our model-based forgetting. We then introduce algorithms that implement forgetting in DL-Lite knowledge bases. We prove that the algorithms are correct with respect to the semantic definition of forgetting. We establish a general framework for defining and comparing different definitions of forgetting by introducing a parameterized forgetting called query-based forgetting. In this framework we identify three specific query-based forgettings. In particular, we show that the model-based forgetting can be embedded in our framework by showing that it coincides with one of these query-based forgettings.

Keywords description logics · forgetting · ontology

1 Introduction

An ontology is a specification of a shared conceptualization of a domain [16]. Ontologies are widely used for representing, storing and processing structural domain knowledge, and

Z. Wang · K. Wang · R. Topor
Griffith University, Australia

J. Z. Pan
The University of Aberdeen, UK

The corresponding author: Kewen Wang
E-mail: k.wang@griffith.edu.au

have been applied in a wide range of practical domains such as medical informatics, bio-informatics and, more recently, the Semantic Web [1]. Among various representation formalisms, description logics (DLs) [4] are well accepted as one of the most successful underlying formalisms for ontologies. Description logics are a class of logics with expressive languages, precisely defined semantics and powerful reasoning systems. By representing an ontology as a DL knowledge base (KB), which consists of a TBox and an ABox, description logics provide ontology applications with logical foundations and reasoning mechanisms. In particular, OWL (Web Ontology Language) [9], the latest W3C standard for ontology markup languages, is based on DLs.

An important and interesting research problem in description logic community is the trade-off between the expressive power of DLs and the efficiency of their reasoning. Much work has been done on restricting the expressive power of DLs in appropriate ways, so that the computational complexity of their reasoning problems can be reduced, and preferably become tractable. As a result, several tractable DLs have been proposed, among which the most influential are the DL-Lite family [6–8] and the $\mathcal{EL}$ -family [3,5]. The DL-Lite family, including a basic description logic DL-Lite$_{core}$ and several extensions, are specially tailored for efficient query answering over ontologies with large amounts of data. In particular, logics in the DL-Lite family have polynomial time computational complexity with respect to most standard reasoning tasks (such as consistency, subsumption and instance checking, but not conjunctive query answering), and LogSpace data complexity with respect to complex query answering.

Recently, a more expressive DL language, called DL-Lite$^{\mathcal{N}}_{bool}$, was proposed in [2]. DL-Lite$^{\mathcal{N}}_{bool}$ extends the basic DL-Lite languages [8] with full boolean operators and number restrictions in its knowledge bases. Also, a class of queries, called positive existential queries (PEQ), was investigated and shown to have low query answering complexity in DL-Lite$^{\mathcal{N}}_{bool}$. In particular, the PEQ answering problem in the Horn subset of DL-Lite$^{\mathcal{N}}_{bool}$, namely DL-Lite$^{\mathcal{N}}_{horn}$, still possesses LogSpace data complexity. This means the LogSpace upper bound for conjunctive queries in DL-Lite is preserved for PEQs in DL-Lite$^{\mathcal{N}}_{horn}$.

As ontologies become larger and more complex, an important problem is how to construct, reuse, update and refine large ontologies efficiently. Recently, ontology reuse has received intensive interest, and different approaches have been proposed. Among several approaches to ontology reuse, the *forgetting* operator has attracted extensive interests in the communities of knowledge representation and DL-based ontologies. Informally, forgetting is a particular form of reasoning that allows a set of elements F (such as propositional variables, predicates, concepts and roles) in a KB to be discarded or hidden in such a way that future reasoning on information irrelevant to F will not be affected. Forgetting has been well investigated in classical logic [24,23] and logic programming [11,12,33].

Forgetting is especially interesting for dynamic ontology management. In applications of extracting, reusing and merging ontologies, we are often required to modify a large ontology into a (smaller) new ontology in a way that certain concepts and roles are omitted/hidden in the new ontology, while the 'meaning' of the original ontology (w.r.t. certain reasoning tasks, such as query answering for a class of queries) is still preserved. Given that OWL and several other major ontology languages are based on description logics, the problem of modifying DL-based ontologies for various application requirements is receiving intensive research interests (see Section 6 for further details).

Forgetting for DLs can be defined in two ways that are closely related: one is analogous to the classical forgetting [24,23], and the other is through uniform interpolation [32]. Classical forgetting is a model-based approach which preserves model equivalence (over certain signatures), whereas uniform interpolation is defined to preserve certain logical entailment.

Although these two approaches coincide in propositional logic and first order logic, they turn out to be different in DL-Lite.

Model-based forgetting has been proposed in [34], which preserves all forms of reasoning in DL-Lite and thus is the strongest form of forgetting. Model-based forgetting can be used to forget about both concepts and roles in DL-Lite TBoxes. The result of forgetting about concepts are always expressible in DL-Lite and a simple algorithm is provided in [34]. However, the result of forgetting about roles in a DL-Lite TBox may not be expressible in general. For this reason, a weaker form of forgetting for DL-Lite TBoxes (uniform interpolants) is introduced in [21] (Definition 16). This definition of forgetting is based on the idea of preserving only DL-Lite concept inclusions in DL-Lite$^{\mathcal{N}}_{bool}$. In short, we refer to this form of forgetting as *b-forgetting* where "*b*" is for "bool". It is shown in [21] that the result of b-forgetting about both concepts and roles is expressible in DL-Lite$^{\mathcal{N}}_{bool}$. It is pointed out in [21] that b-forgetting is too weak to preserve some important semantic properties in DL-Lite$^{\mathcal{N}}_{bool}$. For this reason, b-forgetting is strengthened by preserving more expressive inclusions. Specifically, the syntax of DL-Lite$^{\mathcal{N}}_{bool}$ is extended to a new language DL-Lite$^{u}_{bool}$ by allowing to express that a concept is nonempty, and as a result, a slightly stronger form of forgetting is defined by requiring to preserve the inclusions in DL-Lite$^{u}_{bool}$ rather than only in DL-Lite$^{\mathcal{N}}_{bool}$ [21] (Definition 20). We refer to this forgetting as *u-forgetting*. Although it is not expressible in DL-Lite$^{\mathcal{N}}_{bool}$, the result of u-forgetting is expressible in DL-Lite$^{u}_{bool}$.

We remark that the above three definitions of forgetting are defined only for TBoxes. However, in most applications, an ontology in DL-Lite is expressed as a KB, which is a pair consisting of an ABox and a TBox and thus we believe that dynamic operators for ontology reuse should be defined for DL-Lite KBs rather only for TBoxes.

In this paper, we investigate the issue of semantic forgetting for DL-Lite$^{\mathcal{N}}_{bool}$ KBs. We first define model-based forgetting for DL-Lite$^{\mathcal{N}}_{bool}$ KBs and show several important properties of forgetting. We also introduce a transformation-based algorithm for concept forgetting in DL-Lite$^{\mathcal{N}}_{bool}$ KBs, whose completeness implies the existence of concept forgetting. To provide a unifying framework for defining and comparing different definitions of forgetting, we introduce a parameterized forgetting called *query-based forgetting*, which is a natural generalization of b-forgetting and u-forgetting. The three notions of forgetting introduced in [34, 21] can be naturally extended from TBoxes to KBs. In particular, we show that model-based forgetting, b-forgetting and u-forgetting can all be characterized by query-based forgetting. Thus, our approach actually provides a hierarchy of forgetting for DL-Lite.

We choose DL-Lite$^{\mathcal{N}}_{bool}$ in this paper for at least two reasons: First, it is one of the most expressive members of the DL-Lite family. Second, it is unclear to us if the algorithms developed in this paper can be extended to more expressive DLs. We agree with [13] that it would be an interesting but challenging problem to develop algorithms for determining the existence of and computing forgetting for DLs such as $\mathcal{ALC}$ and $\mathcal{SHIQ}$. A first attempt in this direction is reported in [35]

In addition, we note that, while DL-Lite$^{\mathcal{N}}_{bool}$ is a fragment of first order logic (FOL) and forgetting for FOL has been investigated in [24], one cannot define forgetting for DL-Lite$^{\mathcal{N}}_{bool}$ KBs by transforming them into theories in FOL. There are two reasons for this: First, the result of forgetting in FOL may not be in FOL as mentioned in [24]. Second, even if the result of forgetting is in FOL, it may not correspond to a KB in DL-Lite$^{\mathcal{N}}_{bool}$.

The work in this paper significantly extends our conference paper [34] in at least three ways: First, all definitions and results have been extended from TBoxes to KBs. Next, proofs of all results are included. Last, we introduce and apply query-based forgetting as a general framework for forgetting.

While it is not hard to extend the definitions of forgetting to KBs, our efforts show that it is non-trivial to extend results of forgetting in TBoxes to forgetting in KBs, due to the involvement of ABoxes. This can be seen from the following aspects: 1) the algorithm of forgetting in KBs is more complex, as changes in the TBox and the ABox both affect the models of the KB in a complex way; 2) since KB reasoning tasks are different from those in a single TBox, forgetting in KBs is expected to possess some different reasoning properties concerning KB reasoning; 3) forgetting in KBs has different expressibility properties from those of TBoxes; 4) some properties of forgetting in TBoxes are not straightforward to be generalized to forgetting in KBs, because of the logical connection between TBox and ABox, which can be seen from the proofs.

The main contributions of this paper can be summarized as follows:

- We introduce a model-based definition of forgetting about both concepts and roles for KBs in DL-Lite$^{\mathcal{N}}_{bool}$. Reasoning and expressibility properties of forgetting in KBs are studied in detail, as they are important for applications of DL-Lite ontology reuse and combination. The model-based definition of forgetting describes an intuitive ontology forgetting operation, and these properties can serve as criteria for evaluating various ontology forgetting operations.
- We provide a resolution-like algorithm for forgetting about concepts in DL-Lite$^{\mathcal{N}}_{bool}$ KBs. The algorithm is capable to handle concept disjunction, which is one of the major extensions in DL-Lite$^{\mathcal{N}}_{bool}$ of traditional DL-Lite languages, and which is also the cause of a exponential blow up in algorithm complexity. It is proved that the algorithm is complete for concept forgetting in DL-Lite$^{\mathcal{N}}_{bool}$ KBs. The algorithm provides a basis for implementing forgetting operations in DL-Lite ontology applications.
- We propose and study several alternative definitions of forgetting based on query answering (that is, query-based forgetting). In particular, three definitions of query-based forgetting are proposed and their expressibility properties are investigated. We show these three query-based forgettings correspond to model-based forgetting, b-forgetting and u-forgetting.

The rest of the paper is organized as follows. Some basics of DL-Lite$^{\mathcal{N}}_{bool}$ and DL-Lite$^{\mathcal{N}}_{horn}$ are briefly recalled in Section 2. We present the model-based definition of forgetting in DL-Lite$^{\mathcal{N}}_{bool}$ KBs in Section 3 and show the result of forgetting has desirable properties. In Section 4, we introduce our algorithms for computing the result of forgetting about concepts in a DL-Lite$^{\mathcal{N}}_{bool}$ KB, and show the algorithms are correct with respect to the semantic definition. In Section 5, we define query-based forgetting and discuss three interesting variants of it. We also present a detailed discussion about the connection between query-based forgetting and model-based forgetting and uniform interpolation. Finally, Section 6 provides related work and Section 7 concludes the paper.

2 Preliminaries

DL-Lite is designed as a family of lightweight ontology languages. DL-Lite is able to express most features in UML [1] class diagrams and still has low reasoning complexity [6]. Besides standard reasoning tasks such as subsumption between concepts and consistency of knowledge bases, the issue of answering complex queries is especially considered.

[1] `http://www.nlm.nih.gov/research/umls`

In DL-Lite$^{\mathcal{N}}_{bool}$ language, complex roles and concepts are defined as follows:

$$\begin{aligned} R &\longleftarrow P \mid P^- \\ B &\longleftarrow \top \mid \bot \mid A \mid \ \geqslant n\, R \\ C &\longleftarrow B \mid \neg C \mid C_1 \sqcap C_2 \end{aligned}$$

Here, $n \geq 1$ is a constant, A is a concept and P is a role name (with P^- as its inverse). B is called a *basic concept* and C is called a *general concept*. Other concept constructors such as $\exists R$, $\leqslant n\, R$ and $C_1 \sqcup C_2$ will be used as standard abbreviations. We will also call $\top$ ($\bot$) an *empty conjunction* (resp., *empty disjunction*).

A general concept C is said to be in disjunctive normal form (DNF) if C is a disjunction of conjunctions whose conjuncts are all basic concepts or their negations. C is said to be in conjunctive normal form (CNF) if C is a conjunction of disjunctions whose disjuncts are all basic concepts or their negations. It is not hard to see that any DL-Lite$^{\mathcal{N}}_{bool}$ concept can be transformed into DNF or CNF through De Morgan's laws and distributive laws.

A DL-Lite$^{\mathcal{N}}_{bool}$ TBox $\mathcal{T}$ is a finite set of *concept inclusions*, or briefly *inclusions*, of the form $C_1 \sqsubseteq C_2$, where C_1 and C_2 are general concepts. A DL-Lite$^{\mathcal{N}}_{bool}$ ABox $\mathcal{A}$ is a finite set of *membership assertions*, or briefly *assertions*, of the form $C(a)$ or $R(a, b)$, where a and b are individual names. A DL-Lite$^{\mathcal{N}}_{bool}$ knowledge base (KB) is a pair $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$.

Given a KB $\mathcal{K}$, $\mathsf{Ind}(\mathcal{K})$ denotes the set of all individual names in $\mathcal{K}$ and $\mathsf{Num}(\mathcal{K})$ the set of all numerical parameters in $\mathcal{K}$ together with 1.

The semantics of DL-Lite is specified by interpretations. An interpretation $\mathcal{I}$ is a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set called the *domain* and $\cdot^{\mathcal{I}}$ is an interpretation function which associates each atomic concept A with a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, each atomic role P with a binary relation $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and each individual name a with an element $a^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ such that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ for each pair of individual names a, b (unique name assumption).

Using $\sharp(S)$ to denote the cardinality of a set S, the interpretation function $\cdot^{\mathcal{I}}$ can be extended to general concepts:

$$\begin{aligned} (P^-)^{\mathcal{I}} &= \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid (b^{\mathcal{I}}, a^{\mathcal{I}}) \in P^{\mathcal{I}}\} \\ (\geqslant n\, R)^{\mathcal{I}} &= \{a^{\mathcal{I}} \mid \sharp(\{b^{\mathcal{I}} \mid (a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}\}) \geq n\} \\ (\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\ (C_1 \sqcap C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \end{aligned}$$

Two general concepts are said to be *equivalent* if they are associated with the same set in any interpretation.

An interpretation $\mathcal{I}$ is a model of inclusion $C_1 \sqsubseteq C_2$ iff $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$. $\mathcal{I}$ is a model of assertion $C(a)$ (resp., $R(a, b)$) if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ (resp., $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$). Note that assertion $\bot(a)$ is allowed and is an assertion with no model. $\mathcal{I}$ is called a model of a TBox $\mathcal{T}$ (or ABox $\mathcal{A}$) if $\mathcal{I}$ is a model of each inclusion (resp., assertion) in $\mathcal{T}$ (resp., $\mathcal{A}$). Two inclusions (or resp., assertions, TBoxes, ABoxes) are said to be *equivalent* if they have exactly the same models.

$\mathcal{I}$ is a model of a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ if $\mathcal{I}$ is a model of both $\mathcal{T}$ and $\mathcal{A}$. We use $\mathsf{Mod}(\mathcal{K})$ to denote the set of all models of $\mathcal{K}$. Two KBs $\mathcal{K}_1, \mathcal{K}_2$ that have the same models are said to be equivalent, denoted $\mathcal{K}_1 \equiv \mathcal{K}_2$. A KB $\mathcal{K}$ logically implies an inclusion or assertion α (resp., KB $\mathcal{K}'$), denoted $\mathcal{K} \models \alpha$ (resp., $\mathcal{K} \models \mathcal{K}'$), if all models of $\mathcal{K}$ are also models of α (resp., $\mathcal{K}'$). Note that $\mathcal{K}_1 \models \mathcal{K}_2$ iff $\mathcal{K}_1 \models \alpha$ for each inclusion and each assertion α in $\mathcal{K}_2$. $\mathcal{K}_1 \equiv \mathcal{K}_2$ iff $\mathcal{K}_1 \models \mathcal{K}_2$ and $\mathcal{K}_2 \models \mathcal{K}_1$.

A KB $\mathcal{K}$ is consistent if it has at least one model. Given a set $\mathcal{S}$ of concept and role names in $\mathcal{K}$, we say $\mathcal{K}$ is *coherent over* $\mathcal{S}$ if there is a model $\mathcal{I}$ of $\mathcal{K}$ such that for each concept or role name $E \in \mathcal{S}$, $E^{\mathcal{I}} \neq \emptyset$. $\mathcal{K}$ is said to be *coherent* if $\mathcal{K}$ is coherent over $\mathcal{S}$, with $\mathcal{S}$ being the set of all concept and role names in $\mathcal{K}$.

A positive existential query (PEQ) $q(\mathbf{x})$ (or simply q) over a KB $\mathcal{K}$ is a (first order logic) formula $\exists \mathbf{y}.\varphi(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}, \mathbf{y}$ are lists of variables, $\varphi(\mathbf{x}, \mathbf{y})$ is constructed, using only $\wedge$ and $\vee$, from atoms of the form $C(t)$ or $R(t_1, t_2)$ and each t is either a variable from $\mathbf{x}, \mathbf{y}$ or an individual name. A PEQ is grounded if it does not have any free variable.

The Horn fragment of DL-Lite$^{\mathcal{N}}_{bool}$, denoted DL-Lite$^{\mathcal{N}}_{horn}$, is defined in a way analogous to the Horn fragment in first order logic. An inclusion in DL-Lite$^{\mathcal{N}}_{horn}$ TBox is of the form $D \sqsubseteq B$ where $D = \bigsqcap_{k \geq 0} B_k$ is a (possibly empty) conjunction of basic concepts (*i.e.*, $D = \top$ when $k = 0$). An assertion in DL-Lite$^{\mathcal{N}}_{horn}$ ABox is of the form $B(a)$ or $R(a, b)$.

The data complexity of the PEQ answering problem for DL-Lite$^{\mathcal{N}}_{horn}$ KBs is in LogSpace, while for DL-Lite$^{\mathcal{N}}_{bool}$ it is coNP-complete.

In the following example we present a DL-Lite$^{\mathcal{N}}_{bool}$ KB.

Example 2.1 Suppose a KB $\mathcal{K}$ for a research center defines four concepts ($\mathit{Researcher}$, Paper, $\mathit{Professor}$ and RA) and one role $\mathit{hasPublications}$, together with the information about some researchers and their publications. "$\mathit{RA}(a)$" states that a is a research assistant, while "$\mathit{hasPublications}(a, b)$" means that researcher a has paper b published.

The TBox $\mathcal{T}$ of $\mathcal{K}$ consists of the following inclusions:

(1) $\exists \mathit{hasPublications} \sqsubseteq \mathit{Researcher}$,
(2) $\exists \mathit{hasPublications}^- \sqsubseteq \mathit{Paper}$,
(3) $\mathit{Professor} \sqsubseteq \geqslant 5\ \mathit{hasPublications}$,
(4) $\mathit{Researcher} \sqsubseteq \mathit{Professor} \sqcup \mathit{RA}$,
(5) $\mathit{Professor} \sqcap \mathit{RA} \sqsubseteq \bot$.

The meaning of the inclusions (1) and (2) is obvious. The inclusion (3) states that a professor of the center must have at least 5 papers published. The inclusion (4) specifies that a researcher in the center must be either a professor or a research assistant while (5) requires that no one is both a professor and research assistant.

The ABox $\mathcal{A}$ in $\mathcal{K}$ consists of the following assertions:

$\mathit{Professor}(\mathit{John})$ and $\mathit{hasPublications}(\mathit{John}, \mathit{P75})$,

which states that John is a professor and he has published the paper $\mathit{P75}$.

