

Forgetting for Knowledge Bases in DL-Lite_{bool}

Zhe Wang, Kewen Wang and Rodney Topor
Griffith University, Australia

Abstract

We address the problem of term elimination in DL-Lite ontologies by adopting techniques from classical forgetting theory. Specifically, we generalize our previous results on forgetting in DL-Lite_{core} TBox to forgetting in DL-Lite_{bool} KBs. We also introduce query-based forgetting, a parameterized definition of forgetting, which provides a unifying framework for defining and comparing different definitions of forgetting in DL-Lite ontologies.

1 Introduction

An ontology is a formal description of the terminological information of an application domain. An ontology is usually represented as a DL knowledge base (KB), which consists of a TBox and an ABox. As ontologies in Semantic Web applications are becoming larger and more complex, a challenge is how to construct and maintain large ontologies efficiently. Recently, ontology reuse and merging have received intensive interest, and different approaches have been proposed. Among several approaches, the forgetting operator is particularly important, which concerns the elimination of terms in ontologies. Informally, forgetting is a particular form of reasoning that allows a set of attributes 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 [Lin and Reiter, 1994; Lang *et al.*, 2003] and logic programming [Eiter and Wang, 2008; Wang *et al.*, 2005].

Efforts have also been made to define forgetting in DL-Lite [Kontchakov *et al.*, 2008; Wang *et al.*, 2008], a family of lightweight ontology languages. However, a drawback in these approaches is that forgetting is defined only for TBoxes. Although a syntactic notion of forgetting in ontologies is discussed in [Wang *et al.*, 2008], no semantic justification is provided. In most applications, an ontology is expressed as a KB, which is a pair of a TBox and an ABox. We believe that forgetting should be defined for DL-Lite KBs rather than only for TBoxes. Although 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.

In this paper, we investigate the issue of semantic forgetting for DL-Lite KBs. The main contributions of this paper can be summarized as follows:

- We introduce a model-based definition of forgetting for DL KBs. We investigate some reasoning and expressibility properties of forgetting, which are important for DL-Lite ontology reuse and merging.
- We provide a resolution-like algorithm for forgetting about concepts in DL-Lite_{bool} KBs. The algorithm can also be applied to KB forgetting in DL-Lite_{horn}, and to any specific query-based forgetting. It is proved that the algorithm is complete.
- As a general framework for defining and comparing various notions of forgetting, we introduce a parameterized forgetting based on query-answering, by which a given collection of queries determines the result of forgetting. Thus, our approach actually provides a hierarchy of forgetting for DL-Lite.

2 Forgetting in DL-Lite_{bool} Knowledge Bases

In this section, we define the operation of forgetting a signature from a DL-Lite_{bool} KB. We assume the readers' familiarity with DL-Lite_{bool}. For the syntax and semantics details of DL-Lite_{bool}, the readers should refer to [Artale *et al.*, 2007].

Let \mathcal{L} and \mathcal{L}_h , respectively, denote the languages DL-Lite_{bool} and DL-Lite_{horn}. Without special mentioning, we use \mathcal{K} to denote a KB in \mathcal{L} and \mathcal{S} a signature (a finite set of concept names and role names) in \mathcal{L} . Our model-based definition of forgetting in DL-Lite is analogous to the definition for forgetting in classical logic [Lin and Reiter, 1994; Lang *et al.*, 2003].

Let $\mathcal{I}_1, \mathcal{I}_2$ be two interpretations of \mathcal{L} . Define $\mathcal{I}_1 \sim_{\mathcal{S}} \mathcal{I}_2$ iff

1. $\Delta^{\mathcal{I}_1} = \Delta^{\mathcal{I}_2}$, and $a^{\mathcal{I}_1} = a^{\mathcal{I}_2}$ for each individual name a .
2. For each concept name A not in \mathcal{S} , $A^{\mathcal{I}_1} = A^{\mathcal{I}_2}$.
3. For each role name P not in \mathcal{S} , $P^{\mathcal{I}_1} = P^{\mathcal{I}_2}$.

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

Definition 1 We call KB \mathcal{K}' a result of model-based forgetting about \mathcal{S} in \mathcal{K} if:

- $\text{Sig}(\mathcal{K}') \subseteq \text{Sig}(\mathcal{K}) - \mathcal{S}$,
- $\text{Mod}(\mathcal{K}') = \{\mathcal{I}' \mid \exists \mathcal{I} \in \text{Mod}(\mathcal{K}) \text{ s.t. } \mathcal{I} \sim_{\mathcal{S}} \mathcal{I}'\}$.