Note that inclusion $\mathit{Researcher} \sqsubseteq \mathit{Professor} \sqcup \mathit{RA}$ is not allowed in DL-Lite$^{\mathcal{N}}_{horn}$. The KB consisting of all other inclusions and assertions in $\mathcal{K}$ is a DL-Lite$^{\mathcal{N}}_{horn}$ KB.

In this paper, a *signature* is a finite set of concept and role names. Individual names are not included in signatures. Given an expression (*i.e.*, a concept, a TBox, an ABox, a query, a KB or a language) E, we will denote by $\mathsf{Sig}(E)$ the set of all concept and role names in E.

3 Forgetting in DL-Lite$^{\mathcal{N}}_{bool}$ Knowledge Bases

In this section, we define the operation of forgetting a set of concept and role names from a DL-Lite$^{\mathcal{N}}_{bool}$ KB. We will first give a definition of forgetting based on model equivalence and investigate the properties of forgetting. After this, we will discuss the expressibility properties of the forgetting operations both in DL-Lite$^{\mathcal{N}}_{bool}$ and DL-Lite$^{\mathcal{N}}_{horn}$.

Throughout the paper, we use $\mathcal{L}$ as a abbreviation of DL-Lite$^{\mathcal{N}}_{bool}$. Suppose $\mathcal{K}$ is a KB in $\mathcal{L}$. For simplicity, we also call $\mathcal{K}$ an $\mathcal{L}$-KB. Let $\mathcal{S}$ be a set of concept and role names in $\mathcal{L}$. Informally, the KB that results from forgetting about $\mathcal{S}$ in $\mathcal{K}$ should: (1) not contain any new concept or role name, or any occurrence of concept or role name in $\mathcal{S}$, (2) be logically weaker than $\mathcal{K}$, and (3) preserve the original meanings of the concepts and roles other than those in $\mathcal{S}$. Our model-based definition of forgetting in DL-Lite is an adaption of the corresponding definition for forgetting in classical logic [24, 23].

As with classical forgetting, a notion of *model equivalence* is needed for defining forgetting in DL-Lite. Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be two interpretations of $\mathcal{L}$. We define $\mathcal{I}_1 \sim_{\mathcal{S}} \mathcal{I}_2$ if $\mathcal{I}_1$ and $\mathcal{I}_2$ agree on all individual, concept and role names except for those in $\mathcal{S}$, *i.e.*,

1. $\mathcal{I}_1$ and $\mathcal{I}_2$ have the same domain, and interpret every individual name identically (*i.e.*, $a^{\mathcal{I}_1} = a^{\mathcal{I}_2}$ for each individual name a);
2. for every concept name A not in $\mathcal{S}$, $A^{\mathcal{I}_1} = A^{\mathcal{I}_2}$;
3. for every role name P not in $\mathcal{S}$, $P^{\mathcal{I}_1} = P^{\mathcal{I}_2}$.

Clearly, $\sim_{\mathcal{S}}$ is an equivalence relation.

Definition 3.1 Let $\mathcal{K}$ be an $\mathcal{L}$-KB and $\mathcal{S}$ a signature. We call KB $\mathcal{K}'$ a *result of model-based forgetting* about $\mathcal{S}$ in $\mathcal{K}$ if:

- $\mathsf{Sig}(\mathcal{K}') \subseteq \mathsf{Sig}(\mathcal{K}) - \mathcal{S}$, and
- $\mathsf{Mod}(\mathcal{K}') = \{\, \mathcal{I}'$ is an interpretation in $\mathcal{L} \mid$ there is an $\mathcal{I} \in \mathsf{Mod}(\mathcal{K})$ s.t. $\mathcal{I} \sim_{\mathcal{S}} \mathcal{I}' \,\}$.

Before introducing alternative definitions of forgetting, we will refer to model-based forgetting as *forgetting*.

While forgetting is defined in DL-Lite$^{\mathcal{N}}_{bool}$ here, we note that the definition can be applied to any DL language.

It follows from the above definition that the result of forgetting about a signature $\mathcal{S}$ in a DL KB $\mathcal{K}$, when it exists, is unique up to KB equivalence. That is, if both $\mathcal{K}'$ and $\mathcal{K}''$ are results of forgetting about $\mathcal{S}$ in $\mathcal{K}$, then they are equivalent. For this reason, we use $\mathsf{forget}(\mathcal{K}, \mathcal{S})$ to denote a result of forgetting about $\mathcal{S}$ in $\mathcal{K}$. In the remainder of this paper, whenever $\mathsf{forget}(\mathcal{K}, \mathcal{S})$ is used, we always assume that it exists.

Example 3.1 (Cont. of Example 2.1) Suppose we want to forget about concept $\mathit{Professor}$ in $\mathcal{K}$, then $\mathsf{forget}(\mathcal{K}, \{\mathit{Professor}\})$ consists of the following inclusions and assertions:

$\exists \mathit{hasPublications} \sqsubseteq \mathit{Researcher}$,
$\exists \mathit{hasPublications}^- \sqsubseteq \mathit{Paper}$,
$\mathit{Researcher} \sqsubseteq \geqslant 5\ \mathit{hasPublications} \sqcup \mathit{RA}$,
$\geqslant 5\ \mathit{hasPublications}(\mathit{John})$, $\neg \mathit{RA}(\mathit{John})$, and $\mathit{hasPublications}(\mathit{John}, \mathit{P75})$.

The inclusion $\mathit{Researcher} \sqsubseteq \geqslant 5\ \mathit{hasPublications} \sqcup \mathit{RA}$ can be equivalently presented as $\mathit{Researcher} \sqcap \leqslant 4\ \mathit{hasPublications} \sqsubseteq \mathit{RA}$, which says that those researchers who have at most 4 papers are research assistants. The assertions state that John has at least 5 papers pulished, among which is $\mathit{P75}$, and he is not an research assistant.

We first give an equivalent characterization of the model-based forgetting, which is helpful in proofs.

Proposition 3.1 *Let $\mathcal{K}$ be an $\mathcal{L}$-KB and $\mathcal{S}$ a signature. Then any $\mathcal{K}'$ over* $\mathsf{Sig}(\mathcal{K}) - \mathcal{S}$ *satisfying the following two conditions is a result of forgetting about $\mathcal{S}$ in $\mathcal{K}$, i.e.,* $\mathsf{forget}(\mathcal{K}, \mathcal{S}) = \mathcal{K}'$*:*

(1) $\mathcal{K} \models \mathcal{K}'$, and
(2) for each model $\mathcal{I}'$ of $\mathcal{K}'$, there exists a model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \sim_{\mathcal{S}} \mathcal{I}'$.

Proof Denote $\mathcal{M} = \mathsf{Mod}(\mathsf{forget}(\mathcal{K}, \mathcal{S})) = \{\, \mathcal{I}'$ is an interpretation in $\mathcal{L} \mid$ there is an $\mathcal{I} \in \mathsf{Mod}(\mathcal{K})$ s.t. $\mathcal{I} \sim_{\mathcal{S}} \mathcal{I}' \,\}$, which we will simply present as $\{\, \mathcal{I}' \mid \exists \mathcal{I} \in \mathsf{Mod}(\mathcal{K}), \mathcal{I} \sim_{\mathcal{S}} \mathcal{I}' \,\}$ in the following proofs.

By the above condition (2), $\mathsf{Mod}(\mathcal{K}') \subseteq \mathcal{M}$. On the other hand, if $\mathcal{I}' \in \mathcal{M}$, then there exists $\mathcal{I}$ such that $\mathcal{I} \in \mathsf{Mod}(\mathcal{K})$ and $\mathcal{I} \sim_{\mathcal{S}} \mathcal{I}'$. From condition (1) and $\mathcal{I} \in \mathsf{Mod}(\mathcal{K})$, $\mathcal{I} \in \mathsf{Mod}(\mathcal{K}')$. Note that $\mathcal{I}'$ and $\mathcal{I}$ coincide on $\mathsf{Sig}(\mathcal{K}) - \mathcal{S}$. Thus, by $\mathsf{Sig}(\mathcal{K}') \subseteq \mathsf{Sig}(\mathcal{K}) - \mathcal{S}$, we have $\mathcal{I}' \in \mathsf{Mod}(\mathcal{K}')$. This implies $\mathcal{M} \subseteq \mathsf{Mod}(\mathcal{K}')$.

Therefore, $\mathsf{Mod}(\mathcal{K}') = \mathcal{M}$. That is, $\mathsf{forget}(\mathcal{K}, \mathcal{S}) = \mathcal{K}'$.

■

In the rest of this section, we show that our definition of forgetting for DL-Lite KBs possesses several desirable properties. In particular, it preserves reasoning properties of the KB.

Proposition 3.2 *Let $\mathcal{K}$ be an $\mathcal{L}$-KB and $\mathcal{S}$ a signature. Then the following properties are satisfied:*

Consistency*:* $\mathsf{forget}(\mathcal{K}, \mathcal{S})$ *is consistent iff $\mathcal{K}$ is consistent;*
Coherence*:* $\mathsf{forget}(\mathcal{K}, \mathcal{S})$ *is coherent iff $\mathcal{K}$ is coherent over* $\mathsf{Sig}(\mathcal{K}) - \mathcal{S}$*;*
Consequence Invariance*: for any inclusion or assertion α with* $\mathsf{Sig}(\alpha) \cap \mathcal{S} = \emptyset$, $\mathsf{forget}(\mathcal{K}, \mathcal{S}) \models \alpha$ *iff* $\mathcal{K} \models \alpha$*;*
PEQ Invariance*: for any grounded PEQ q with* $\mathsf{Sig}(q) \cap \mathcal{S} = \emptyset$, $\mathsf{forget}(\mathcal{K}, \mathcal{S}) \models q$ *iff* $\mathcal{K} \models q$.

By Definition 3.1, the above properties are straightforward, as our forgetting preserves the exact meaning of the remaining concepts and roles.

The following proposition says that if $\mathcal{K}_2$ is logically weaker than (resp., equivalent to) $\mathcal{K}_1$, then after forgetting, the results still have the same relation. Thus it shows that forgetting preserves logical relations between KBs.

Proposition 3.3 (KB Implication) *Let $\mathcal{K}_1, \mathcal{K}_2$ be two $\mathcal{L}$-KBs and $\mathcal{S}$ a signature. Then*

1. $\mathcal{K}_1 \models \mathcal{K}_2$ *implies* $\mathsf{forget}(\mathcal{K}_1, \mathcal{S}) \models \mathsf{forget}(\mathcal{K}_2, \mathcal{S})$*, and*
2. $\mathcal{K}_1 \equiv \mathcal{K}_2$ *implies* $\mathsf{forget}(\mathcal{K}_1, \mathcal{S}) \equiv \mathsf{forget}(\mathcal{K}_2, \mathcal{S})$.

Proof Denote $\mathcal{M}'_i = \mathsf{Mod}(\mathsf{forget}(\mathcal{K}_i, \mathcal{S})) = \{\mathcal{I}' \mid \exists \mathcal{I} \in \mathsf{Mod}(\mathcal{K}_i), \mathcal{I} \sim_{\mathcal{S}} \mathcal{I}'\}$ for $i = 1, 2$. Then $\mathsf{Mod}(\mathcal{K}_1) \subseteq \mathsf{Mod}(\mathcal{K}_2)$ implies $\mathcal{M}'_1 \subseteq \mathcal{M}'_2$.

■

The following property is useful for ontology extension and partial reuse. It says that for two ontologies, as long as they do not share common concepts or roles in signature $\mathcal{S}$, forgetting about $\mathcal{S}$ in their combination is the same as combining their respective results of forgetting.

Proposition 3.4 (KB Union) *Let $\mathcal{K}_1, \mathcal{K}_2$ be two $\mathcal{L}$-KBs and $\mathcal{S}$ a signature. If* $\mathsf{Sig}(\mathcal{K}_1) \cap \mathsf{Sig}(\mathcal{K}_2) \cap \mathcal{S} = \emptyset$*, then*

$$\mathsf{forget}(\mathcal{K}_1 \cup \mathcal{K}_2, \mathcal{S}) = \mathsf{forget}(\mathcal{K}_1, \mathcal{S}) \cup \mathsf{forget}(\mathcal{K}_2, \mathcal{S}).$$

Proof Let $\mathcal{M}_i = \mathsf{Mod}(\mathcal{K}_i)$ and $\mathcal{M}'_i = \mathsf{Mod}(\mathsf{forget}(\mathcal{K}_i, \mathcal{S}))$ for $i = 1, 2$. Denote $\mathcal{M}' = \mathsf{Mod}(\mathsf{forget}(\mathcal{K}_1 \cup \mathcal{K}_2, \mathcal{S})) = \{\mathcal{I}' \mid \exists \mathcal{I} \in \mathcal{M}_1 \cap \mathcal{M}_2, \mathcal{I} \sim_{\mathcal{S}} \mathcal{I}'\}$. It is easy to see that $\mathcal{M}' \subseteq \mathcal{M}'_1$ and $\mathcal{M}' \subseteq \mathcal{M}'_2$, and thus $\mathcal{M}' \subseteq \mathcal{M}'_1 \cap \mathcal{M}'_2$. We want to show that $\mathcal{M}'_1 \cap \mathcal{M}'_2 \subseteq \mathcal{M}'$.

For any model $\mathcal{I}' \in \mathcal{M}'_1 \cap \mathcal{M}'_2$, there exists a model $\mathcal{I}_1 \in \mathcal{M}_1$ with $\mathcal{I}_1 \sim_{\mathcal{S}} \mathcal{I}'$ and a model $\mathcal{I}_2 \in \mathcal{M}_2$ with $\mathcal{I}_2 \sim_{\mathcal{S}} \mathcal{I}'$. Thus, $\mathcal{I}' \sim_{\mathcal{S}} \mathcal{I}_1 \sim_{\mathcal{S}} \mathcal{I}_2$.

Since $\mathsf{Sig}(\mathcal{K}_1) \cap \mathcal{S}$ and $\mathsf{Sig}(\mathcal{K}_2) \cap \mathcal{S}$ are disjoint, we can construct an interpretation $\mathcal{I}$ such that: (1) $\mathcal{I} \sim_{\mathcal{S}} \mathcal{I}'$; (2) $\mathcal{I}$ and $\mathcal{I}_i$ coincide on $\mathsf{Sig}(\mathcal{K}_i) \cap \mathcal{S}$ for $i = 1, 2$. Obviously, $\mathcal{I}$ is a model of both $\mathcal{K}_1$ and $\mathcal{K}_2$, *i.e.*, $\mathcal{I} \in \mathcal{M}_1 \cap \mathcal{M}_2$. So we have $\mathcal{I}' \in \mathcal{M}'$ and thus $\mathcal{M}'_1 \cap \mathcal{M}'_2 \subseteq \mathcal{M}'$.

■

An interesting special case of Proposition 3.4 is when the signature of $\mathcal{K}_1$ or $\mathcal{K}_2$ is disjoint with $\mathcal{S}$.

Corollary 3.1 *Let $\mathcal{K}_1, \mathcal{K}_2$ be two $\mathcal{L}$-KBs and $\mathcal{S}$ a signature. If $\mathsf{Sig}(\mathcal{K}_2) \cap \mathcal{S} = \emptyset$, then*

$$\mathsf{forget}(\mathcal{K}_1 \cup \mathcal{K}_2, \mathcal{S}) = \mathsf{forget}(\mathcal{K}_1, \mathcal{S}) \cup \mathcal{K}_2.$$

In a scenario of ontology extension and partial reuse, the above property guarantees that, after forgetting about $\mathcal{S}$ in $\mathcal{K}_1$, it is safe to extend $\mathsf{forget}(\mathcal{K}_1, \mathcal{S})$ with any other ontology $\mathcal{K}_2$ (or reuse $\mathsf{forget}(\mathcal{K}_1, \mathcal{S})$ within context ontology $\mathcal{K}_2$) that does not contain concepts or roles in $\mathcal{S}$.

This property is also useful for computation of forgetting. Note that each KB $\mathcal{K}$ can be divided into two parts $\mathcal{K} = \mathcal{K}_1 \cup \mathcal{K}_2$ where $\mathsf{Sig}(\mathcal{K}_2) \cap \mathcal{S} = \emptyset$. As $\mathsf{forget}(\mathcal{K}, \mathcal{S}) = \mathsf{forget}(\mathcal{K}_1, \mathcal{S}) \cup \mathcal{K}_2$, we only need to consider the subset $\mathcal{K}_1$ when computing the result of forgetting about $\mathcal{S}$ in $\mathcal{K}$.

Another special case is when ontologies share no common concept or role name. In this case, forgetting can be performed in an ad hoc way before extension.

Combining Proposition 3.2 and Proposition 3.4, we have the following corollary.

Corollary 3.2 *Let $\mathcal{K}_1, \mathcal{K}_2$ be two $\mathcal{L}$-KBs and $\mathcal{S}$ a signature. If $\mathsf{Sig}(\mathcal{K}_1) \cap \mathsf{Sig}(\mathcal{K}_2) \cap \mathcal{S} = \emptyset$, then*

1. $\mathsf{forget}(\mathcal{K}_1, \mathcal{S}) \cup \mathsf{forget}(\mathcal{K}_2, \mathcal{S})$ *is consistent iff $\mathcal{K}_1 \cup \mathcal{K}_2$ is consistent;*
2. $\mathsf{forget}(\mathcal{K}_1, \mathcal{S}) \cup \mathsf{forget}(\mathcal{K}_2, \mathcal{S})$ *is coherent iff $\mathcal{K}_1 \cup \mathcal{K}_2$ is coherent over* $\mathsf{Sig}(\mathcal{K}_1 \cup \mathcal{K}_2) - \mathcal{S}$*;*
3. *for any inclusion or assertion α with $\mathsf{Sig}(\alpha) \cap \mathcal{S} = \emptyset$, $\mathsf{forget}(\mathcal{K}_1, \mathcal{S}) \cup \mathsf{forget}(\mathcal{K}_2, \mathcal{S}) \models \alpha$ iff $\mathcal{K}_1 \cup \mathcal{K}_2 \models \alpha$;*
4. *for any grounded PEQ q with $\mathsf{Sig}(q) \cap \mathcal{S} = \emptyset$, $\mathsf{forget}(\mathcal{K}_1, \mathcal{S}) \cup \mathsf{forget}(\mathcal{K}_2, \mathcal{S}) \models q$ iff $\mathcal{K}_1 \cup \mathcal{K}_2 \models q$.*

We have shown properties of forgetting concerning relations between KBs. Now we discuss some properties concerning signatures. The following proposition shows that the forgetting operation can be divided into steps, with a part of the signature forgotten in each step.

Proposition 3.5 (Signature Union) *Let $\mathcal{K}$ be an $\mathcal{L}$-KB and $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathsf{Sig}(\mathcal{L})$. Then*

$$\mathsf{forget}(\mathcal{K}, \mathcal{S}_1 \cup \mathcal{S}_2) = \mathsf{forget}(\mathsf{forget}(\mathcal{K}, \mathcal{S}_1), \mathcal{S}_2).$$

Proof Let $\mathcal{M}' = \mathsf{Mod}(\mathsf{forget}(\mathcal{K}, \mathcal{S}_1)) = \{\mathcal{I}' \mid \exists \mathcal{I} \in \mathsf{Mod}(\mathcal{K}), \mathcal{I} \sim_{\mathcal{S}_1} \mathcal{I}'\}$ and $\mathcal{M}'' = \{\mathcal{I}' \mid \exists \mathcal{I} \in \mathcal{M}', \mathcal{I} \sim_{\mathcal{S}_2} \mathcal{I}'\}$. We have $\mathcal{M}'' = \mathsf{Mod}(\mathsf{forget}(\mathsf{forget}(\mathcal{K}, \mathcal{S}_1), \mathcal{S}_2))$. Then it is not hard to see that $\mathcal{M}'' = \{\mathcal{I}' \mid \exists \mathcal{I} \in \mathsf{Mod}(\mathcal{K}), \mathcal{I} \sim_{\mathcal{S}_1 \cup \mathcal{S}_2} \mathcal{I}'\}$. That is $\mathcal{M}'' = \mathsf{Mod}(\mathsf{forget}(\mathcal{K}, \mathcal{S}_1 \cup \mathcal{S}_2))$.

■

To compute the result of forgetting about $\mathcal{S}$ in $\mathcal{K}$, it is equivalent to forget the concept and role names in $\mathcal{S}$ one by one.

A direct conclusion is that the forgetting operation does not rely on the order in which concept and role names are forgotten.

Corollary 3.3 *Let $\mathcal{K}$ be an $\mathcal{L}$-KB and $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathsf{Sig}(\mathcal{L})$. Then* $\mathsf{forget}(\mathsf{forget}(\mathcal{K}, \mathcal{S}_1), \mathcal{S}_2) \equiv \mathsf{forget}(\mathsf{forget}(\mathcal{K}, \mathcal{S}_2), \mathcal{S}_1)$.

As more concepts and roles are forgotten, the result of forgetting becomes logically weaker.

Proposition 3.6 *Let $\mathcal{K}$ be an $\mathcal{L}$-KB and $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathsf{Sig}(\mathcal{L})$. If $\mathcal{S}_1 \subseteq \mathcal{S}_2$, then* $\mathsf{forget}(\mathcal{K}, \mathcal{S}_1) \models \mathsf{forget}(\mathcal{K}, \mathcal{S}_2)$.

Proof Let $\mathcal{M}' = \{\mathcal{I}' \mid \exists \mathcal{I} \in \mathsf{Mod}(\mathcal{K}), \mathcal{I} \sim_{\mathcal{S}_1} \mathcal{I}'\}$ and $\mathcal{M}'' = \{\mathcal{I}' \mid \exists \mathcal{I} \in \mathsf{Mod}(\mathcal{K}), \mathcal{I} \sim_{\mathcal{S}_2} \mathcal{I}'\}$. If $\mathcal{S}_1 \subseteq \mathcal{S}_2$, then $\mathcal{M}' \subseteq \mathcal{M}''$.

■

Definition 3.1 clearly captures our informal understanding of forgetting. However, given a DL-Lite$^{\mathcal{N}}_{bool}$ KB $\mathcal{K}$ and a set $\mathcal{S}$ of concept and role names, a result of forgetting about $\mathcal{S}$ in $\mathcal{K}$ may not be expressible in DL-Lite$^{\mathcal{N}}_{bool}$. The following example provides some intuition about this point.

Example 3.2 From the KB $\mathcal{K}$ in Example 2.1, we know that $John$ has at least 5 publications. Suppose we want to forget about role name $hasPublications$ in $\mathcal{K}$, we need to express in the result of forgetting that there are at least 5 publications for the whole center. However, it seems that DL-Lite$^{\mathcal{N}}_{bool}$ is unable to express such a constraint unless new individual names or new role names are introduced.

However, if $\mathcal{S}$ contains only concept names, we have the following positive result.

Theorem 3.1 *Let $\mathcal{K}$ be an $\mathcal{L}$-KB and $\mathcal{S}$ a set of concept names. Then* $\mathsf{forget}(\mathcal{K}, \mathcal{S})$ *is always expressible in DL-Lite$^{\mathcal{N}}_{bool}$.*

A natural question is whether the result of forgetting about concept names from a DL-Lite$^{\mathcal{N}}_{horn}$ KB is still expressible in DL-Lite$^{\mathcal{N}}_{horn}$. Unfortunately, this is not always the case. This can be seen from Example 3.1: One cannot express non-membership relations in DL-Lite$^{\mathcal{N}}_{horn}$ KBs, *e.g.*, $\neg RA(John)$ is not expressible.

However, when the ABox of a KB is empty, we have the following positive result.

Theorem 3.2 *For any DL-Lite$^{\mathcal{N}}_{horn}$ KB $\mathcal{K} = \langle \mathcal{T}, \emptyset \rangle$ and any set $\mathcal{S}$ of concept names,* $\mathsf{forget}(\mathcal{K}, \mathcal{S})$ *is always expressible in DL-Lite$^{\mathcal{N}}_{horn}$.*

In the next section, we will introduce an algorithm for computing the result of forgetting concept names in a DL-Lite$^{\mathcal{N}}_{bool}$ KB. From the soundness and completeness of the algorithm, we can immediately conclude the correctness of Theorems 3.1 and 3.2.

4 Computing Concept Forgetting in DL-Lite$^{\mathcal{N}}_{bool}$

In this section, we introduce an algorithm for computing the results of forgetting about concepts in DL-Lite$^{\mathcal{N}}_{bool}$ KBs. We prove that our algorithm is sound and complete with respect to the semantic definition of forgetting in the previous section. Our algorithm shows that the result of forgetting about concepts in a DL-Lite$^{\mathcal{N}}_{bool}$ KB can always be obtained using simple syntax-based transformations.

Before presenting the algorithm, we will first show that each DL-Lite$^{\mathcal{N}}_{bool}$ KB can be equivalently transformed into a normal form. In what follows, we will call a basic concept or its negation a *literal concept*.

We first introduce a normal form for TBoxes in DL-Lite$^{\mathcal{N}}_{bool}$.

Definition 4.1 A TBox $\mathcal{T}$ in DL-Lite$^{\mathcal{N}}_{bool}$ is in *normal form* if all of its inclusions are of the form $B_1 \sqcap \ldots \sqcap B_m \sqsubseteq B_{m+1} \sqcup \ldots \sqcup B_n$ where $0 \leq m \leq n$ and $B_1, \ldots, B_n$ are basic concepts such that $B_i \neq B_j$ for all $i \neq j$.

Algorithm 1 shows that every DL-Lite$^{\mathcal{N}}_{bool}$ TBox can be transformed into an equivalent TBox in normal form.

Algorithm 1 (Transform a DL-Lite$^{\mathcal{N}}_{bool}$ TBox into normal form)
Input: A TBox $\mathcal{T}$ in DL-Lite$^{\mathcal{N}}_{bool}$.
Output: A TBox $\mathcal{T}'$ in normal form.
Method:
Step 1. For each inclusion $C \sqsubseteq D$, replace C with its DNF and replace D with its CNF.
Step 2. For each resulting inclusion $C_1 \sqcup \cdots \sqcup C_m \sqsubseteq D_1 \sqcap \cdots \sqcap D_n$, where each C_i is a conjunction of literal concepts and each D_j is a disjunction of literal concepts, replace the inclusion with the set of inclusions $\{ C_i \sqsubseteq D_j \mid 1 \leq i \leq m, 1 \leq j \leq n \}$.
Step 3. Replace each inclusion $C_i' \sqcap \neg B \sqsubseteq D_j$, where B is a basic concept, with $C_i' \sqsubseteq B \sqcup D_j$ to eliminate the negation. Similarly, replace each $C_i \sqsubseteq \neg B \sqcup D_j'$ with $C_i \sqcap B \sqsubseteq D_j'$.
Step 4. Remove any inclusion with the same concept names appearing on both sides of the inclusion, and return the resulting TBox as $\mathcal{T}'$.

Fig. 1 Transform DL-Lite$^{\mathcal{N}}_{bool}$ TBoxes into normal form

Lemma 4.1 *For any DL-Lite$^{\mathcal{N}}_{bool}$ TBox $\mathcal{T}$, the TBox $\mathcal{T}'$ returned in Algorithm 1 is in normal form and is equivalent to $\mathcal{T}$.*

Proof It is not hard to see that Steps 1, 2 and 4 preserve the equivalence of the inclusions. We only need to show Step 3 transforms the inclusions equivalently.

We want to show that for general concepts C_1, C_2 and C_3, the following two inclusions are equivalent:

$C_1 \sqcap C_2 \sqsubseteq C_3$ and $C_1 \sqsubseteq \neg C_2 \sqcup C_3$.

For each model $\mathcal{I}$ of the first inclusion, we have $C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \subseteq C_3^{\mathcal{I}}$. Thus $C_1^{\mathcal{I}} \subseteq \overline{C_2^{\mathcal{I}}} \cup C_3^{\mathcal{I}}$, where $\overline{C_2^{\mathcal{I}}} = \Delta^{\mathcal{I}} - C_2^{\mathcal{I}}$. That is, $\mathcal{I}$ is also a model of the second inclusion. Similarly we can show that each model of the second inclusion is also a model of the first inclusion.

■

In our algorithm for computing the result of forgetting in a KB, we need also to transform each DL-Lite$^{\mathcal{N}}_{bool}$ ABox into a normal form.

Definition 4.2 A DL-Lite$^{\mathcal{N}}_{bool}$ ABox $\mathcal{A}$ is in *normal form* if $\mathcal{A} = \{C_1(a_1), \ldots, C_s(a_s)\} \cup \mathcal{A}_r$ for some $s \geq 0$ and the following conditions are satisfied:

1. $\mathcal{A}_r$ contains only role assertions,
2. $a_i \neq a_j$ for $1 \leq i < j \leq s$, and
3. C_i is in DNF, for each $1 \leq i \leq s$.

Algorithm 2 explains how to transform a DL-Lite$^{\mathcal{N}}_{bool}$ ABox into normal form.

Note that after ABox $\mathcal{A}$ is transformed into its normal form, each individual a is associated with only one assertion $C(a)$ in $\mathcal{A}$, and C is in DNF.

Lemma 4.2 *Given a DL-Lite$^{\mathcal{N}}_{bool}$ ABox $\mathcal{A}$, the ABox $\mathcal{A}'$ returned in Algorithm 2 is in normal form and is equivalent to $\mathcal{A}$.*

Algorithm 2 (Transform a DL-Lite$^{\mathcal{N}}_{bool}$ ABox into normal form)
Input: An ABox $\mathcal{A}$ in DL-Lite$^{\mathcal{N}}_{bool}$.
Output: An ABox $\mathcal{A}'$ in normal form.
Method:
Step 1. Let $a_1, \ldots, a_s$ be the distinct individual names in $\mathcal{A}$ and let $As(a_i) = \{ C(a_i) \mid C(a_i) \in \mathcal{A} \}$, for $i = 1, \ldots, s$. Replace each set of assertions $As(a_i) = \{C_1(a_i), \ldots, C_n(a_i)\}$, with a single assertion $(C_1 \sqcap \ldots \sqcap C_n)(a_i)$.
Step 2. Transform each assertion $(C_1 \sqcap \cdots \sqcap C_n)(a)$ into its DNF, *i.e.*, $C(a) = (D_1 \sqcup \cdots \sqcup D_m)(a)$, where each D_k is a conjunction of literal concepts, and no conjunction contains both a literal concept and its negation.
Step 3. Return the resulting set of concept assertions $C(a)$ together with the original role assertions in $\mathcal{A}$.

Fig. 2 Transform DL-Lite$^{\mathcal{N}}_{bool}$ ABoxes into normal form

A KB is said to be in *normal form* if both its TBox and ABox are in normal form.

We are now ready to present our algorithm for computing the result of forgetting about a set $\mathcal{S}$ of concept names in a DL-Lite$^{\mathcal{N}}_{bool}$ KB $\mathcal{K}$. The basic idea of Algorithm 3 is to first transform the given KB into its normal form, and after generating all inclusions and assertions that should be included in the result of forgetting, remove all occurrences of concepts in $\mathcal{S}$.

Algorithm 3 (Compute the result of forgetting a set of concept names in a DL-Lite$_{bool}$ KB)
Input: A DL-Lite$_{bool}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and a set $\mathcal{S}$ of concept names.
Output: $\mathsf{forget}(\mathcal{K}, \mathcal{S})$.
Method:
Step 1. Using Algorithms 1 and 2, transform the KB $\mathcal{K}$ into its normal form.
Step 2. For each pair of inclusions $A \sqcap C \sqsubseteq D$ and $C' \sqsubseteq A \sqcup D'$ in $\mathcal{T}$, where $A \in \mathcal{S}$, add inclusion $C \sqcap C' \sqsubseteq D \sqcup D'$ to $\mathcal{T}$ if it contains no concept name A' appearing on both sides of the inclusion.
Step 3. For each concept name $A \in \mathcal{S}$ occurring in $\mathcal{A}$, add $A \sqsubseteq \top$ and $\bot \sqsubseteq A$ to $\mathcal{T}$;
Step 4. For each assertion $C(a)$ in $\mathcal{A}$, each inclusion $A \sqcap D_1 \sqsubseteq D_2$ and each inclusion $D_3 \sqsubseteq A \sqcup D_4$ in $\mathcal{T}$, where $A \in \mathcal{S}$, add $C'(a)$ to $\mathcal{A}$, where C' is obtained by replacing each occurrence of A in C with $\neg D_1 \sqcup D_2$ and $\neg A$ with $\neg D_3 \sqcup D_4$, and C' is transformed into its DNF.
Step 5. Remove all inclusions of the form $C \sqsubseteq \top$ or $\bot \sqsubseteq C$, and all assertions of the form $\top(a)$.
Step 6. Remove all inclusions and assertions that contain any concept name in $\mathcal{S}$.
Step 7. Return the resulting KB as $\mathsf{forget}(\mathcal{K}, \mathcal{S})$.

Fig. 3 Forget concepts in a DL-Lite$_{bool}$ KB.

In Algorithm 3, Step 1 transforms the input KB into normal form. For each concept name A, Step 2 forgets about A from TBox inclusions in a resolution-like manner. However, the original inclusions are not discarded immediately because they will be used in performing forgetting in the ABox. After Step 2, all the inclusions to be included in the result of forgetting have been generated. Step 3 is to make the specification of Step 4 simpler, by ensuring that the subsumer and subsumee of A is explicitly stated. In Step 4, each positive occurrence of A in the ABox is replaced by its subsumer, and each negative occurrence by its subsumee. Note that in each assertion $C(a)$, C is always in DNF. Finally, Step 5 eliminates redundant inclusions and assertions, and Step 6 removes the original inclusions and assertions containing concept names in $\mathcal{S}$.

Algorithm 3 is demonstrated in the following example.

Example 4.1 (Cont. of Example 2.1) To compute $\mathsf{forget}(\mathcal{K}, \{Professor\})$, note that $\mathcal{K}$ is already in its normal form. In Step 2, two new inclusions are generated,

$Researcher \sqsubseteq \geqslant 5\ hasPublications \sqcup RA$ and
$Researcher \sqcap RA \sqsubseteq RA$.

The second inclusion contains concept RA on both sides and is not added to the TBox.

Then, in Step 4, two assertions $\geqslant 5\ hasPublications(John)$ and $\neg RA(John)$ are added into $\mathcal{A}$.

After all inclusions and assertions containing concept $Professor$ are removed in Step 6, the algorithm returns the KB in Example 3.1 as $\mathsf{forget}(\mathcal{K}, \{Professor\})$.

It is easy to see that Algorithm 3 always terminates. In the worst case, the algorithm is exponential in time. However, the exponential blow up is introduced only by Step 1, where the given KB is transformed into its normal form. Note that the transformations in Steps 2 and 4 take only polynomial time. If the input KB is in normal form, Algorithm 3 takes only polynomial time to compute the result of forgetting.

Algorithm 3 is sound and complete with respect to the semantic definition of forgetting.

Theorem 4.1 *Let $\mathcal{K}$ be a DL-Lite$^{\mathcal{N}}_{bool}$ KB and $\mathcal{S}$ a set of concept names. Then Algorithm 3 always returns* $\mathsf{forget}(\mathcal{K}, \mathcal{S})$.

Suppose the size of $\mathcal{S}$ is fixed. When the input $\mathcal{K}$ is in normal form, the time complexity of Algorithm 3 is $O(|\mathcal{K}|^3)$.

Before proving Theorem 4.1, we first show the following lemma.

Lemma 4.3 *In Algorithm 3, Steps 2 – 5 are equivalence-preserving transformations for KBs.*

Proof **Step 2**: We want to show that each inclusion added in this step is a logical consequence of $\mathcal{T}$. As shown in the proof of Lemma 4.1, each $A \sqcap C \sqsubseteq D$ is equivalent to $A \sqsubseteq \neg C \sqcup D$, and each $C' \sqsubseteq A \sqcup D'$ to $C' \sqcap \neg D' \sqsubseteq A$. Thus each new inclusion $C \sqcap C' \sqsubseteq D \sqcup D$ added, which is equivalent to $C' \sqcap \neg D' \sqsubseteq \neg C \sqcup D$, is a logical consequence of $\mathcal{T}$. Note that any inclusion containing concept name A' on both sides is a tautology inclusion. Thus we have shown that Step 2 is a equivalence-preserving transformation.
Step 3: Each inclusion added in this step is a tautology inclusion.
Step 4: Again, each $A \sqcap D_1 \sqsubseteq D_2$ is equivalent to $A \sqsubseteq \neg D_1 \sqcup D_2$ while each $D_3 \sqsubseteq A \sqcup D_4$ is equivalent to $\neg A \sqsubseteq \neg D_3 \sqcup D_4$. For each assertion of the form

$$[\ \bigsqcup (A \sqcap D_i) \sqcup \bigsqcup (\neg A \sqcap D_j) \sqcup \bigsqcup D_k\ ](a)$$

in $\mathcal{A}$, where each D with subscript is a conjunction of literal concepts and does not contain A, the new assertion added in Step 4,

$$[\ \bigsqcup \big((\neg D_1 \sqcup D_2) \sqcap D_i\big) \sqcup \bigsqcup \big((\neg D_3 \sqcup D_4) \sqcap D_j\big) \sqcup \bigsqcup D_k\ ](a),$$

is a logical consequence of $\mathcal{K}$.
Step 5: Each inclusion or assertion removed in this step is a tautology inclusion or assertion. ■

With Lemma 4.1, 4.2 and 4.3, we are now ready to present the proof of Theorem 4.1.
Proof of Theorem 4.1 By Lemma 4.1 and 4.2, Step 1 transforms a KB into an equivalent KB. By Lemma 4.3, each of Steps 2 – 5 also transforms a KB into an equivalent KB.

1 KW: Rodney, is ”order-insensitive” a right word here?

[1]Note that in Step 6, the removal of inclusions and assertions is order-insensitive. Without loss of generality, we assume:

- the inclusions and assertions that are added in Step 2 – 4 are firstly removed;
- suppose $\mathcal{S} = \{A_1, \ldots, A_n\}$, then the inclusions and assertions containing A_i are removed before those containing A_{i+1} for $i = 1, \ldots, n-1$;
- assertions are removed first, followed by inclusions of the form $A \sqcap C \sqsubseteq D$, and then of the form $C \sqsubseteq A \sqcup D$.

Let $\mathcal{K}_0$ be the KB obtained in Step 6 after removing all inclusions and assertions containing concepts in $\mathcal{S}$ that are added in Steps 2 – 4. Denote $\mathcal{K}_{i+1}$ to be the resulting KB obtained from $\mathcal{K}_i$ by removing one assertion or inclusion containing some concept name $A \in \mathcal{S}$. To prove the result returned in Step 7 is $\mathsf{forget}(\mathcal{K}, \mathcal{S})$, by Proposition 3.1, we only need to show that for each $i > 0$ and each model $\mathcal{I}'$ of $\mathcal{K}_{i+1}$, there always exists a model $\mathcal{I}$ of $\mathcal{K}_i$ s.t. $\mathcal{I} \sim_{\{A\}} \mathcal{I}'$.