It follows from the definition that the result of forgetting about \mathcal{S} in \mathcal{K} is unique up to KB equivalence. So we will use $\text{forget}(\mathcal{K}, \mathcal{S})$ to denote the result of forgetting about \mathcal{S} in \mathcal{K} throughout the paper.

Example 1 Let \mathcal{K} consist of the following axioms:

$Lecturer \sqsubseteq \geq 2 \text{ teaches}, \exists \text{ teaches}^- \sqsubseteq Course,$
 $\exists \text{ teaches} \sqsubseteq Lecturer \sqcup PhD,$
 $Lecturer \sqcap Course \sqsubseteq \perp, PhD \sqcap Course \sqsubseteq \perp.$
 $Lecturer(John), \text{teaches}(John, AI).$

Suppose we want to forget about $\{Lecturer, PhD\}$, then $\text{forget}(\mathcal{K}, \{Lecturer, PhD\})$ consists of the following axioms:

$\exists \text{ teaches}^- \sqsubseteq Course, \exists \text{ teaches} \sqcap Course \sqsubseteq \perp,$
 $\geq 2 \text{ teaches}(John), \neg Course(John),$
 $\text{teaches}(John, AI).$

The result of forgetting possesses several desirable properties. In particular, it preserves reasoning properties of the KB.

Proposition 1 We have

1. $\text{forget}(\mathcal{K}, \mathcal{S})$ is consistent iff \mathcal{K} is consistent.
2. for any inclusion or assertion α with $\text{Sig}(\alpha) \cap \mathcal{S} = \emptyset$, $\text{forget}(\mathcal{K}, \mathcal{S}) \models \alpha$ iff $\mathcal{K} \models \alpha$.
3. for any grounded query q with $\text{Sig}(q) \cap \mathcal{S} = \emptyset$, $\text{forget}(\mathcal{K}, \mathcal{S}) \models q$ iff $\mathcal{K} \models q$.

The following property is useful for ontology partial reuse and merging.

Proposition 2 Let $\mathcal{K}_1, \mathcal{K}_2$ be two \mathcal{L} -KBs. Suppose $\text{Sig}(\mathcal{K}_1) \cap \text{Sig}(\mathcal{K}_2) \cap \mathcal{S} = \emptyset$, then we have:

$$\text{forget}(\mathcal{K}_1 \cup \mathcal{K}_2, \mathcal{S}) \equiv \text{forget}(\mathcal{K}_1, \mathcal{S}) \cup \text{forget}(\mathcal{K}_2, \mathcal{S}).$$

An interesting special case is $\text{Sig}(\mathcal{K}_2) \cap \mathcal{S} = \emptyset$. In this case, it is safe to forget about \mathcal{S} from \mathcal{K}_1 before merging \mathcal{K}_1 and \mathcal{K}_2 .

The following proposition shows that the forgetting operation can be divided into steps.

Proposition 3 Let $\mathcal{S}_1, \mathcal{S}_2$ be two signatures. Then we have

$$\text{forget}(\mathcal{K}, \mathcal{S}_1 \cup \mathcal{S}_2) \equiv \text{forget}(\text{forget}(\mathcal{K}, \mathcal{S}_1), \mathcal{S}_2).$$

In the following part of the section, we introduce a syntactic algorithm for computing the results of forgetting about concepts in DL-Lite_{bool} KBs. The algorithm first transforms a DL-Lite_{bool} KB into an equivalent normal form, and then eliminates concepts from the KB.

We call a basic concept or its negation a *literal concept*.

Definition 2 $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is in normal form if:

- All the inclusions in \mathcal{T} are of the form $B_1 \sqcap \dots \sqcap B_m \sqsubseteq B_{m+1} \sqcup \dots \sqcup B_n$ where $0 \leq m \leq n$ and B_1, \dots, B_n are basic concepts such that $B_i \neq B_j$ for all $i < j$.
- $\mathcal{A} = \{C_1(a_1), \dots, C_s(a_s)\} \cup \mathcal{A}_r$, where $s \geq 0$, satisfies the following conditions:
 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 disjunction of conjunctions of literal concepts (DNF), for each $1 \leq i \leq s$.