Consider the following three cases:

Case 1. Consider removing assertion $\left[\ \bigsqcup(A \sqcap D_i) \sqcup \bigsqcup(\neg A \sqcap D_j) \sqcup \bigsqcup D_k\ \right](a)$, which is equivalent to $\left[\ (A \sqcap C_1) \sqcup (\neg A \sqcap C_2) \sqcup C_3\ \right](a)$ where $C_1 = \bigsqcup D_i$, $C_2 = \bigsqcup D_j$ and $C_3 = \bigsqcup D_k$:

For each model $\mathcal{I}'$ of $\mathcal{K}_{i+1}$, denote $\varGamma = \bigcap_{(A \sqcap D_1 \sqsubseteq D_2) \in \mathcal{K}_{i+1}} (\neg D_1 \sqcup D_2)^{\mathcal{I}'}$ and $\varGamma' = \bigcap_{(D_3 \sqsubseteq A \sqcup D_4) \in \mathcal{K}_{i+1}} (\neg D_3 \sqcup D_4)^{\mathcal{I}'}$. Then $\overline{\varGamma'} \subseteq A^{\mathcal{I}'} \subseteq \varGamma$. Construct $\mathcal{I}$ such that $\mathcal{I} \sim_{\{A\}} \mathcal{I}'$ and satisfies one of the following three conditions:

$$
\begin{aligned}
&(1)\ A^{\mathcal{I}} = A^{\mathcal{I}'} \cup \{a^{\mathcal{I}'}\},\ \text{if } a^{\mathcal{I}'} \in (\varGamma - A^{\mathcal{I}'}) \cap (C_1^{\mathcal{I}'} - C_2^{\mathcal{I}'});\\
&(2)\ A^{\mathcal{I}} = A^{\mathcal{I}'} - \{a^{\mathcal{I}'}\},\ \text{if } a^{\mathcal{I}'} \in A^{\mathcal{I}'} \cap \varGamma' \cap (C_2^{\mathcal{I}'} - C_1^{\mathcal{I}'});\\
&(3)\ A^{\mathcal{I}} = A^{\mathcal{I}'} \text{ otherwise.}
\end{aligned}
$$

To show that $\mathcal{I} \models \mathcal{K}_i$, we only need to show that $\mathcal{I}$ satisfies each assertion or inclusion in $\mathcal{K}_i$ containing A.

Consider the removed assertion $\left[\ (A \sqcap C_1) \sqcup (\neg A \sqcap C_2) \sqcup C_3\ \right](a)$, and we want to show that

$$a^{\mathcal{I}} \in (A^{\mathcal{I}} \cap C_1^{\mathcal{I}}) \cup (\overline{A^{\mathcal{I}}} \cap C_2^{\mathcal{I}}) \cup C_3^{\mathcal{I}}. \qquad (*)$$

For each inclusion $A \sqcap D_1 \sqsubseteq D_2$ and each inclusion $D_3 \sqsubseteq A \sqcup D_4$ in $\mathcal{K}_{i+1}$, assertion

$$\left[\ \big((\neg D_1 \sqcup D_2) \sqcap C_1\big) \sqcup \big((\neg D_3 \sqcup D_4) \sqcap C_2\big) \sqcup C_3\ \right](a)$$

has been added into the KB in Step 4. These assertions are still in $\mathcal{K}_{i+1}$, and thus satisfied by $\mathcal{I}'$. Combining all such assertions, we have

$$a^{\mathcal{I}'} \in (\varGamma \cap C_1^{\mathcal{I}'}) \cup (\varGamma' \cap C_2^{\mathcal{I}'}) \cup C_3^{\mathcal{I}'}.$$

In case (1), we have $a^{\mathcal{I}'} \in A^{\mathcal{I}} \cap C_1^{\mathcal{I}'}$, thus $(*)$ holds. In case (2), we have $a^{\mathcal{I}'} \in \overline{A^{\mathcal{I}}} \cap C_2^{\mathcal{I}'}$, and $(*)$ still holds. In case (3), we have either $a^{\mathcal{I}'} \in A^{\mathcal{I}'} \cap (C_1^{\mathcal{I}'} - C_2^{\mathcal{I}'})$, or $a^{\mathcal{I}'} \in \overline{A^{\mathcal{I}'}} \cap (C_2^{\mathcal{I}'} - C_1^{\mathcal{I}'})$, or $a^{\mathcal{I}'} \in C_1^{\mathcal{I}'} \cap C_2^{\mathcal{I}'}$. That is, $(*)$ always holds.

As all the assertions added in Steps 2 – 4 are removed first, $\left[\ (A \sqcap C_1) \sqcup (\neg A \sqcap C_2) \sqcup C_3\ \right](a)$ is the only assertion about a in $\mathcal{K}_i$. Obviously, $\mathcal{I}$ also satisfies the other assertions in $\mathcal{K}_i$.

Now we consider inclusions containing A in $\mathcal{K}_i$. For each inclusion $A \sqcap D_1 \sqsubseteq D_2$, since it is also in $\mathcal{K}_{i+1}$, we have $A^{\mathcal{I}'} \subseteq \overline{D_1^{\mathcal{I}'}} \cup D_2^{\mathcal{I}'}$. In case (1), from $\varGamma \subseteq \overline{D_1^{\mathcal{I}'}} \cup D_2^{\mathcal{I}'}$, we have $a^{\mathcal{I}'} \in \overline{D_1^{\mathcal{I}'}} \cup D_2^{\mathcal{I}'}$, and thus $A^{\mathcal{I}} \subseteq \overline{D_1^{\mathcal{I}}} \cup D_2^{\mathcal{I}}$. In case (2) or (3), obviously, $A^{\mathcal{I}} \subseteq \overline{D_1^{\mathcal{I}}} \cup D_2^{\mathcal{I}}$. That is, $\mathcal{I}$ satisfies $A \sqcap D_1 \sqsubseteq D_2$. For each inclusion $D_3 \sqsubseteq A \sqcup D_4$ in $\mathcal{K}$, it can be shown in a similar way that $\mathcal{I}$ also satisfies $D_3 \sqsubseteq A \sqcup D_4$.

We have shown that $\mathcal{I} \models \mathcal{K}_i$.

Case 2. Consider removing inclusion $A \sqcap C \sqsubseteq D$: For each model $\mathcal{I}'$ of $\mathcal{K}_{i+1}$, construct $\mathcal{I}$ such that $\mathcal{I} \sim_{\{A\}} \mathcal{I}'$ and $A^{\mathcal{I}} = A^{\mathcal{I}'} \cap (\neg C \sqcup D)^{\mathcal{I}'}$. Since assertions containing A are removed first, we only need to show that $\mathcal{I}$ satisfies each inclusion in $\mathcal{K}_i$ containing A.

Obviously, $\mathcal{I}$ satisfies $A \sqcap C \sqsubseteq D$ and all the other inclusions in $\mathcal{K}_i$ of the form $A \sqcap D_1 \sqsubseteq D_2$. For each inclusion in $\mathcal{K}_i$ of the form $D_3 \sqsubseteq A \sqcup D_4$, it is also in $\mathcal{K}_{i+1}$ and we have $D_3^{\mathcal{I}'} \subseteq A^{\mathcal{I}'} \cup D_4^{\mathcal{I}'}$. Also, inclusion $C \sqcap D_3 \sqsubseteq D \sqcup D_4$ has been added into the KB in Step 2 and is still in $\mathcal{K}_i$. Thus, we have $D_3^{\mathcal{I}'} \subseteq \overline{C^{\mathcal{I}'}} \cup D^{\mathcal{I}'} \cup D_4^{\mathcal{I}'}$. Combining these two facts, we have $D_3^{\mathcal{I}'} \subseteq (A^{\mathcal{I}'} \cap (\neg C \sqcup D)^{\mathcal{I}'}) \cup D_4^{\mathcal{I}'}$, which is $D_3^{\mathcal{I}} \subseteq A^{\mathcal{I}} \cup D_4^{\mathcal{I}}$. That is, $\mathcal{I}$ satisfies $D_3 \sqsubseteq A \sqcup D_4$.

We have shown that $\mathcal{I} \models \mathcal{K}_i$.

Case 3. Consider removing inclusion $C \sqsubseteq A \sqcup D$: For each model $\mathcal{I}'$ of $\mathcal{K}_{i+1}$, construct $\mathcal{I}$ such that $\mathcal{I} \sim_{\{A\}} \mathcal{I}'$ and $A^{\mathcal{I}} = A^{\mathcal{I}'} \cup (C \sqcap \neg D)^{\mathcal{I}'}$. Obviously, $\mathcal{I}$ satisfies $C \sqsubseteq A \sqcup D$ and all the other inclusions in $\mathcal{K}_i$ of the form $C' \sqsubseteq A \sqcap D'$. And, again, we have shown that $\mathcal{I} \models \mathcal{K}_i$.

Based on the discussions of Cases 1 – 3, we can draw the conclusion that the KB returned in Step 7 is $\mathsf{forget}(\mathcal{K}, \mathcal{S})$.

Last, we show that the computational complexity of this algorithm is in polynomial time when the input $\mathcal{K}$ is in normal form. Let n be the size of $\mathcal{K}$.

Since the size of $\mathcal{S}$ is a constant, without loss of generality, we assume that $\mathcal{S}$ contains only one concept name A. In Step 2, at most n^2 pairs of inclusions are considered. In Step 4, at most n^3 triples of assertion and inclusions are considered. Moreover, it is in linear time to transform the resulting concept description C'. The other steps except for Step 1 are obviously in linear time. Therefore, the time complexity of Algorithm 3 is $O(|\mathcal{K}|^3)$ when the input is in normal form.

■

If the original KB is in DL-Lite$^{\mathcal{N}}_{horn}$, then the KB is already in normal form, and we can apply Steps 2 – 7 of Algorithm 3 directly to the KB. Note that the new inclusions added in Step 2 are still inclusions in DL-Lite$^{\mathcal{N}}_{horn}$. But, after Step 4 is done, the resulting ABox may not be in DL-Lite$^{\mathcal{N}}_{horn}$. However, when the ABox is empty, the result of forgetting is always in DL-Lite$^{\mathcal{N}}_{horn}$.

Theorem 4.2 *Let $\mathcal{K}$ be a DL-Lite$^{\mathcal{N}}_{horn}$ KB s.t. $\mathcal{K} = \langle \mathcal{T}, \emptyset \rangle$, and $\mathcal{S}$ be a set of concept names. Then* $\mathsf{forget}(\mathcal{K}, \mathcal{S})$ *can be computed by Algorithm 3 in polynomial time and the result is in DL-Lite$^{\mathcal{N}}_{horn}$.*

Proof We only need to show that the result is in DL-Lite$^{\mathcal{N}}_{horn}$. It is enough to note that in Step 2, for each pair $A \sqcap D \sqsubseteq B$ and $D' \sqsubseteq A$, where B is a basic concept and D, D' are conjunctions of basic concepts, the new inclusion added, $D \sqcap D' \sqsubseteq B$, is in DL-Lite$^{\mathcal{N}}_{horn}$.

■

Theorems 3.1 and 3.2 are direct consequences of the above two theorems.

5 Query-Based Forgetting for DL-Lite Knowledge Bases

As mentioned in Section 1, three forms of forgetting have been proposed for DL-Lite TBoxes in the literature, which are complementary to each other. It is also argued in [21] that practical application domains may need different definitions of forgetting. In the setting of DL-Lite KBs, we have the same requirement from ontology applications (*i.e.*, various forms of forgetting might co-exist). A natural question arises: Can we establish a unifying framework for defining and comparing various definitions of forgetting for DL-Lite KBs? To this end, we introduce a hierarchy of forgetting for DL-Lite$^{\mathcal{N}}_{bool}$ KBs, which can be used as a unifying framework for forgetting in DL-Lite$^{\mathcal{N}}_{bool}$ KBs. In particular, we show that three forms of forgetting for DL-Lite$^{\mathcal{N}}_{bool}$ KBs can be defined/embedded in our framework (two are natural generalizations of those two forgettings for TBoxes in [21], while the other one is the model-based forgetting defined in Definition 3.1)

As the DL-Lite family is especially designed for efficient query answering, our hierarchy of forgetting is defined in terms of preserving query answering.

para

$\boxed{2}$

5.1 Definitions and Basic Properties

The intuition behind our query-based forgetting is based on the following conditions that are naturally obtained from the informal description of forgetting described earlier. Specifically, the result of forgetting about a signature $\mathcal{S}$ in $\mathcal{K}$ should be a KB $\mathcal{K}'$ such that (1) $\mathcal{K}'$ does not contain new concepts or roles, or any occurrence of concept or role name in $\mathcal{S}$, (2) $\mathcal{K}'$ is weaker than $\mathcal{K}$, and (3) $\mathcal{K}$ and $\mathcal{K}'$ give the same answers to all queries that is irrelevant to $\mathcal{S}$ in a given query language.

In this section, we assume that $\mathcal{Q}$ is a query language for DL-Lite$^{\mathcal{N}}_{bool}$ and in particular, specifies an inference relation $\mathcal{K} \models q$ for every KB $\mathcal{K}$ in DL-Lite$^{\mathcal{N}}_{bool}$ and every query q in $\mathcal{Q}$. Note that the notion of query is very general here. A query can be an assertion, an inclusion, or even a formula in a logic language such as the first order logic.

Definition 5.1 (query-based forgetting) Let $\mathcal{K}$ be an $\mathcal{L}$-KB, $\mathcal{S}$ be a signature and $\mathcal{Q}$ be a query language for $\mathcal{L}$. A KB $\mathcal{K}'$ is a *result of $\mathcal{Q}$-forgetting about $\mathcal{S}$ in $\mathcal{K}$* if the following three conditions are satisfied:

- $\mathsf{Sig}(\mathcal{K}') \subseteq \mathsf{Sig}(\mathcal{K}) - \mathcal{S}$,
- $\mathcal{K} \models \mathcal{K}'$, and
- $\mathcal{K} \models q$ implies $\mathcal{K}' \models q$, for any grounded query q in $\mathcal{Q}$ with $\mathsf{Sig}(q) \cap \mathcal{S} = \emptyset$.

Definition 5.1 generalizes the b-forgetting and u-forgetting in [21] in at least two ways: 1) it is defined for KBs rather only TBoxes; 2) it is a parameterized definition of forgetting in the sense that each query language determines a definition of forgetting.

In Section 5.2, we will identify three interesting query languages and thus three query-based forgettings are defined. By proving that one of them coincides with the model-based forgetting, we show that the model-based forgetting can be embedded in our parameterized framework. We remark that Definition 5.1 applies to any other DL languages.

One thing worth mentioning here is that $\mathcal{K}'$ does not necessarily *query entail* $\mathcal{K}$ [2], as query entailment requires an arbitrary ABox. This also shows the difference of $\mathcal{Q}$-forgetting for KBs and u-forgetting for TBoxes (Definition 20 in [21]).

The results of $\mathcal{Q}$-forgetting are not necessarily unique up to KB equivalence in general. An example of this is given as follows. Suppose we take $\mathcal{Q}$ as the set of all inclusions. Given a consistent $\mathcal{L}$-KB $\mathcal{K}$, if $\mathcal{L}$-KB $\langle \mathcal{T}', \mathcal{A}' \rangle$ is a result of $\mathcal{Q}$-forgetting in $\mathcal{K}$, then for any subset

[2] query entailment is defined in [21], TBox $\mathcal{T}'$ query entails $\mathcal{T}$ iff for any ABox $\mathcal{A}$ and query q, $\langle \mathcal{T}, \mathcal{A} \rangle \models q$ implies $\langle \mathcal{T}', \mathcal{A} \rangle \models q$.

$\mathcal{A}''$ of $\mathcal{A}'$, $\langle \mathcal{T}', \mathcal{A}'' \rangle$ is also a result of $\mathcal{Q}$-forgetting in $\mathcal{K}$. This is because all the inclusion consequences of a consistent $\mathcal{L}$-KB is entailed from its TBox.

We will denote the set of all results of $\mathcal{Q}$-forgetting about $\mathcal{S}$ in $\mathcal{K}$ as $\mathsf{Forget}^{\mathcal{Q}}(\mathcal{K}, \mathcal{S})$.

The model-based forgetting requires preserving model equivalence and thus the result of model-based forgetting is also a result of $\mathcal{Q}$-forgetting for any query language $\mathcal{Q}$. In this sense, the model-based forgetting is the strongest notion of forgetting for DL-Lite$^{\mathcal{N}}_{bool}$.

Theorem 5.1 *Let $\mathcal{K}$ be an $\mathcal{L}$-KB, $\mathcal{S}$ be a signature and $\mathcal{Q}$ be a query language for $\mathcal{L}$. Then*

1. $\mathsf{forget}(\mathcal{K}, \mathcal{S}) \in \mathsf{Forget}^{\mathcal{Q}}(\mathcal{K}, \mathcal{S})$*, and*
2. *for each* $\mathcal{K}' \in \mathsf{Forget}^{\mathcal{Q}}(\mathcal{K}, \mathcal{S})$*, we have* $\mathsf{forget}(\mathcal{K}, \mathcal{S}) \models \mathcal{K}'$.

Proof 1. First, note that $\mathcal{K} \models \mathsf{forget}(\mathcal{K}, \mathcal{S})$. For any grounded query q such that $\mathsf{Sig}(q) \cap \mathcal{S} = \emptyset$, we need only to show that $\mathcal{K} \models q$ implies $\mathsf{forget}(\mathcal{K}, \mathcal{S}) \models q$.

In fact, for each model $\mathcal{I}'$ of $\mathsf{forget}(\mathcal{K}, \mathcal{S})$, by the definition of model-based forgetting, there exists a model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \sim_{\mathcal{S}} \mathcal{I}'$. If $\mathcal{K} \models q$, then $\mathcal{I} \models q$. However, notice that $\mathcal{I}'$ and $\mathcal{I}$ coincide on $\mathsf{Sig}(\mathcal{L}) - \mathcal{S}$ and q does not contain any symbol in $\mathcal{S}$. Thus $\mathcal{I}' \models q$.

2. Similarly, we can show that for any KB $\mathcal{K}'$ over $\mathsf{Sig}(\mathcal{K}) - \mathcal{S}$, $\mathcal{K} \models \mathcal{K}'$ implies $\mathsf{forget}(\mathcal{K}, \mathcal{S}) \models \mathcal{K}'$. ■

This theorem shows that Algorithm 3 can also be used to compute a result of $\mathcal{Q}$-forgetting.

In general, a larger query language defines a stronger notion of query-based forgetting.

Proposition 5.1 *Let $\mathcal{K}$ be an $\mathcal{L}$-KB and $\mathcal{S}$ be a signature. If $\mathcal{Q}$ and $\mathcal{Q}'$ are two query languages for $\mathcal{L}$ such that $\mathcal{Q}' \subseteq \mathcal{Q}$, then* $\mathsf{Forget}^{\mathcal{Q}}(\mathcal{K}, \mathcal{S}) \subseteq \mathsf{Forget}^{\mathcal{Q}'}(\mathcal{K}, \mathcal{S})$.

Proof For each $\mathcal{K}' \in \mathsf{Forget}^{\mathcal{Q}}(\mathcal{K}, \mathcal{S})$, we have for any grounded query $q \in \mathcal{O}'$ with $\mathsf{Sig}(q) \cap \mathcal{S} = \emptyset$, $\mathcal{K} \models q$ implies $\mathcal{K}' \models q$. That is $\mathcal{K}' \in \mathsf{Forget}^{\mathcal{Q}'}(\mathcal{K}, \mathcal{S})$. ■

5.2 Specific Query-Based Forgetting

In what follows, we will examine three interesting query languages for DL-Lite$^{\mathcal{N}}_{bool}$ and the corresponding notions of query-based forgetting. We will also show how model-based forgetting is characterized by query-based forgetting. Proofs of most results in this subsection are given in the next subsection.

We note that several results in this section generalize the corresponding results in [21] but proofs of these results show that the generalizations from TBoxes to KBs are highly non-trivial as we can see in the next subsection.

The first choice of $\mathcal{Q}$ is the set of concept inclusions $C_1 \sqsubseteq C_2$, assertions $C(a)$ and $R(a, b)$ in the DL-Lite$^{\mathcal{N}}_{bool}$ language $\mathcal{L}$, denoted $\mathcal{Q}_{\mathcal{L}}$. We can see that $\mathcal{Q}_{\mathcal{L}}$-forgetting for KBs extends the b-forgetting for TBoxes in [21] (note that assertions are not needed for TBoxes in their case).

Example 5.1 (Cont. of Example 2.1) One result of $\mathcal{Q}_{\mathcal{L}}$-forgetting about role name $hasPublications$ in $\mathcal{K}$, denoted $\mathcal{K}'$, consists of the following inclusions and assertions:

$Professor \sqsubseteq Researcher$,
$Researcher \sqsubseteq Professor \sqcup RA$,
$Professor \sqcap RA \sqsubseteq \bot$,
$Professor(John)$, and $Paper(P75)$.

$\mathcal{Q}_\mathcal{L}$-forgetting possesses most desirable properties that hold for model-based forgetting. Before presenting these properties, we first note the following lemma.

Lemma 5.1 *Let $\mathcal{K}$ and $\mathcal{K}'$ be two $\mathcal{L}$-KBs. Then $\mathcal{K} \equiv \mathcal{K}'$ iff for every $q \in \mathcal{Q}_\mathcal{L}$, $\mathcal{K} \models q$ iff $\mathcal{K}' \models q$.*

Proof The "only if" direction is obvious. We only need to show the "if" direction. For each inclusion or assertion α in $\mathcal{K}'$, $\alpha \in \mathcal{Q}_\mathcal{L}$ and thus $\mathcal{K}' \models \alpha$, which implies $\mathcal{K} \models \alpha$ for every $\alpha \in \mathcal{K}'$. That is, $\mathcal{K} \models \mathcal{K}'$. Similarly, $\mathcal{K}' \models \mathcal{K}$. ■

KB Implication holds for $\mathcal{Q}_\mathcal{L}$-forgetting, as long as the results are expressible in DL-Lite$^\mathcal{N}_{bool}$. This can be seen from the fact that, given $\mathcal{K}_1 \models \mathcal{K}_2$ and the results $\mathcal{K}_1', \mathcal{K}_2'$ of $\mathcal{Q}_\mathcal{L}$-forgetting about $\mathcal{S}$ in, respectively, $\mathcal{K}_1$ and $\mathcal{K}_2$, if $\mathcal{K}_1', \mathcal{K}_2'$ are both expressible in DL-Lite$^\mathcal{N}_{bool}$, then for any inclusion or assertion α in $\mathcal{K}_2'$, we have by the definition of $\mathcal{Q}_\mathcal{L}$-forgetting, $\mathcal{K}_1' \models \alpha$.

Uniqueness holds for $\mathcal{Q}_\mathcal{L}$-forgetting, as a direct consequence of **KB Implication**. That is, the result of $\mathcal{Q}_\mathcal{L}$-forgetting is unique in DL-Lite$^\mathcal{N}_{bool}$, up to KB equivalence. However, as we will show later, when more expressive languages are considered (*e.g.*, DL-Lite$^u_{bool}$), the uniqueness of results may not hold anymore.

Also, it follows directly from the definition that $\mathcal{Q}_\mathcal{L}$-forgetting satisfies **Consistency**, **Coherence** and **Consequence Invariance**. We recall that a KB $\mathcal{K}$ is inconsistent if and only if $\mathcal{K} \models (\top \sqsubseteq \bot)$, and $\mathcal{K}$ is incoherent over $\mathcal{S}$ if and only if for some concept name $A \in \mathcal{S}$, $\mathcal{K} \models (A \sqsubseteq \bot)$, or for some role name $P \in \mathcal{S}$, $\mathcal{K} \models (\exists P \sqsubseteq \bot)$.

Another important property of $\mathcal{Q}_\mathcal{L}$-forgetting is **Signature Union**.

However, $\mathcal{Q}_\mathcal{L}$-forgetting does not possess **PEQ Invariance** and **KB Union** in general. This can be seen from the following example. 3

Example 5.2 Consider a knowledge base $\mathcal{K}_0 = \langle \mathcal{T}_0, \mathcal{A}_0 \rangle$ where

$\mathcal{T}_0 = \{\mathit{Professor} \sqsubseteq \geqslant 5\ \mathit{hasPublications},\ \exists \mathit{hasPublications}^- \sqsubseteq \mathit{Paper}\}$ and

$\mathcal{A}_0 = \{\mathit{Professor}(\mathit{John})\}$.

Then KB $\mathcal{K}_0' = \langle \emptyset, \mathcal{A}_0 \rangle$ is a result of $\mathcal{Q}_\mathcal{L}$-forgetting about role name $\mathit{hasPublications}$ in $\mathcal{K}_0$. Thus we can see that

(1) For PEQ q of the form $\exists x.\mathit{Paper}(x)$, we have $\mathcal{K}_0 \models q$ but $\mathcal{K}_0' \not\models q$. That is, **PEQ Invariance** does not hold.

(2) Let $\mathcal{K}_1 = \langle \{\mathit{Paper} \sqsubseteq \bot\}, \emptyset \rangle$. Then $\mathcal{K}_0 \cup \mathcal{K}_1$ is inconsistent, and thus $\mathcal{K}_0 \cup \mathcal{K}_1 \models (\top \sqsubseteq \bot)$, but $\mathcal{K}_0' \cup \mathcal{K}_1 \not\models (\top \sqsubseteq \bot)$. That is, $\mathcal{K}_0' \cup \mathcal{K}_1$ is not a result of $\mathcal{Q}_\mathcal{L}$-forgetting about role $\mathit{hasPublications}$ in $\mathcal{K}_0 \cup \mathcal{K}_1$. It shows that **KB Union** does not hold either.

An advantage of $\mathcal{Q}_\mathcal{L}$-forgetting is that it possesses nice *existence* property, that is, there always exists a result of $\mathcal{Q}_\mathcal{L}$-forgetting that is expressible in DL-Lite$^\mathcal{N}_{bool}$.

Theorem 5.2 (Existence) *Let $\mathcal{K}$ be an $\mathcal{L}$-KB and $\mathcal{S}$ a signature. Then there always exists a DL-Lite$^\mathcal{N}_{bool}$ KB $\mathcal{K}'$ such that $\mathcal{K}'$ is a result of $\mathcal{Q}_\mathcal{L}$-forgetting about $\mathcal{S}$ in $\mathcal{K}$.*

The above result generalizes Theorem 18 in [21] which is stated only for TBoxes.

To obtain a more expressive form of forgetting, an extension of DL-Lite$^\mathcal{N}_{bool}$ named DL-Lite$^u_{bool}$ is introduced in [21], which extends DL-Lite$^\mathcal{N}_{bool}$ by introducing new concepts of the form $\exists u.C$, where C is a concept in DL-Lite$^\mathcal{N}_{bool}$. Given an interpretation $\mathcal{I}$, $(\exists u.C)^\mathcal{I} = \Delta^\mathcal{I}$ if $C^\mathcal{I} \neq \emptyset$ and $(\exists u.C)^\mathcal{I} = \emptyset$ if $C^\mathcal{I} = \emptyset$. Informally, $\exists u.C$ is used to represent the fact that concept C is nonempty.

Define $\mathcal{Q}^u_{\mathcal{L}}$ to be the query language extending $\mathcal{Q}_{\mathcal{L}}$ with inclusions $C_1 \sqsubseteq C_2$ and assertions $C_3(a)$, where C_i's are concepts in DL-Lite$^u_{bool}$, negated role assertions $\neg R(a,b)$, and unions $\bigvee q_i$ of queries q_i in $\mathcal{Q}^u_{\mathcal{L}}$ [3].

$\mathcal{Q}^u_{\mathcal{L}}$-forgetting generalizes the u-forgetting for TBoxes introduced in Definition 20 of [21], and is logically stronger than $\mathcal{Q}_{\mathcal{L}}$-forgetting.

As with $\mathcal{Q}_{\mathcal{L}}$-forgetting, $\mathcal{Q}^u_{\mathcal{L}}$-forgetting satisfies **KB Implication** when the results are expressible in DL-Lite$^u_{bool}$, and the result of $\mathcal{Q}^u_{\mathcal{L}}$-forgetting is **Unique** within DL-Lite$^u_{bool}$, up to KB equivalence. Also, $\mathcal{Q}^u_{\mathcal{L}}$-forgetting satisfies **Consistency**, **Coherence**, **Consequence Invariance**, and **Signature Union**.

In contrast to $\mathcal{Q}_{\mathcal{L}}$-forgetting, $\mathcal{Q}^u_{\mathcal{L}}$-forgetting satisfies **PEQ Invariance** and **KB Union**.

Proposition 5.2 *Let $\mathcal{K}$ be an $\mathcal{L}$-KB and $\mathcal{S}$ a signature. Suppose $\mathcal{K}'$ is a result of $\mathcal{Q}^u_{\mathcal{L}}$-forgetting about $\mathcal{S}$ in $\mathcal{K}$, then for any grounded PEQ q with $\mathsf{Sig}(q) \cap \mathcal{S} = \emptyset$, $\mathcal{K}' \models q$ iff $\mathcal{K} \models q$.*

Proposition 5.3 *Let $\mathcal{K}_1, \mathcal{K}_2$ be two $\mathcal{L}$-KBs and $\mathcal{S}$ a signature, satisfying $\mathsf{Sig}(\mathcal{K}_1) \cap \mathsf{Sig}(\mathcal{K}_2) \cap \mathcal{S} = \emptyset$. Suppose $\mathcal{K}'_1$ and $\mathcal{K}'_2$ are results of $\mathcal{Q}^u_{\mathcal{L}}$-forgetting about $\mathcal{S}$ in, respectively, $\mathcal{K}_1$ and $\mathcal{K}_2$. Then $\mathcal{K}'_1 \cup \mathcal{K}'_2$ is a result of $\mathcal{Q}^u_{\mathcal{L}}$-forgetting about $\mathcal{S}$ in $\mathcal{K}_1 \cup \mathcal{K}_2$.*

While possessing similar properties to model-based forgetting, $\mathcal{Q}^u_{\mathcal{L}}$-forgetting is not equivalent to model-based forgetting. It is still a logically weaker notion than model-based forgetting. This means, on the one hand, $\mathcal{Q}^u_{\mathcal{L}}$-forgetting may possess better expressibility properties than model-based forgetting. On the other hand, it suffers information loss when $\mathcal{Q}^u_{\mathcal{L}}$-forgetting is performed. This can be seen as follows: the KB $\mathcal{K}'$ in Example 5.1 is also a result of $\mathcal{Q}^u_{\mathcal{L}}$-forgetting about role name $\mathit{hasPublications}$ in $\mathcal{K}$, whereas model-based forgetting is not expressible (as discussed in Example 3.2). The knowledge "the whole center has at least 5 publications" is missing after $\mathcal{Q}^u_{\mathcal{L}}$-forgetting.

Although $\mathcal{Q}^u_{\mathcal{L}}$-forgetting has better expressibility properties than model-based forgetting, similar to the case of TBoxes [21], the results of $\mathcal{Q}^u_{\mathcal{L}}$-forgetting in KBs may not be expressible in DL-Lite$^{\mathcal{N}}_{bool}$ either. Recall the KB $\mathcal{K}_0$ in Example 5.2. $\mathcal{K}_0 \models (\mathit{Professor} \sqsubseteq \exists u.\mathit{Paper})$, *i.e.*, Paper is nonempty whenever $\mathit{Professor}$ is nonempty, but this fact is not expressible in DL-Lite$^{\mathcal{N}}_{bool}$. Note that saying "Paper is nonempty" is different from "Paper is a satisfiable concept" or "$\mathcal{K}_0$ is coherent", as the first statement requires all the models of $\mathcal{K}_0$ to interpret Paper with only nonempty sets, whereas the second and the third statements require at least one model to interpret Paper with a nonempty set.

However, there always exists a result of $\mathcal{Q}^u_{\mathcal{L}}$-forgetting that is expressible in DL-Lite$^u_{bool}$.

Theorem 5.3 (Existence) *Let $\mathcal{K}$ be an $\mathcal{L}$-KB and $\mathcal{S}$ a signature. Then there always exists a DL-Lite$^u_{bool}$ KB $\mathcal{K}'$ such that $\mathcal{K}'$ is a result of $\mathcal{Q}^u_{\mathcal{L}}$-forgetting about $\mathcal{S}$ in $\mathcal{K}$.*

Recall Example 5.2, the result of $\mathcal{Q}^u_{\mathcal{L}}$-forgetting about role name $\mathit{hasPublications}$ in $\mathcal{K}_0$ is KB $\langle \{\mathit{Professor} \sqsubseteq \exists u.\mathit{Paper}\}, \mathcal{A}_0 \rangle$. Note that this KB is also a result of $\mathcal{Q}_{\mathcal{L}}$-forgetting about $\mathit{hasPublications}$ in $\mathcal{K}_0$ (by Proposition 5.1). We have shown in Example 5.2 that $\langle \emptyset, \mathcal{A}_0 \rangle$ is a result of $\mathcal{Q}_{\mathcal{L}}$-forgetting about $\mathit{hasPublications}$ in $\mathcal{K}_0$. It shows that **Uniqueness** holds for $\mathcal{Q}_{\mathcal{L}}$-forgetting only in DL-Lite$^{\mathcal{N}}_{bool}$. Similarly, we can show that **Uniqueness** holds for $\mathcal{Q}^u_{\mathcal{L}}$-forgetting only in DL-Lite$^u_{bool}$.

[3] Union of queries can be informally understood as disjunction of queries. See the definition of *union of conjuctive queries* in [8].

Now we consider how to characterize model-based forgetting by query-based forgetting. As Example 3.2 shows, the results of model-based forgetting may not be expressible in DL-Lite$^{\mathcal{N}}_{bool}$ or DL-Lite$^{u}_{bool}$. The major reason for this is that they do not have a construct to represent the cardinality of a concept.

For this reason, we extend DL-Lite$^{u}_{bool}$ further to DL-Lite$^{c}_{bool}$ by introducing new concepts of the form $\geqslant n\ u.C$, where C is a concept in DL-Lite$^{\mathcal{N}}_{bool}$ and n is a natural number. Given an interpretation $\mathcal{I}$, $(\geqslant n\ u.C)^{\mathcal{I}} = \Delta^{\mathcal{I}}$ if $\sharp(C^{\mathcal{I}}) \geq n$ and $(\geqslant n\ u.C)^{\mathcal{I}} = \emptyset$ if $\sharp(C^{\mathcal{I}}) \leq n-1$.

Define $\mathcal{Q}^{c}_{\mathcal{L}}$ to be the query language extending $\mathcal{Q}^{u}_{\mathcal{L}}$ with inclusions $C_1 \sqsubseteq C_2$ and assertions $C_3(a)$, where C_i's are concepts in DL-Lite$^{c}_{bool}$, and unions $\bigvee q_i$ of queries q_i in $\mathcal{Q}^{c}_{\mathcal{L}}$.

We can show $\mathcal{Q}^{c}_{\mathcal{L}}$-forgetting is equivalent to model-based forgetting.

Theorem 5.4 *Let $\mathcal{K}$ be an $\mathcal{L}$-KB and $\mathcal{S}$ a signature. Then the following two assertions are equivalent:*

(1) $\mathcal{K}'$ is a result of $\mathcal{Q}^{c}_{\mathcal{L}}$-forgetting about $\mathcal{S}$ in $\mathcal{K}$, and
(2) $\mathcal{K}'$ is a result of model-based forgetting about $\mathcal{S}$ in $\mathcal{K}$, i.e., $\mathsf{forget}(\mathcal{K}, \mathcal{S}) = \mathcal{K}'$.

Example 5.3 (Cont. of Example 2.1) The result of model-based forgetting about role name $hasPublications$ in $\mathcal{K}$, *i.e.*,
$\mathsf{forget}(\mathcal{K}, \{hasPublications\})$, consists of the following inclusions and assertions:

$Professor \sqsubseteq Researcher$,
$Researcher \sqsubseteq Professor \sqcup RA$,
$Professor \sqcap RA \sqsubseteq \bot$,
$Professor \sqsubseteq \geqslant 5\ u.Paper$,
$Professor(John)$, and $Paper(P75)$.

However, the results of $\mathcal{Q}^{c}_{\mathcal{L}}$-forgetting in an $\mathcal{L}$-KB may not always expressible in DL-Lite$^{c}_{bool}$. The following example can help us in informally understanding this.

Example 5.4 Let $\mathcal{K}_2$ be an $\mathcal{L}$-KB whose TBox consisting of the following inclusions:

$Professor \sqsubseteq \exists hasID$,
$\exists hasID^{-} \sqsubseteq IDNumber$, and
$IDNumber \sqcap \geqslant 2\ hasID^{-} \sqsubseteq \bot$.

$\mathcal{K}_2$ says that every professor has some ID number, and each ID number can only be associated with no more than one professor. Then if we have n professors, we must have at least n ID numbers.

After forgetting about role name $hasID$ in $\mathcal{K}_2$, the above relation between $Professor$ and $IDNumber$ should still hold. That is, by the definition of model-based forgetting, for any model $\mathcal{I}$ of the result of forgetting, we must have $\sharp(Professor^{\mathcal{I}}) \leq \sharp(IDNumber^{\mathcal{I}})$. This can only be expressed through

$$\geqslant n_x\ u.Professor \sqsubseteq \geqslant n_x\ u.IDNumber$$

with a variable n_x ranging over natural numbers. However, such an expression seems already beyond the expressibility of first order logic.

Table 1 summarizes the properties of the three specific query-based forgettings.

It would be interesting to identify various query languages in terms of computational complexity and requirements from practical applications.

in KBs	*in TBoxes*	**Existence**	**KB Impl.**	**Uniqueness**	**Sign. Union**
$\mathcal{Q}_{\mathcal{L}}$-forgetting	b-forgetting [21]	✓	✓*	✓*	✓
$\mathcal{Q}^u_{\mathcal{L}}$-forgetting	u-forgetting [21]	✓	✓**	✓**	✓
$\mathcal{Q}^c_{\mathcal{L}}$-forgetting	forgetting [34]		✓	✓	✓
	Consistency	**Coherence**	**Cons. Inva.**	**PEQ Inva.**	**KB Union**
$\mathcal{Q}_{\mathcal{L}}$-forgetting	✓	✓	✓		
$\mathcal{Q}^u_{\mathcal{L}}$-forgetting	✓	✓	✓	✓	✓
$\mathcal{Q}^c_{\mathcal{L}}$-forgetting	✓	✓	✓	✓	✓

Table 1 Properties of $\mathcal{Q}$-forgettings ('*' holds only for DL-Lite$^{\mathcal{N}}_{bool}$, and '**' holds only for DL-Lite$^u_{bool}$)

5.3 Type-Based Characterizations of Query-Based Forgetting

In this subsection, based on the notion of types, we first introduce an alternative semantics for DL-Lite$^{\mathcal{N}}_{bool}$ and then present characterizations for the three specific query-based forgettings introduced in last subsection. In turn, these semantic characterizations pave a way for proofs of the results in Section 5.2.

Since DL-Lite$^c_{bool}$ is defined as supremum of both DL-Lite$^{\mathcal{N}}_{bool}$ and DL-Lite$^u_{bool}$, without specification, the KBs mentioned in this subsection are all in DL-Lite$^c_{bool}$, and the model-theoretic characterizations also apply to KBs in DL-Lite$^{\mathcal{N}}_{bool}$ or DL-Lite$^u_{bool}$.

Let $\mathcal{S}$ be a signature and N a set of natural numbers including 1. We call a literal concept over signature $\mathcal{S}$ with number parameters in N an $\mathcal{S}N$-literal. An *$\mathcal{S}N$-type* is a set τ of $\mathcal{S}N$-literals containing $\top$ and satisfying the following three conditions:

- for every $\mathcal{S}N$-literal L, $L \in \tau$ iff $\neg L \notin \tau$;
- for any $m, n \in N$ with $m < n$, $\geqslant n\ R \in \tau$ implies $\geqslant m\ R \in \tau$;
- for any $m, n \in N$ with $m < n$, $\leqslant m\ R \in \tau$ implies $\leqslant n\ R \in \tau$.

For ease of presentation, in what follows, we assume $\mathcal{S}$ and N are fixed, and we treat the conjunction of all its literals, $\bigsqcap_{L\in\tau} L$ as an alternative representation of a type τ.