We can show that every KB in \mathcal{L} can be equivalently transformed into its normal form.

Now we present in Algorithm 1 how to compute the result of forgetting about a set of concept names in a \mathcal{L} -KB.

Algorithm 1 (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: $\text{forget}(\mathcal{K}, \mathcal{S})$.

Method:

Step 1. Transform \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 $\perp \sqsubseteq A$ 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 $\perp \sqsubseteq C$, 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 $\text{forget}(\mathcal{K}, \mathcal{S})$.

Figure 1: Forget concepts in a DL-Lite_{bool} KB.

In Algorithm 1, Step 2 generates all the inclusions in $\text{forget}(\mathcal{K}, \mathcal{S})$ in a resolution-like manner. Step 3 is to make the specification of Step 4 simpler. In Step 4, each positive occurrence of A in the ABox is replaced by its ‘supersets’, and each negative occurrence by its ‘subsets’.

Algorithm 1 always terminates. The following theorem shows that it is sound and complete with respect to the semantic definition of forgetting.

Theorem 1 Let \mathcal{S} be a set of concept names. Then $\text{forget}(\mathcal{K}, \mathcal{S})$ is expressible in \mathcal{L} , and Algorithm 1 always returns $\text{forget}(\mathcal{K}, \mathcal{S})$.

Suppose \mathcal{S} is fixed. When the input \mathcal{K} is in normal form, Algorithm 1 takes only polynomial time to compute $\text{forget}(\mathcal{K}, \mathcal{S})$.

3 Query-Based Forgetting for DL-Lite Knowledge Bases

In this section, we introduce a parameterized definition of forgetting for DL-Lite_{bool} KBs, which can be used as a unifying framework for forgetting in DL-Lite_{bool} KBs.

We assume that \mathcal{Q} is a query language for DL-Lite_{bool}. The notion of query is very general here. A query can be an assertion, an inclusion, or even a formula in a logic language.

Definition 3 (query-based forgetting) We call KB \mathcal{K}' a result of \mathcal{Q} -forgetting about \mathcal{S} in \mathcal{K} if the following three conditions are satisfied:

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

The above definition implicitly introduces a family of forgetting in the sense that each query language determines a notion of forgetting for DL-Lite KBs.

The results of \mathcal{Q} -forgetting are not necessarily unique. We denote the set of all results of \mathcal{Q} -forgetting about \mathcal{S} in \mathcal{K} as $\text{Forget}^{\mathcal{Q}}(\mathcal{K}, \mathcal{S})$.

As model-based forgetting requires preserving model equivalence, it is easy to see that 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_{bool}.

Theorem 2 We have

1. $\text{forget}(\mathcal{K}, \mathcal{S}) \in \text{Forget}^{\mathcal{Q}}(\mathcal{K}, \mathcal{S})$;
2. for each $\mathcal{K}' \in \text{Forget}^{\mathcal{Q}}(\mathcal{K}, \mathcal{S})$, $\text{forget}(\mathcal{K}, \mathcal{S}) \models \mathcal{K}'$.

This theorem shows that Algorithm 1 can also be used to compute a result of \mathcal{Q} -forgetting for any \mathcal{Q} .

Now we consider how to characterize model-based forgetting by query-based forgetting. The results of model-based forgetting may not be expressible in DL-Lite_{bool}. The major reason for this, as our efforts show, is that DL-Lite_{bool} does not have a construct to represent the cardinality of a concept.

For this reason, we extend \mathcal{L} to \mathcal{L}^c by introducing new concepts of the form $\geq n \text{ u.C}$, where C is a \mathcal{L} -concept and n is a natural number. Given an interpretation \mathcal{I} , $(\geq n \text{ u.C})^{\mathcal{I}} = \Delta^{\mathcal{I}}$ if $\sharp(C^{\mathcal{I}}) \geq n$ and $(\geq n \text{ u.C})^{\mathcal{I}} = \emptyset$ if $\sharp(C^{\mathcal{I}}) \leq n - 1$, where $\sharp(S)$ denotes the cardinality of set S .

We are interested in the query language $\mathcal{Q}_{\mathcal{L}^c}^c$, which is the set of concept inclusions and (negated) assertions in \mathcal{L}^c , and possibly their unions. We can show that $\mathcal{Q}_{\mathcal{L}^c}^c$ -forgetting is equivalent to model-based forgetting.