Each general concept C over $\mathcal{S}$ and N in DL-Lite$^{\mathcal{N}}_{bool}$ can be equivalently represented as a disjunction of $\mathcal{S}N$-types. If we denote the set of all such types for C as $Ts(C)$, then C is equivalent to $\bigsqcup_{\tau\in Ts(C)} \bigsqcap_{L\in\tau} L$. Similarly, $\neg C$ is equivalent to $\bigsqcup_{\tau\in \overline{Ts(C)}} \bigsqcap_{L\in\tau} L$ where $\overline{Ts(C)} = \{\tau \mid \tau \text{ is a } \mathcal{S}N\text{-type, but } \tau \notin Ts(C)\}$.

Given an interpretation $\mathcal{I}$ and an individual $d \in \Delta^{\mathcal{I}}$, the type *realized by* $\mathcal{I}$ on d is defined as the $\mathcal{S}N$-type $\tau_{\mathcal{I}}(d) = \{L \mid L \text{ is a } \mathcal{S}N\text{-literal s.t. } d \in L^{\mathcal{I}}\}$. Define $\Xi_{\mathcal{I}} = \{\tau_{\mathcal{I}}(d) \mid d \in \Delta^{\mathcal{I}}\}$ to be the $\mathcal{S}N$-type set *realized by* $\mathcal{I}$. Given a KB $\mathcal{K}$, define $\Xi_{\mathcal{K}} = \bigcup_{\mathcal{I}\in \mathsf{Mod}(\mathcal{K})} \Xi_{\mathcal{I}}$ to be the $\mathcal{S}N$-type set *realized by* $\mathcal{K}$.

The definition of types are previously introduced in [21] as a model-theoretic characterization for TBox concept/query entailment in DL-Lite$^{\mathcal{N}}_{bool}$. Now we extend it to provide a model-theoretic characterization for ABoxes.

Given a set $\mathcal{O}$ of individual names (objects), we define an *$\mathcal{O}$-graph* over $\mathcal{S}$ and N, denoted $\mathcal{G} = (\mathcal{O}, E, F)$, to be a finite directed graph such that each node $a \in \mathcal{O}$ is labeled by a set $F(a)$ of $\mathcal{S}N$-types and each edge $(a, b) \in E$ is labeled by a set $F(a, b)$ of role names in $\mathcal{S}$.

We are mainly interested in those $\mathcal{O}$-graphs that are determined by interpretations. Given an interpretation $\mathcal{I}$, the $\mathcal{O}$-graph *realized by* $\mathcal{I}$ is defined as follows:

- for each $a \in \mathcal{O}$, $F(a)$ is a singleton of the type realized by $\mathcal{I}$ on a, *i.e.*, $F(a) = \{\tau_{\mathcal{I}}(a^{\mathcal{I}})\}$;

- for each pair $a, b \in \mathcal{O}$, $F(a, b)$ is the set of roles relating a and b in $\mathcal{I}$, *i.e.*, $(a, b) \in E$ iff there exists a role name $P \in \mathcal{S}$ with $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$, and $F(a, b) = \{P \in \mathcal{S} \mid (a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}\}$.

An $\mathcal{O}$-graph realized by some interpretation can be viewed as a 'Herbrand interpretation' for ABoxes over $\mathcal{O}$.

Alternatively, we will also omit E and use the pair $(\mathcal{O}, F)$ to represent an $\mathcal{O}$-graph $\mathcal{G} = (\mathcal{O}, E, F)$ by defining $F(a, b) = \emptyset$ for every $(a, b) \notin E$.

Given a KB $\mathcal{K}$, the $\mathcal{O}$-graph *realized by* $\mathcal{K}$, denoted $\mathcal{G}_{\mathcal{K}} = (\mathcal{O}, F_{\mathcal{K}})$, is the graph obtained by combining $\mathcal{O}$-graphs $\mathcal{G}_{\mathcal{I}} = (\mathcal{O}, F_{\mathcal{I}})$ realized by the models $\mathcal{I}$ of $\mathcal{K}$:

- for each $a \in \mathcal{O}$, $F_{\mathcal{K}}(a) = \bigcup_{\mathcal{I} \in \mathsf{Mod}(\mathcal{K})} F_{\mathcal{I}}(a)$;
- for each pair $a, b \in \mathcal{O}$, $F_{\mathcal{K}}(a, b) = \bigcap_{\mathcal{I} \in \mathsf{Mod}(\mathcal{K})} F_{\mathcal{I}}(a, b)$.

For two $\mathcal{O}$-graphs $\mathcal{G}_1 = (\mathcal{O}, F_1)$ and $\mathcal{G}_2 = (\mathcal{O}, F_2)$, we call $\mathcal{G}_2$ a *sub-graph* of $\mathcal{G}_1$ if

- for each $a \in \mathcal{O}$, $F_1(a) \subseteq F_2(a)$;
- for each pair $a, b \in \mathcal{O}$, $F_2(a, b) \subseteq F_1(a, b)$.

Denote $\mathcal{G}_1 = \mathcal{G}_2$ if $\mathcal{G}_1$ and $\mathcal{G}_2$ are sub-graphs of each other.

Recall that, for a KB $\mathcal{K}$, $\mathsf{Ind}(\mathcal{K})$ denotes the set of all individual names in $\mathcal{K}$ and $\mathsf{Num}(\mathcal{K})$ the set of all numerical parameters in $\mathcal{K}$ together with 1.

The following lemma shows that the entailment of inclusions and assertions in DL-Lite$^{\mathcal{N}}_{bool}$ can be charaterized by, respectively, the type set and $\mathcal{O}$-graph realized by the KB.

Lemma 5.2 *Let $\mathcal{K}_1, \mathcal{K}_2$ be two KBs, $\mathcal{S}$ be a signature, N be a set of natural numbers including* 1 *and $\mathcal{O}$ be a set of individual names. Then,*

(1) $\Xi_{\mathcal{K}_1} \subseteq \Xi_{\mathcal{K}_2}$ iff for any $\mathcal{L}$-inclusion α over $\mathcal{S}$ and N, $\mathcal{K}_2 \models \alpha$ implies $\mathcal{K}_1 \models \alpha$;
(2) $\mathcal{G}_{\mathcal{K}_2}$ is a sub-graph of $\mathcal{G}_{\mathcal{K}_1}$ iff for any $\mathcal{L}$-assertion β over $\mathcal{S}$, N and $\mathcal{O}$, $\mathcal{K}_2 \models \beta$ implies $\mathcal{K}_1 \models \beta$.

We note that Theorem 11 in [21] is a special case of the above item (1) when $\mathcal{K}$ is a TBox.

Proof Since the type set $\Xi_{\mathcal{K}}$ is independent of the ABox of $\mathcal{K}$, the item (1) immediately follows from Theorem 11 in [21]. So we only need to show the item (2). Let $\mathcal{G}_{\mathcal{K}_1} = (\mathcal{O}, F_1)$ and $\mathcal{G}_{\mathcal{K}_2} = (\mathcal{O}, F_2)$.

The "if" direction: On the contrary, suppose $\mathcal{G}_{\mathcal{K}_2}$ is not a sub-graph of $\mathcal{G}_{\mathcal{K}_1}$. There are two possible cases:

- There exists some $a \in \mathcal{O}$ and $\mathcal{S}N$-type τ such that $\tau \in F_1(a)$ but $\tau \notin F_2(a)$. In this case, we have for each model $\mathcal{I}_2$ of $\mathcal{K}_2$, $a^{\mathcal{I}_2} \notin (\bigsqcap_{L \in \tau} L)^{\mathcal{I}_2}$. That is, $\mathcal{K}_2 \models (\neg \bigsqcap_{L \in \tau} L)(a)$, where $(\neg \bigsqcap_{L \in \tau} L)(a)$ is an $\mathcal{L}$-assertion over $\mathcal{O}$, $\mathcal{S}$ and N. However, $\mathcal{K}_1 \not\models (\neg \bigsqcap_{L \in \tau} L)(a)$.

- There exists a pair $a, b \in \mathcal{O}$ and $P \in \mathcal{S}$ such that $P \in F_2(a, b)$ but $P \notin F_1(a, b)$. In this case, every model of $\mathcal{K}_2$ satisfies $P(a, b)$, that is, $\mathcal{K}_2 \models P(a, b)$. $P(a, b)$ is an $\mathcal{L}$-assertion over $\mathcal{O}$ and $\mathcal{S}$ but $\mathcal{K}_1 \not\models P(a, b)$.

In both cases, we have shown that there always exists an $\mathcal{L}$-assertion β over $\mathcal{S}$, N and $\mathcal{O}$ such that $\mathcal{K}_2 \models \beta$ but $\mathcal{K}_1 \not\models \beta$.

The "only if" direction: Suppose that there exists an $\mathcal{L}$-assertion β over $\mathcal{S}$, N and $\mathcal{O}$ such that $\mathcal{K}_2 \models \beta$ but $\mathcal{K}_1 \not\models \beta$.

- If β is of the form $C(a)$, then there exists a model $\mathcal{I}_1$ of $\mathcal{K}_1$ with $a^{\mathcal{I}_1} \in (\neg C)^{\mathcal{I}_1}$. Denote $\tau = \tau_{\mathcal{I}_1}(a^{\mathcal{I}_1})$. We have $\tau \in F_1(a)$ and $\tau \in \overline{Ts(C)}$. For each model $\mathcal{I}_2$ of $\mathcal{K}_2$, $a^{\mathcal{I}_2} \in C^{\mathcal{I}_2}$. It must be the case that $F_2(a) \subseteq Ts(C)$ and thus, $\tau \notin F_2(a)$. That is, $F_1(a) \not\subseteq F_2(a)$.

- If β is of the form $P(a, b)$ or $P^-(b, a)$ with P a role name, then every model of $\mathcal{K}_2$ satisfies $P(a, b)$, which implies $P \in F_2(a, b)$. However, there exists a model $\mathcal{I}_1$ of $\mathcal{K}_1$ with $\mathcal{I}_1 \not\models P(a, b)$. This implies that $P \notin F_1(a, b)$ and thus $F_2(a, b) \not\subseteq F_1(a, b)$.

In either case, $\mathcal{G}_{\mathcal{K}_2}$ is not a sub-graph of $\mathcal{G}_{\mathcal{K}_1}$.

■

We present the model-theoretic characterization for $\mathcal{Q}_{\mathcal{L}}$-forgetting as follows.

Theorem 5.5 *Let $\mathcal{K}$ be a KB and $\mathcal{S}$ a signature. Denote $\Sigma = \mathsf{Sig}(\mathcal{K}) - \mathcal{S}$, $N = \mathsf{Num}(\mathcal{K})$ and $\mathcal{O} = \mathsf{Ind}(\mathcal{K})$. Given a KB $\mathcal{K}'$ with $\mathsf{Sig}(\mathcal{K}') \subseteq \Sigma$ such that $\mathcal{K} \models \mathcal{K}'$, the following two conditions are equivalent:*

(1) $\mathcal{K}'$ is a result of $\mathcal{Q}_{\mathcal{L}}$-forgetting about $\mathcal{S}$ in $\mathcal{K}$;
(2) $\mathcal{K}$ and $\mathcal{K}$ realize the same ΣN-type set and $\mathcal{O}$-graph.

Proof (1) $\Rightarrow$ (2): From the definition of $\mathcal{Q}_{\mathcal{L}}$-forgetting, $\mathcal{K} \models q$ iff $\mathcal{K}' \models q$ for any query $q \in \mathcal{Q}_{\mathcal{L}}$ over Σ, N and $\mathcal{O}$. Since q can be an arbitrary $\mathcal{L}$-inclusion or $\mathcal{L}$assertion, from Lemma 5.2, we have $\Xi_{\mathcal{K}} = \Xi_{\mathcal{K}'}$ and $\mathcal{G}_{\mathcal{K}} = \mathcal{G}_{\mathcal{K}'}$, *i.e.*, (2) holds.

(2) $\Rightarrow$ (1): By Lemma 5.2, we have $\mathcal{K} \models q$ implies $\mathcal{K}' \models q$ for any query $q \in \mathcal{Q}_{\mathcal{L}}$ over Σ, N and $\mathcal{O}$.

When extending Σ to any larger signature Σ' s.t. $\Sigma' \cap \mathcal{S} = \emptyset$, and $N, \mathcal{O}$ to, respectively $N', \mathcal{O}'$, the corresponding $\Sigma' N'$-type set and $\mathcal{O}'$-graph realized by $\mathcal{K}$ is just the trivial extension of the original one. From $\mathcal{K} \models \mathcal{K}'$, the $\Sigma' N'$-type set and $\mathcal{O}'$-graph realized by $\mathcal{K}'$ must also be trivially extended. Thus we still have $\Xi_{\mathcal{K}} = \Xi_{\mathcal{K}'}$ and $\mathcal{G}_{\mathcal{K}} = \mathcal{G}_{\mathcal{K}'}$. As a result, the above conclusion can be extended to a larger query set (over Σ', N' and $\mathcal{O}'$). That is, for any query $q \in \mathcal{Q}_{\mathcal{L}}$ such that $\mathsf{Sig}(q) \cap \mathcal{S} = \emptyset$, $\mathcal{K} \models q$ implies $\mathcal{K}' \models q$.

■

To provide model-theoretic characterization for $\mathcal{Q}^u_{\mathcal{L}}$-forgetting, we need to introduce a notion of *multiple model*, which is originally introduced in [21]. The idea of introducing multiple model is to include copies of a given model in one model so that two models whose domains have different cardinalities can be compared while the semantics is not changed.

Denote $\mathcal{I}^n$ the interpretation obtained from $\mathcal{I}$ and a number $n \geq 1$ in the following way:

- $\Delta^{\mathcal{I}^n} = \{d^{(i)} \mid d \in \Delta^{\mathcal{I}},\ 1 \leq i \leq n\}$;
- for each individual name a, $a^{\mathcal{I}^n} = (a^{\mathcal{I}})^{(1)}$;
- for each concept name A, $A^{\mathcal{I}^n} = \{d^{(i)} \mid d \in A^{\mathcal{I}},\ 1 \leq i \leq n\}$.
- for each role name P, $P^{\mathcal{I}^n} = \{(d^{(i)}, e^{(i)}) \mid (d, e) \in P^{\mathcal{I}},\ 1 \leq i \leq n\}$.

Note that the second item above requires that under $\mathcal{I}^n$, every individual name is assigned to an element of the first copy.

Then we have the following observations:
(1) for any inclusion, assertion or grounded PEQ β, $\mathcal{I} \models \beta$ iff $\mathcal{I}^n \models \beta$;
(2) $\tau_{\mathcal{I}^n}(d^{(i)}) = \tau_{\mathcal{I}}(d)$ for all $1 \leq i \leq n$;
(3) $\Xi_{\mathcal{I}} = \Xi_{\mathcal{I}^n}$ and $\mathcal{G}_{\mathcal{I}} = \mathcal{G}_{\mathcal{I}^n}$.

When n is infinitely large, we will denote $\mathcal{I}^n$ as $\mathcal{I}^\infty$.

The following lemma shows that if two KBs have models realizing the same type sets and $\mathcal{O}$-graphs, then some correspondence can be found between their models. Specificly, a weak model equivalence relation (over their multiple models) holds for corresponding models.

Lemma 5.3 *Let $\mathcal{K}_1, \mathcal{K}_2$ be two KBs and $\mathcal{S}$ a signature. Let $\Sigma = \mathsf{Sig}(\mathcal{K}_1) - \mathcal{S}$, $N = \mathsf{Num}(\mathcal{K}_1)$, $\mathcal{O} = \mathsf{Ind}(\mathcal{K}_1)$ and $\mathsf{Sig}(\mathcal{K}_2) \subseteq \Sigma$.*

Given a model $\mathcal{I}_2$ of $\mathcal{K}_2$, suppose that there is a model $\mathcal{I}_1$ of $\mathcal{K}_1$ such that $\mathcal{I}_1$ realizes the same ΣN-type set and $\mathcal{O}$-graph as $\mathcal{I}_2$. Then there exists a model $\mathcal{I}$ of $\mathcal{K}_1$ such that $\mathcal{I} \sim_{\mathcal{S}} \mathcal{I}_2^\infty$.

Proof Since $\mathcal{I}_1$ and $\mathcal{I}_2$ realize the same type set, $\mathcal{I}_1^\infty$ and $\mathcal{I}_2^\infty$ also realize the same type set, denoted as Ξ. Given each type in Ξ is realized by $\mathcal{I}_1^\infty$ and $\mathcal{I}_2^\infty$, respectively, with infinitely many individuals, there is a bijection $f : \Delta^{\mathcal{I}_1^\infty} \to \Delta^{\mathcal{I}_2^\infty}$, such that for all $d \in \Delta^{\mathcal{I}_1^\infty}$, $\mathcal{I}_1^\infty$ realize the same type on d with $\mathcal{I}_2^\infty$ on $f(d)$, *i.e.*, $\tau_{\mathcal{I}_1^\infty}(d) = \tau_{\mathcal{I}_2^\infty}(f(d))$.

Also, since $\mathcal{I}_1$ and $\mathcal{I}_2$ realize the same $\mathcal{O}$-graph, $\mathcal{I}_1$ and $\mathcal{I}_2$ realize the same type on each individual $a \in \mathcal{O}$, *i.e.*, $\tau_{\mathcal{I}_1}(a^{\mathcal{I}_1}) = \tau_{\mathcal{I}_2}(a^{\mathcal{I}_2})$. We have $\mathcal{I}_1^\infty$ and $\mathcal{I}_2^\infty$ realize the same type on each a. Thus f is still well-defined if we require, additionally, $f(a^{\mathcal{I}_1^\infty}) = a^{\mathcal{I}_2^\infty}$ for each $a \in \mathcal{O}$.

We will construct $\mathcal{I}$ from $\mathcal{I}_1^\infty$ and $\mathcal{I}_2^\infty$ in the following way:

- take $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}_2^\infty}$;
- for each $a \in \mathcal{O}$, $a^{\mathcal{I}} = a^{\mathcal{I}_2^\infty}$;
- for each concept name A, if $A \in \mathcal{S}$, $A^{\mathcal{I}} = \{f(d) \mid d \in A^{\mathcal{I}_1^\infty}\}$, otherwise, $A^{\mathcal{I}} = A^{\mathcal{I}_2^\infty}$;
- for each role name P, if $P \in \mathcal{S}$, $P^{\mathcal{I}} = \{(f(d), f(e)) \mid (d, e) \in P^{\mathcal{I}_1^\infty}\}$, otherwise, $P^{\mathcal{I}} = P^{\mathcal{I}_2^\infty}$.

By the construction of $\mathcal{I}$ we can see that $\mathcal{I} \sim_{\mathcal{S}} \mathcal{I}_2^\infty$. We only need to show that $\mathcal{I} \models \mathcal{K}_1$.

For each concept name A and each individual $d \in \Delta^{\mathcal{I}_1^\infty}$, suppose $A \in \mathcal{S}$, according to the definition of $\mathcal{I}$, we have $d \in A^{\mathcal{I}_1^\infty}$ iff $f(d) \in A^{\mathcal{I}}$. Otherwise if $A \notin \mathcal{S}$, since $\tau_{\mathcal{I}_1^\infty}(d) = \tau_{\mathcal{I}_2^\infty}(f(d))$, we have $d \in A^{\mathcal{I}_1^\infty}$ iff $f(d) \in A^{\mathcal{I}_2^\infty}$. Since $A^{\mathcal{I}} = A^{\mathcal{I}_2^\infty}$ in this case, we still have $d \in A^{\mathcal{I}_1^\infty}$ iff $f(d) \in A^{\mathcal{I}}$. Similarly, we can show that for any concept $\geqslant n\ R$ with $n \in N$, $d \in (\geqslant n\ R)^{\mathcal{I}_1^\infty}$ iff $f(d) \in (\geqslant n\ R)^{\mathcal{I}}$. Thus for any general concept C, $d \in C^{\mathcal{I}_1^\infty}$ iff $f(d) \in C^{\mathcal{I}}$.

Now we have shown that for any inclusion $C_1 \sqsubseteq C_2$ in $\mathcal{K}_1$, $C_1^{\mathcal{I}_1^\infty} \subseteq C_2^{\mathcal{I}_1^\infty}$ iff $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$. Since $\mathcal{I}_1^\infty \models (C_1 \sqsubseteq C_2)$ iff $\mathcal{I}_1 \models (C_1 \sqsubseteq C_2)$, we have $\mathcal{I}_1 \models (C_1 \sqsubseteq C_2)$ iff $\mathcal{I} \models (C_1 \sqsubseteq C_2)$.

For each assertion $C(a)$ in $\mathcal{K}_1$, as shown above, $a^{\mathcal{I}_1^\infty} \in C^{\mathcal{I}_1^\infty}$ iff $f(a^{\mathcal{I}_1^\infty}) \in C^{\mathcal{I}}$. Since for each $a \in \mathcal{O}$, $f(a^{\mathcal{I}_1^\infty}) = a^{\mathcal{I}_2^\infty} = a^{\mathcal{I}}$, and $\mathcal{I}_1^\infty \models C(a)$ iff $\mathcal{I}_1 \models C(a)$, we have $\mathcal{I}_1 \models C(a)$ iff $\mathcal{I} \models C(a)$.

For each assertion $P(a, b)$ or $P^-(b, a)$ in $\mathcal{K}_1$, if $P \in \mathcal{S}$, according to the definition of $\mathcal{I}$, $(a^{\mathcal{I}_1^\infty}, b^{\mathcal{I}_1^\infty}) \in P^{\mathcal{I}_1^\infty}$ iff $(f(a^{\mathcal{I}_1^\infty}), f(b^{\mathcal{I}_1^\infty})) \in P^{\mathcal{I}}$. Since $f(a^{\mathcal{I}_1^\infty}) = a^{\mathcal{I}_2^\infty} = a^{\mathcal{I}}$ and $f(b^{\mathcal{I}_1^\infty}) = b^{\mathcal{I}_2^\infty} = b^{\mathcal{I}}$, we have $\mathcal{I}_1^\infty \models P(a, b)$ iff $\mathcal{I} \models P(a, b)$. That is, $\mathcal{I}_1 \models P(a, b)$ iff $\mathcal{I} \models P(a, b)$. Otherwise, $P \notin \mathcal{S}$, we have $P^{\mathcal{I}} = P^{\mathcal{I}_2^\infty}$. Since $\mathcal{I}_1$ and $\mathcal{I}_2$ realize the same $\mathcal{O}$-graph, $\mathcal{I}_1^\infty \models P(a, b)$ iff $\mathcal{I}_2^\infty \models P(a, b)$. As $a^{\mathcal{I}_2^\infty} = a^{\mathcal{I}}$ and $b^{\mathcal{I}_2^\infty} = b^{\mathcal{I}}$, we have $\mathcal{I}_2^\infty \models P(a, b)$ iff $\mathcal{I} \models P(a, b)$. Thus, $\mathcal{I}_1^\infty \models P(a, b)$ iff $\mathcal{I} \models P(a, b)$, and again, $\mathcal{I}_1 \models P(a, b)$ iff $\mathcal{I} \models P(a, b)$.

Since $\mathcal{I} \models \alpha$ for each inclusion or assertion $\alpha \in \mathcal{K}_1$, we have shown that $\mathcal{I} \models \mathcal{K}_1$.

■

The following result shows that $\mathcal{Q}^u_{\mathcal{L}}$-forgetting can be charaterized by the type sets and $\mathcal{O}$-graphs realised by the models of the KB.

Theorem 5.6 *Let $\mathcal{K}$ be a KB and $\mathcal{S}$ a signature. Denote $\Sigma = \mathsf{Sig}(\mathcal{K}) - \mathcal{S}$, $N = \mathsf{Num}(\mathcal{K})$ and $\mathcal{O} = \mathsf{Ind}(\mathcal{K})$. Given a KB $\mathcal{K}'$ with $\mathsf{Sig}(\mathcal{K}') \subseteq \Sigma$ such that $\mathcal{K} \models \mathcal{K}'$, the following three conditions are equivalent:*

(1) $\mathcal{K}'$ is a result of $\mathcal{Q}^u_{\mathcal{L}}$-forgetting about $\mathcal{S}$ in $\mathcal{K}$;

(2) for each model $\mathcal{I}'$ of $\mathcal{K}'$, there always exists a model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I}$ and $\mathcal{I}'$ realize the same ΣN-type set and $\mathcal{O}$-graph;

(3) for each model $\mathcal{I}'$ of $\mathcal{K}'$, there always exists a model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \sim_{\mathcal{S}} (\mathcal{I}')^{\infty}$.

Proof (1) $\Rightarrow$ (2): Conversely, suppose there is a model $\mathcal{I}'$ of $\mathcal{K}'$ realizing ΣN-type set Ξ and $\mathcal{O}$-graph $\mathcal{G} = (\mathcal{O}, F)$, such that there exists no model of $\mathcal{K}$ realizing the same type set and $\mathcal{O}$-graph. We want to construct a query $q \in \mathcal{Q}^u_{\mathcal{L}}$ over Σ, N and $\mathcal{O}$, such that $\mathcal{K} \models q$ but $\mathcal{K}' \not\models q$.

For each model $\mathcal{I}$ of $\mathcal{K}$, denote the ΣN-type set realized by $\mathcal{I}$ as $\Xi_{\mathcal{I}}$, and the $\mathcal{O}$-graph realized by $\mathcal{I}$ as $\mathcal{G}_{\mathcal{I}} = (\mathcal{O}, F_{\mathcal{I}})$. Let $\mathcal{P} = \{ \ P(a,b) \mid a,b \in \mathcal{O}, P \notin F(a,b)$ but $P \in F_{\mathcal{I}}(a,b)$ for some $\mathcal{I} \in \mathsf{Mod}(\mathcal{K}) \ \}$.

We can construct a query q, which is the union of all (possibly negated role) assertions of the following forms in DL-Lite$^u_{bool}$:

- $\neg(\exists\, u. \bigsqcap_{L \in \tau} L)(a_\tau)$ with $\tau \in \Xi$, where a_τ is a new individual name for τ;
- $(\exists\, u. \bigsqcap_{L \in \tau} L)(a_\tau)$ with τ a ΣN-type but $\tau \notin \Xi$, and a_τ a new individual name for τ;
- $\neg(\bigsqcap_{L \in \tau_a} L)(a)$ with $a \in \mathcal{O}$ and $F(a) = \{\tau_a\}$;
- $\neg P(a,b)$ with $a,b \in \mathcal{O}$ and $P \in F(a,b)$;
- $P(a,b)$ with $P(a,b) \in \mathcal{P}$.

Informally, the first two types of assertions capture the difference between $\Xi_{\mathcal{I}}$ and Ξ, for some model $\mathcal{I}$ of $\mathcal{K}$, and the other three capture the difference between $\mathcal{G}_{\mathcal{I}}$ and $\mathcal{G}$. Since there exists no model of $\mathcal{K}$ realizing exactly the type set Ξ and $\mathcal{O}$-graph $\mathcal{G}$, we have $\mathcal{I} \models q$ for each model $\mathcal{I}$ of $\mathcal{K}$. That is, $\mathcal{K} \models q$. However, as model $\mathcal{I}'$ does not satisfy q, $\mathcal{K}' \not\models q$.

(2) $\Rightarrow$ (3) follows directly from Lemma 5.3.

(3) $\Rightarrow$ (1): For any query $q \in \mathcal{Q}^u_{\mathcal{L}}$ with $\mathsf{Sig}(q) \cap \mathcal{S} = \emptyset$, suppose $\mathcal{K}' \not\models q$, then there is a model $\mathcal{I}'$ of $\mathcal{K}'$ such that $\mathcal{I}' \not\models q$. There exists a model $\mathcal{I}$ of $\mathcal{K}$ with $\mathcal{I} \sim_{\mathcal{S}} (\mathcal{I}')^{\infty}$. Since $(\mathcal{I}')^{\infty} \not\models q$, we have $\mathcal{I} \not\models q$, and thus $\mathcal{K} \not\models q$.

■

In $\mathcal{Q}^c_{\mathcal{L}}$, one is allowed to inquire cardinality of concepts. In order to generalize the above model-theoretic characterization of $\mathcal{Q}^u_{\mathcal{L}}$-forgetting to apply to $\mathcal{Q}^c_{\mathcal{L}}$-forgetting, a notion of *type-cardinality* is needed.

Given an interpretation $\mathcal{I}$ and a type τ, we define $\mathsf{Card}_{\mathcal{I}}(\tau) = \sharp(\{d \in \Delta^{\mathcal{I}} \mid \tau_{\mathcal{I}}(d) = \tau\})$. That is, $\mathsf{Card}_{\mathcal{I}}(\tau)$ is the number of individuals on which τ is realized in $\mathcal{I}$. Note that $\mathsf{Card}_{\mathcal{I}}(\tau)$ can be ∞.

We have the following lemma, which is similar to Lemma 5.3 but with additional restriction on the cardinality of types and guaranteeing a stronger model equivalence relation.

Lemma 5.4 *Let $\mathcal{K}_1, \mathcal{K}_2$ be two KBs, $\mathcal{S}$ be a signature, and $\mathsf{Sig}(\mathcal{K}_2) \subseteq \mathsf{Sig}(\mathcal{K}_1) - \mathcal{S}$. Denote $\Sigma = \mathsf{Sig}(\mathcal{K}_1) - \mathcal{S}$, $N = \mathsf{Num}(\mathcal{K}_1)$ and $\mathcal{O} = \mathsf{Ind}(\mathcal{K}_1)$. Given a model $\mathcal{I}_2$ of $\mathcal{K}_2$, suppose there is a model $\mathcal{I}_1$ of $\mathcal{K}_1$ such that*

- *$\mathcal{I}_1$ and $\mathcal{I}_2$ realize the same ΣN-type set Ξ;*
- *for each type $\tau \in \Xi$, $\mathcal{I}_1$ and $\mathcal{I}_2$ realize τ with the same number of individuals;*
- *$\mathcal{I}_1$ and $\mathcal{I}_2$ realize the same $\mathcal{O}$-graph.*

Then there exists a model $\mathcal{I}$ of $\mathcal{K}_1$ such that $\mathcal{I} \sim_{\mathcal{S}} \mathcal{I}_2$.

Proof Since $\mathsf{Card}_{\mathcal{I}_1}(\tau) = \mathsf{Card}_{\mathcal{I}_2}(\tau)$ for each $\tau \in \Xi$, we have $\sharp(\Delta^{\mathcal{I}_1}) = \sharp(\Delta^{\mathcal{I}_2})$, and there is a bijection $f : \Delta^{\mathcal{I}_1} \to \Delta^{\mathcal{I}_2}$, such that $\tau_{\mathcal{I}_1}(d) = \tau_{\mathcal{I}_2}(f(d))$ for all $d \in \Delta^{\mathcal{I}_1}$. Also, since $\mathcal{I}_1$ and $\mathcal{I}_2$ realize the same $\mathcal{O}$-graph, f is still well-defined if we require, additionally, $f(a^{\mathcal{I}_1}) = a^{\mathcal{I}_2}$ for each $a \in \mathcal{O}$.

We will construct $\mathcal{I}$ from $\mathcal{I}_1$ and $\mathcal{I}_2$ in the following way:

- we take $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}_2}$;
- for each individual name $a \in \mathcal{O}$, $a^{\mathcal{I}} = a^{\mathcal{I}_2}$;
- for each concept name A, if $A \in \mathcal{S}$, $A^{\mathcal{I}} = \{f(d) \mid d \in A^{\mathcal{I}_1}\}$, otherwise, $A^{\mathcal{I}} = A^{\mathcal{I}_2}$;
- for each role name P, if $P \in \mathcal{S}$, $P^{\mathcal{I}} = \{(f(d), f(e)) \mid (d,e) \in P^{\mathcal{I}_1}\}$, otherwise, $P^{\mathcal{I}} = P^{\mathcal{I}_2}$.

By the construction of $\mathcal{I}$ we have $\mathcal{I} \sim_{\mathcal{S}} \mathcal{I}_2$. Similar to the proof of Lemma 5.3, we can also show that $\mathcal{I} \models \mathcal{K}_1$.

■

Also, we have the following similar result as Theorem 5.6.

Theorem 5.7 *Let $\mathcal{K}$ be a KB and $\mathcal{S}$ a signature. Denote $\Sigma = \mathsf{Sig}(\mathcal{K}) - \mathcal{S}$, $N = \mathsf{Num}(\mathcal{K})$ and $\mathcal{O} = \mathsf{Ind}(\mathcal{K})$. Given a KB $\mathcal{K}'$ with $\mathsf{Sig}(\mathcal{K}') \subseteq \Sigma$ such that $\mathcal{K} \models \mathcal{K}'$, the following three conditions are equivalent:*

(1) $\mathcal{K}'$ is a result of $\mathcal{Q}^c_{\mathcal{L}}$-forgetting about $\mathcal{S}$ in $\mathcal{K}$;
(2) for each model $\mathcal{I}'$ of $\mathcal{K}'$, there always exists a model $\mathcal{I}$ of $\mathcal{K}$ such that (a) $\mathcal{I}$ and $\mathcal{I}'$ realize the same ΣN-type set Ξ, (b) for each type $\tau \in \Xi$, $\mathsf{Card}_{\mathcal{I}}(\tau) = \mathsf{Card}_{\mathcal{I}'}(\tau)$, and (c) $\mathcal{I}$ and $\mathcal{I}'$ realize the same $\mathcal{O}$-graph;
(3) for each model $\mathcal{I}'$ of $\mathcal{K}'$, there always exists a model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \sim_{\mathcal{S}} \mathcal{I}'$.

Proof (1) $\Rightarrow$ (2): Conversely, suppose there exists a model $\mathcal{I}'$ of $\mathcal{K}'$ realizing ΣN-type set Ξ and $\mathcal{O}$-graph $\mathcal{G} = (\mathcal{O}, F)$, such that for each model $\mathcal{I}$ of $\mathcal{K}$, realizing type set $\Xi_{\mathcal{I}}$ and $\mathcal{O}$-graph $\mathcal{G}_{\mathcal{I}} = (\mathcal{O}, F_{\mathcal{I}})$, at least one of the following three conditions holds: (A) $\Xi_{\mathcal{I}} \neq \Xi$, (B) $\mathsf{Card}_{\mathcal{I}}(\tau) \neq \mathsf{Card}_{\mathcal{I}'}(\tau)$ for some type $\tau \in \Xi$, and (C) $\mathcal{G}_{\mathcal{I}} \neq \mathcal{G}$.

We want to construct a query $q \in \mathcal{Q}^c_{\mathcal{L}}$ over Σ, N and $\mathcal{O}$, such that $\mathcal{K} \models q$ but $\mathcal{K}' \not\models q$.

Let $\mathcal{P} = \{P(a,b) \mid a,b \in \mathcal{O}, P \notin F(a,b)$ but $P \in F_{\mathcal{I}}(a,b)$ for some $\mathcal{I} \in \mathsf{Mod}(\mathcal{K})\}$.

We can construct a query q, which is the union of all (possibly negated role) assertions of the following forms in DL-Lite$^c_{bool}$:

-
$$\neg\,(\geqslant \mathsf{Card}_{\mathcal{I}'}(\tau)\; u.\prod_{L\in\tau} L \sqcap \leqslant \mathsf{Card}_{\mathcal{I}'}(\tau)\; u.\prod_{L\in\tau} L)(a_\tau)$$
with $\tau \in \Xi$, where a_τ is a new individual name for τ;
- $(\exists\, u.\prod_{L\in\tau} L)(a_\tau)$ with τ a ΣN-type but $\tau \notin \Xi$, and a_τ a new individual name for τ;
- $\neg(\prod_{L\in\tau_a} L)(a)$ with $a \in \mathcal{O}$ and $F(a) = \{\tau_a\}$;
- $\neg P(a,b)$ with $a,b \in \mathcal{O}$ and $P \in F(a,b)$;
- $P(a,b)$ with $P(a,b) \in \mathcal{P}$.

Similar to the proof of Theorem 5.6, the first two types of assertions correspond to items (A) and (B), and the other three correspond to (C). Since for each model $\mathcal{I}$ of $\mathcal{K}$, either (A), or (B), or (C) holds, we have $\mathcal{I} \models q$ for each model $\mathcal{I}$ of $\mathcal{K}$. That is, $\mathcal{K} \models q$. However, as model $\mathcal{I}'$ does not satisfy q, $\mathcal{K}' \not\models q$.

(2) $\Rightarrow$ (3) follows directly from Lemma 5.4.

(3) $\Rightarrow$ (1): For any query $q \in \mathcal{Q}^c_{\mathcal{L}}$ with $\mathsf{Sig}(q) \cap \mathcal{S} = \emptyset$ such that $\mathcal{K}' \not\models q$, there is a model $\mathcal{I}'$ of $\mathcal{K}'$ such that $\mathcal{I}' \not\models q$. There exists a model $\mathcal{I}$ of $\mathcal{K}$ with $\mathcal{I} \sim_{\mathcal{S}} \mathcal{I}'$. We have $\mathcal{I} \not\models q$, and thus $\mathcal{K} \not\models q$.

■

5.4 Proofs for Section 5.2

In what follows, we use the model-theoretic characterizations introduced in previous section to provide proofs for results in Section 5.2.

We show the existence of $\mathcal{Q}_{\mathcal{L}}$-forgetting by constructing a result of forgetting from the type set and $\mathcal{O}$-graph realized by the original KB.

Proof of Theorem 5.2 Let $\Sigma = \mathsf{Sig}(\mathcal{K}) - \mathcal{S}$, $N = \mathsf{Num}(\mathcal{K})$ and $\mathcal{O} = \mathsf{Ind}(\mathcal{K})$. Let Ξ and $\mathcal{G} = (\mathcal{O}, F)$ be, respectively, the type set and $\mathcal{O}$-graph realized by $\mathcal{K}$ over Σ and N. We can construct $\mathcal{K}' = \langle \mathcal{T}', \mathcal{A}' \rangle$ from Ξ and $\mathcal{G}$ in the following way:

Let $\mathcal{T}' = \{ \bigsqcap_{L \in \tau} L \sqsubseteq \bot \mid \tau$ is a ΣN-type, but $\tau \notin \Xi \}$, and

$\mathcal{A}' = \{ (\bigsqcup_{\tau \in F(a)} \bigsqcap_{L \in \tau} L)(a) \mid a \in \mathcal{O}\} \cup \{P(a,b) \mid a, b \in \mathcal{O}, P \in F(a,b) \}$.

Obviously, $\mathcal{K}'$ is over $\mathsf{Sig}(\mathcal{K}) - \mathcal{S}$. As we can show, $\mathcal{K}'$ is constructed to satisfy the following two conditions:

(1) $\mathcal{K} \models \mathcal{K}'$. That is, for each model $\mathcal{I}$ of $\mathcal{K}$, $\mathcal{I}$ is also a model of $\mathcal{K}'$:

- $(\bigsqcap_{L \in \tau} L)^{\mathcal{I}} = \emptyset$ for each $\tau \notin \Xi$;
- for each $a \in \mathcal{O}$, $a^{\mathcal{I}} \in (\bigsqcap_{L \in \tau} L)^{\mathcal{I}}$ for some $\tau \in F(a)$;
- for each pair $a, b \in \mathcal{O}$, $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$ for each $P \in F(a,b)$.