Theorem 3 KB \mathcal{K}' is a result of $\mathcal{Q}_{\mathcal{L}^c}^c$ -forgetting about \mathcal{S} in \mathcal{K} iff $\mathcal{K}' \equiv \text{forget}(\mathcal{K}, \mathcal{S})$.

Example 2 Recall the KB \mathcal{K} in Example 1. We have $\text{forget}(\mathcal{K}, \{\text{teaches}\})$ is not expressible in \mathcal{L} . However, it is expressible in \mathcal{L}^c , and it consists of the following axioms:

- $\text{Lecturer} \sqcap \text{Course} \sqsubseteq \perp$, $\text{PhD} \sqcap \text{Course} \sqsubseteq \perp$,
 $\text{Lecturer} \sqsubseteq \geq 2 \text{ u.Course}$,
 $\text{Lecturer}(\text{John}), \text{Course}(\text{AI})$.

4 Related Work and Conclusions

The works that are close to this paper are [Kontchakov *et al.*, 2008; Wang *et al.*, 2008]. One major difference of our work from them is that we investigate forgetting for KBs while [Kontchakov *et al.*, 2008; Wang *et al.*, 2008] are only restricted to TBoxes. Such an extension (from TBoxes to KBs) is non-trivial because the involvement of ABoxes

makes things more complex. This can be seen from the algorithms and proofs of corresponding results. Our parameterized forgetting generalizes the definition of forgetting (uniform interpolation) in [Kontchakov *et al.*, 2008] in two ways (although it is not technically hard): (1) our definition is defined for KBs and (2) our definition is defined for arbitrary query languages.

Conservative extension and module extraction have some similarity with forgetting but they are different in that the first two approaches support only removing axioms, but cannot modify them. Update and erasure operations in DL-Lite are discussed in [Giacomo *et al.*, 2007]. Although both erasure [Giacomo *et al.*, 2007; Liu *et al.*, 2006] 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} . Whereas forgetting about A in \mathcal{K} involves eliminating all logical relations (e.g., subsumption relation, membership relation, etc.) in \mathcal{K} that refer to A .

We plan to work in different directions including: (1) To identify a query language that can balance computational complexity of the corresponding notion of forgetting and requirements for forgetting from practical applications, such as ontology reuse, merging and update. (2) To establish a general framework of forgetting for more expressive DLs in terms of query-based forgetting.

References

- [Artale *et al.*, 2007] A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyashev. DL-Lite in the light of first-order logic. In *Proc. 22nd AAAI*, pages 361–366, 2007.
- [Eiter and Wang, 2006] T. Eiter and K. Wang. Forgetting and conflict resolving in disjunctive logic programming. In *Proc. 21st AAAI*, pages 238–243, 2006.
- [Eiter and Wang, 2008] T. Eiter and K. Wang. Semantic forgetting in answer set programming. *Artificial Intelligence*, 14:1644–1672, 2008.
- [Giacomo *et al.*, 2007] G. De Giacomo, M. Lenzerini, A. Poggi, and R. Rosati. On the approximation of instance level update and erasure in description logics. In *Proc. 22nd AAAI*, pages 403–408, 2007.
- [Kontchakov *et al.*, 2008] R. Kontchakov, F. Wolter, and M. Zakharyashev. Can you tell the difference between DL-Lite ontologies? In *Proc. 11th KR*, pages 285–295, 2008.
- [Lang *et al.*, 2003] J. Lang, P. Liberatore, and P. Marquis. Propositional independence: Formula-variable independence and forgetting. *J. Artif. Intell. Res. (JAIR)*, 18:391–443, 2003.
- [Lin and Reiter, 1994] F. Lin and R. Reiter. Forget it. In *Proc. AAAI Fall Symposium on Relevance*, pages 154–159, 1994.
- [Liu *et al.*, 2006] H. Liu, C. Lutz, M. Milicic, and F. Wolter. Updating description logic ABoxes. In *Proc. KR*, pages 46–56, 2006.
- [Wang *et al.*, 2005] K. Wang, A. Sattar, and K. Su. A theory of forgetting in logic programming. In *Proc. 20th AAAI*, pages 682–687. AAAI Press, 2005.
- [Wang *et al.*, 2008] Z. Wang, K. Wang, R. Topor, and J. Z. Pan. Forgetting concepts in DL-Lite. In *Proc. 5th ESWC2008*, pages 245–257, 2008.