(2) Denote Ξ' and $\mathcal{G}' = (\mathcal{O}, F')$ to be, respectively, the type set and $\mathcal{O}$-graph realized by $\mathcal{K}'$ over Σ and N. We have $\Xi' = \Xi$ and $\mathcal{G}' = \mathcal{G}$. This can be seen from that:

- clearly, $\Xi' \subseteq \Xi$, and for each $\tau \in \Xi$, we can always construct a model $\mathcal{I}$ of $\mathcal{K}'$ (satisfying $\mathcal{T}'$) with some $d \in (\bigsqcap_{L \in \tau} L)^{\mathcal{I}}$, that is, τ is also in Ξ';
- $F(a) \subseteq F'(a)$ for each $a \in \mathcal{O}$, and given a model $\mathcal{I}$ of $\mathcal{K}'$, $\mathcal{A}'$ states $a^{\mathcal{I}} \in (\bigsqcap_{L \in \tau} L)^{\mathcal{I}}$ for some $\tau \in F(a)$, that is, $F'(a) \subseteq F(a)$;
- $F'(a,b)$ is exactly $F(a,b)$ for each pair $a, b \in \mathcal{O}$.

By Theorem 5.5, $\mathcal{K}'$ is a result of $\mathcal{Q}_{\mathcal{L}}$-forgetting about $\mathcal{S}$ in $\mathcal{K}$.

■

Based on the model-theoretic charaterization of $\mathcal{Q}^u_{\mathcal{L}}$-forgetting, we can show the proofs for Propositions 5.2, 5.3 and Theorem 5.2.

Proof of Proposition 5.2 Suppose $\mathcal{K}' \not\models q$. Then there exists a model $\mathcal{I}'$ of $\mathcal{K}'$ such that $\mathcal{I}' \not\models q$ for each grounded PEQ q with $\mathsf{Sig}(q) \cap \mathcal{S} = \emptyset$. By Theorem 5.6, there exists a model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \sim_{\mathcal{S}} (\mathcal{I}')^{\infty}$. Since $(\mathcal{I}')^{\infty} \not\models q$, we have $\mathcal{I} \not\models q$, and thus $\mathcal{K} \not\models q$.

■

Proof of Proposition 5.3 Obviously, $\mathsf{Sig}(\mathcal{K}'_1 \cup \mathcal{K}'_2) \subseteq \mathsf{Sig}(\mathcal{K}_1 \cup \mathcal{K}_2) - \mathcal{S}$ and $\mathcal{K}_1 \cup \mathcal{K}_2 \models \mathcal{K}'_1 \cup \mathcal{K}'_2$. We only need to show that for any $q \in \mathcal{Q}^u_{\mathcal{L}}$ with $\mathsf{Sig}(q) \cap \mathcal{S} = \emptyset$, $\mathcal{K}_1 \cup \mathcal{K}_2 \models q$ implies $\mathcal{K}'_1 \cup \mathcal{K}'_2 \models q$.

Suppose $\mathcal{K}'_1 \cup \mathcal{K}'_2 \not\models q$, then there exists a model $\mathcal{I}'$ of both $\mathcal{K}'_1$ and $\mathcal{K}'_2$ such that $\mathcal{I}' \not\models q$. By Theorem 5.6, there exists a model $\mathcal{I}_1$ of $\mathcal{K}_1$ with $\mathcal{I}_1 \sim_{\mathcal{S}} (\mathcal{I}')^{\infty}$ and a model $\mathcal{I}_2$ of $\mathcal{K}_2$ with $\mathcal{I}_2 \sim_{\mathcal{S}} (\mathcal{I}')^{\infty}$. Thus $(\mathcal{I}')^{\infty} \sim_{\mathcal{S}} \mathcal{I}_1 \sim_{\mathcal{S}} \mathcal{I}_2$.

Similar to the proof of Proposition 3.4, we can construct an interpretation $\mathcal{I}$ such that (1) $\mathcal{I} \sim_{\mathcal{S}} (\mathcal{I}')^{\infty}$; (2) $\mathcal{I}$ and $\mathcal{I}_i$ coincide on $\mathsf{Sig}(\mathcal{K}_i) \cap \mathcal{S}$ for $i = 1, 2$. Obviously, $\mathcal{I}$ is a model of $\mathcal{K}_1 \cup \mathcal{K}_2$ and $\mathcal{I} \not\models q$. Thus, $\mathcal{K}_1 \cup \mathcal{K}_2 \not\models q$.

■

Now let us prove Theorem 5.3.

Proof of Theorem 5.3 Denote $\Sigma = \mathsf{Sig}(\mathcal{K}) - \mathcal{S}$, $N = \mathsf{Num}(\mathcal{K})$ and $\mathcal{O} = \mathsf{Ind}(\mathcal{K})$. Let Ξ and $\mathcal{G} = (\mathcal{O}, F)$ be, respectively, the type set and $\mathcal{O}$-graph realized by $\mathcal{K}$ over Σ and N. Similar to the proof of Theorem 5.2, we can construct $\mathcal{K}' = \langle \mathcal{T}', \mathcal{A}' \rangle$ from Ξ and $\mathcal{G}$ as follows.

Let

$$\begin{aligned}\mathcal{T}' = &\{ \sqcap_{L\in\tau} L \sqsubseteq \bot \mid \tau \text{ is a } \Sigma N\text{-type, but } \tau \notin \Xi \} \\ &\cup \{ \sqcap_{L\in\tau} L \sqsubseteq \sqcup_{\Xi'\in\Omega_\tau} (\sqcap_{\tau'\in\Xi'} \exists u. \sqcap_{L\in\tau'} L) \mid \tau \in \Xi \}\end{aligned}$$

where for each $\tau \in \Xi$, Ω_τ is the set of all minimal sets Ξ' of ΣN-types such that $\{\tau\} \cup \Xi'$ is realized by some model $\mathcal{I}$ of $\mathcal{K}$.

$\mathcal{A}' = \{(\sqcup_{\tau\in F(a)} \sqcap_{L\in\tau} L)(a) \mid a \in \mathcal{O}\} \cup \{P(a,b) \mid a,b \in \mathcal{O}, P \in F(a,b)\}$.

Obviously, $\mathsf{Sig}(\mathcal{K}') \subseteq \mathsf{Sig}(\mathcal{K}) - \mathcal{S}$, and $\mathcal{K}'$ is constructed to satisfy the following two conditions:

(1) $\mathcal{K} \models \mathcal{K}'$. That is, for each model $\mathcal{I}$ of $\mathcal{K}$, $\mathcal{I}$ is also a model of $\mathcal{K}'$. Compared to the KB constructed in the proof of Theorem 5.2, the only difference here is that $\mathcal{K}'$ contains inclusions of the form $\sqcap_{L\in\tau} L \sqsubseteq \sqcup_{\Xi'\in\Omega_\tau} (\sqcap_{\tau'\in\Xi'} \exists u. \sqcap_{L\in\tau'} L)$. Given $\tau \in \Xi$, there always exists some $\Xi' \in \Omega_\tau$ such that $\{\tau\} \cup \Xi' \subseteq \Xi_\mathcal{I}$. Thus, each $\tau' \in \Xi'$ is realized by $\mathcal{I}$ and $\left(\sqcap_{\tau'\in\Xi'} \exists u. \sqcap_{L\in\tau'} L)\right)^\mathcal{I} = \Delta^\mathcal{I}$. That is, $\mathcal{I}$ satisfies $\sqcap_{L\in\tau} L \sqsubseteq \sqcup_{\Xi'\in\Omega_\tau} (\sqcap_{\tau'\in\Xi'} \exists u. \sqcap_{L\in\tau'} L)$.

(2) For each inclusion or assertion $q \in \mathcal{Q}^u_\mathcal{L}$ with $\mathsf{Sig}(q) \cap \mathcal{S} = \emptyset$, $\mathcal{K} \models q$ implies $\mathcal{K}' \models q$.

When q is a DL-Lite$^u_{bool}$ inclusion, the above relation has been proved in (the proof of Theorem 22 in) [21]. The intuition is that, for each model $\mathcal{I}'$ of $\mathcal{K}'$ and type set Ξ' realized by $\mathcal{I}'$, it is always possible to construct a model $\mathcal{I}$ of $\mathcal{K}$ realizing the same type set Ξ'. Detailed proof for the inclusion case is omitted here. In what follows, we only consider assertions q of the form $C(a)$ or $R(a,b)$ in DL-Lite$^u_{bool}$. Note that negated role assertions $\neg R(a,b)$ cannot follow from any DL-Lite$_{bool}$ KB, and thus are not taken into consideration.

For assertion q of the form $C(a)$, without loss of generality, we can assume C is in its DNF. That is, $C = \sqcup\sqcap E$, where each E is either an $\mathcal{L}$-literal concept L, or a DL-Lite$^u_{bool}$ literal of the form $(\neg)\exists u.D$ (with D a $\mathcal{L}$-concept). We want to show that for each $E(a)$, $\mathcal{K} \models E(a)$ implies $\mathcal{K}' \models E(a)$: If E is a $\mathcal{L}$-literal concept, the above relation is proved in the proof of Theorem 5.2. If E is a DL-Lite$^u_{bool}$ literal, by the definition of $\exists u.D$, assertion $(\exists u.D)(a)$ is equivalent to inclusion $\top \sqsubseteq \exists u.D$, and $\neg(\exists u.D)(a)$ to $\exists u.D \sqsubseteq \bot$. Since the inclusion case is already proved, we have $\mathcal{K} \models E(a)$ implies $\mathcal{K}' \models E(a)$. For assertion q of the form $R(a,b)$, it is shown in the proof of Theorem 5.2 that $\mathcal{K} \models R(a,b)$ implies $\mathcal{K}' \models R(a,b)$.

By the definition of $\mathcal{Q}^u_\mathcal{L}$-forgetting, we have shown that $\mathcal{K}'$ is a result of $\mathcal{Q}^u_\mathcal{L}$-forgetting about $\mathcal{S}$ in $\mathcal{K}$.

■

With Theorem 5.7, Theorem 5.4 is easily shown as follows.

Proof of Theorem 5.4 From Theorem 5.1, $\mathsf{forget}(\mathcal{K},\mathcal{S})$ is a result of $\mathcal{Q}^c_\mathcal{L}$-forgetting about $\mathcal{S}$ in $\mathcal{K}$.

For the other direction, By Theorem 5.7 and Proposition 3.1, it is readily seen that $\mathsf{forget}(\mathcal{K},\mathcal{S}) = \mathcal{K}'$.

■

6 Related Work

The issue of defining suitable operators for ontology reuse, merging and update in DLs has received much interest recently and several approaches have been proposed, including conservative extension [13,26], module extraction [15,20,19], forgetting [34,21,18], update and erasure [14,25], and ontology repair [17,22,27–30].

Technically, forgetting is very close to the concept of uniform interpolant [32]. In some cases they are even identical. [31] investigates the uniform interpolant for concept descriptions in $\mathcal{ALC}$ and [13] briefly discusses a definition of uniform interpolant for TBoxes in $\mathcal{ALC}$. As explained in Section 1, the work in this paper is a significant extension of our conference paper [34], where the model-based forgetting is proposed for DL-Lite TBoxes. Subsequently, [21] then introduce two alternative forgettings for DL-Lite TBoxes (namely, b-forgetting and u-forgetting).

To the best of our knowledge, forgetting in DL-Lite KBs has not been investigated before. Therefore, one major contribution of this paper is the extension of results for TBox forgetting to KB forgetting. Such an extension is non-trivial because of the involvement of ABoxes, which can be seen from the algorithms and proofs presented. The model-based forgetting introduced in this paper also generalizes the forgetting in [34] to the more expressive DL-Lite$_{bool}^{\mathcal{N}}$. This generalization increases the complexity of algorithms, and also affects the expressibility results. Our query-based forgetting generalizes b-forgetting and u-forgetting in two ways. First, query-based forgetting is defined for KBs, although the extension is straightforward. Second, we use query-based forgetting to provide a unifying framework for defining and comparing different definitions of forgetting. In particular, we have shown that three definitions of forgetting can be embedded in our framework. [10] investigated forgetting for OWL/RDF ontologies by translating an ontology into a logic program. However, their approach is applicable to only a small class of OWL/RDF ontologies.

Conservative extension and module extraction have some similarity with forgetting, but they are different in that the first two approaches support only removing inclusions and assertions, but cannot modify them. As a result, if a TBox $\mathcal{K}'$ has a conservative extension $\mathcal{K}$, then $\mathcal{K}'$ is a result of forgetting in $\mathcal{K}$, but a result of forgetting may not be a (conservative) module.

Update and erasure operations in DL-Lite are discussed in [14]. While both erasure and forgetting are concerned with eliminating information from an ontology, they are quite different. When erasing an assertion $A(a)$ from a DL KB $\mathcal{K}$, only the membership relation between individual a and concept A is removed, while concept name A is not necessarily removed from $\mathcal{K}$. However, forgetting about A in $\mathcal{K}$ involves eliminating all logical relations (*e.g.*, subsumption relation, membership relation, etc.) that refer to A in $\mathcal{K}$.

Another stream of research is about ontology repair, where the major issue is to recover the consistency of an inconsistent ontology by removing a smallest subset from the ontology. Obviously, ontology repair has quite different motivation and assumptions from forgetting.

7 Conclusion

Forgetting provides a promising way of extracting, reusing and merging ontologies. However, it is rarely investigated how to adapt forgetting to knowledge bases in Description Logics. To the best of our knowledge, this paper is the first attempt towards investigating forgetting for KBs in DLs. In this paper, we have introduced model-based forgetting for KBs in DL-Lite$_{bool}^{\mathcal{N}}$ and shown that all major properties of forgetting are satisfied by our forgetting. In particular, we have developed a resolution-like algorithm for computing the result of concept forgetting in DL-Lite$_{bool}^{\mathcal{N}}$ KBs and proved that this algorithm is sound and complete. To define and compare various definitions of forgetting, we have established a hierarchy of forgetting by introducing a parameterized query-based forgetting. After showing how b-forgetting and u-forgetting for TBoxes in [21] can be extended to KBs, we have proved that model-based forgetting, b-forgetting and u-forgetting can be characterized by query-based forgetting.

There are still several interesting issues for future research. First, we are currently working on generalizing the results in this paper to expressive DLs, such as $\mathcal{ALC}$ and $\mathcal{SHIQ}$. While it is straightforward to generalize definitions of forgetting, it is less clear whether these notions of forgetting are suitable for expressive DLs and how to compute the result of KB forgetting in these DLs. Second, we have obtained some results on forgetting in other members of DL-Lite (including a more recent member DL-Lite$^{\mathcal{N},\mathcal{R}}_{bool}$). Further investigation is under way. In particular, a systematic comparison of forgetting for various members of DL-Lite will be presented in a separate paper. Third, a systematic study on the complexity of forgetting is needed. In the setting of DL-Lite, the problem of computing forgetting is exponential in general. We plan to investigate the complexity of various reasoning tasks related to forgetting in the DL-Lite family, $\mathcal{EL}$ family and expressive DLs including $\mathcal{ALC}$ and $\mathcal{SHIQ}$. As a result, tractable classes will be identified. Last, one important issue might be the applications of forgetting in extracting, modularizing, reusing and merging ontologies.

References

1. G. Antoniou and F. van Harmelen. *A Semantic Web Primer*. The MIT Press, 2 edition, 2004.
2. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. DL-Lite in the light of first-order logic. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI07)*, pages 361–366, 2007.
3. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence IJCAI-05*, Edinburgh, UK, 2005. Morgan-Kaufmann Publishers.
4. F. Baader, D. Calvanese, D.McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2002.
5. F. Baader, C. Lutz, and B. Suntisrivaraporn. Is tractable reasoning in extensions of the description logic $\mathcal{EL}$ useful in practice? 2007.
6. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05)*, pages 602–607, 2005.
7. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR-06)*, pages 260–270, 2006.
8. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.
9. M. Dean, D. Connolly, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, and L. Stein. OWL web ontology language reference. http://www.w3.org/tr/2004/rec-owl-ref-20040210/, W3C Recommendation, 10 February 2004.
10. T. Eiter, G. Ianni, R. Schindlauer, H. Tompits, and K. Wang. Forgetting in managing rules and ontologies. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI-06)*, pages 411–419, 2006.
11. T. Eiter and K. Wang. Forgetting and conflict resolving in disjunctive logic programming. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, pages 238–243, 2006.
12. T. Eiter and K. Wang. Semantic forgetting in answer set programming. *Artificial Intelligence*, 14:1644–1672, 2008.
13. S. Ghilardi, C. Lutz, and F. Wolter. Did i damage my ontology? a case for conservative extensions in description logics. In *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 187–197, 2006.
14. G. De Giacomo, M. Lenzerini, A. Poggi, and R. Rosati. On the approximation of instance level update and erasure in description logics. In *Proceedings of the 22th National Conference on Artificial Intelligence (AAAI-07)*, pages 403–408, 2007.
15. B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Just the right amount: extracting modules from ontologies. In *Proceedings of the 16th International Conference on World Wide Web (WWW 2007)*, pages 717–726, 2007.

16. T.R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.
17. A. Kalyanpur, B. Parsia, E. Sirin, and B. C. Grau. Repairing unsatisfiable concepts in OWL ontologies. In *Proceedings of the 3rd European Semantic Web Conference (ESWC06)*, pages 170–184, 2006.
18. B. Konev, F. Wolter, and M. Zakharyaschev. Forgetting and uniform interpolation in large-scale description logic terminologies. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 318–323, 2009.
19. R. Kontchakov, L. Pulina, U. Sattler, T. Schneider, P. Selmer, F. Wolter, and M. Zakharyaschev. Minimal module extraction from DL-Lite ontologies using QBF solvers. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 836–841, 2009.
20. R. Kontchakov, F. Wolter, and M. Zakharyaschev. Modularity in DL-Lite. In *Proceedings of the 2007 International Workshop on Description Logics (DL2007)*, 2007.
21. R. Kontchakov, F. Wolter, and M. Zakharyaschev. Can you tell the difference between DL-Lite ontologies? In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR08)*, pages 285–295. AAAI Press, 2008.
22. S. C. Lam, J. Z. Pan, D. H. Sleeman, and W. W. Vasconcelos. A fine-grained approach to resolving unsatisfiable ontologies. In *Proceedings of Web Intelligence*, pages 428–434, 2006.
23. J. Lang, P. Liberatore, and P. Marquis. Propositional independence: Formula-variable independence and forgetting. *J. Artif. Intell. Res. (JAIR)*, 18:391–443, 2003.
24. F. Lin and R. Reiter. Forget it. In *Proceedings of the AAAI Fall Symposium on Relevance*, pages 154–159. New Orleans (LA), 1994.
25. H. Liu, C. Lutz, M. Milicic, and F. Wolter. Updating description logic ABoxes. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR-06)*, pages 46–56, 2006.
26. C. Lutz, D. Walther, and F. Wolter. Conservative extensions in expressive description logics. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 453–458, 2007.
27. T. Meyer, K. Lee, R. Booth, and J. Z. Pan. Finding maximally satisfiable terminologies for the description logic ALC. In *Proceedings of The 21st National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference (AAAI06)*, 2006.
28. S. Schlobach. Debugging and semantic clarification by pinpointing. In *Proceedings of the 2nd European Semantic Web Conference (ESWC05)*, pages 226–240, 2005.
29. S. Schlobach. Diagnosing terminologies. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI05)*, pages 670–675, 2005.
30. S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI03)*, pages 355–362, 2003.
31. B. ten Cate, W. Conradie, M. Marx, and Y. Venema. Definitorially complete description logics. In *Proc. of KR*, pages 79–89, 2006.
32. Albert Visser. Uniform interpolation and layered bisimulation. In Petr Hájek, editor, *Proceedings of Gödel'96: Logical Foundations of Mathematics, Computer Science, and Physics – Kurt Gödel's Legacy, Brno, Czech Republic*, volume 6 of *Lecture Notes Logic*, pages 139–164, Berlin, Germany, 1996. Springer Verlag.
33. K. Wang, A. Sattar, and K. Su. A theory of forgetting in logic programming. In *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 682–687. AAAI Press, 2005.
34. Z. Wang, K. Wang, R. Topor, and J. Z. Pan. Forgetting concepts in DL-Lite. In *Proceedings of 5th European Semantic Web Conference (ESWC2008)*, pages 245–257, 2008.
35. Z. Wang, K. Wang, R. Topor, J. Z. Pan, and G. Antoniou. Eliminating concepts and roles from ontologies in description logic. Technical Report. 2009.