Inconsistency-tolerant Forgetting in DL-lite

Peng Xiao, Kewen Wang, and Zhe Wang

Griffith University, Australia

1 Introduction

In ontology engineering, it is essential to develop techniques for module extraction, ontology reuse and ontology comparison. Since it was introduced in DL-Lite [1], *forgetting* has become into a major method of extracting modules in DL ontologies. It has been applied in some other domains such as ontology comparison [2]. Given a (consistent) ontology $\mathcal{K}$ and a set of concepts and roles F, the result of forgetting is a new ontology $\mathcal{K}'$ that does not contain symbols in F and preserves certain reasoning or query answering tasks. In existing approaches, forgetting is defined for only *consistent* ontologies. However, ontologies can be error-prone and inconsistent.

To our best knowledge, the problem of establishing a framework for inconsistent ontologies has not been investigated yet.

In order to provide a definition of forgetting for a possibly inconsistent DL ontology, we could first repair it and then perform standard forgetting on the repaired ontology. However, there are some issues with this naive approach. In particular, it can give unwanted results for some ontologies.

Example 1. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T} = \{A \sqsubseteq D, B \sqsubseteq C, A \sqsubseteq \neg B\}$ and $\mathcal{A} = \{A(a), B(a)\}$. Then $\mathcal{K}$ has two repairs (or maximal consistent components) $\{A(a)\}$ and $\{B(a)\}$.

The forgetting results of B in these repairs are $\mathcal{K}_1 = \langle \mathcal{T}', \{A(a)\} \rangle, \mathcal{K}_2 = \langle \mathcal{T}', \{C(a)\} \rangle$, respectively, where $\mathcal{T}' = \{A \sqsubseteq D\}$. Under the semantics IAR for inconsistency-tolerant querying, $\mathcal{K} \not\models_{IAR} C(a)$, $\mathcal{K} \not\models_{AR} C(a)$, and $\mathcal{K} \models_{CAR} D(a)$. Apparently, these querying results are not satisfied in $\mathcal{K}_1$ or $\mathcal{K}_2$.

In this paper, we propose a general framework of forgetting for inconsistent ontologies and study its properties. Then we present an algorithm for computing the forgetting and its implementation based on the graph database system Neo4j. Our experimental results show that the system prototype is efficient and able to compute the result of large inconsistent ontologies in seconds or at most one minute.

2 Preliminaries

DL-lite An ontology in DL-lite is a pair $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ is a set of inclusion axioms and $\mathcal{A}$ is a set of assertions. The concepts and roles of DL-lite$_{core}$ are defined by

$$B \rightarrow A \mid \exists R, \quad R \rightarrow P \mid P^{-}, \quad C \rightarrow B \mid \neg B$$

where A is a concept name and R is a role name.

An inclusion axiom in DL-lite$_{core}$ is of the form $B \sqsubseteq C$. An assertion is of the form $A(a)$ or $R(a, b)$, where a and b are individuals.

The semantics of DL-lite is omitted here.

Inconsistency-tolerant query answering We consider four inconsistency-tolerant semantics AR, IAR, CAR, ICAR [5]. For a possibly inconsistent pair of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$, a *repair* $\mathcal{A}'$ of $\mathcal{A}$ w.r.t. $\mathcal{T}$ is a maximal subset of $\mathcal{A}$ such that $\mathcal{T} \cup \mathcal{A}'$ is consistent. Let $\mathsf{repair}_{\mathcal{T}}(\mathcal{A})$ consist of all the repairs of $\mathcal{A}$ w.r.t. $\mathcal{T}$, and $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ consist of all the ABox assertions α s.t. $\mathcal{T} \cup \mathcal{A}' \models \alpha$ for some $\mathcal{A}' \in \mathsf{repair}_{\mathcal{T}}(\mathcal{A})$. The entailment of a Boolean conjunctive query (BCQ) q according to a inconsistency-tolerant semantics γ is defined in Table 2.

γ	$\mathcal{T} \cup \mathcal{A} \models_\gamma q$ if
AR	for each $\mathcal{A}' \in \mathsf{repair}_{\mathcal{T}}(\mathcal{A})$, $\mathcal{T} \cup \mathcal{A}' \models q$
IAR	$\mathcal{T} \cup \bigcap_{\mathcal{A}' \in \mathsf{repair}_{\mathcal{T}}(\mathcal{A})} \mathcal{A}' \models q$
CAR	for each $\mathcal{A}' \in \mathsf{repair}_{\mathcal{T}}(\mathsf{clc}_{\mathcal{T}}(\mathcal{A}))$, $\mathcal{T} \cup \mathcal{A}' \models q$
ICAR	$\mathcal{T} \cup \bigcap_{\mathcal{A}' \in \mathsf{repair}_{\mathcal{T}}(\mathsf{clc}_{\mathcal{T}}(\mathcal{A}))} \mathcal{A}' \models q$

Table 1. Inconsistency-tolerant semantics for query answering

3 Inconsistency-tolerant Forgetting

In this section, we introduce a definition of forgetting for inconsistent KBs and present two useful properties. Our definition is quite general and actually applies to any DLs.

The signature of a TBox $\mathcal{T}$, denoted $\mathsf{sig}(\mathcal{T})$ is the set of concept names and role names that occur in $\mathcal{T}$. This notion extends to TBox axioms, ABox assertions, ABoxes and KBs. As with inconsistency-tolerant query answering approaches, we assume the TBox is consistent. For a DL $\mathcal{L}$ and a signature Σ, TBox $\mathcal{T}'$ is a result of $\mathcal{L}$-forgetting about Σ in $\mathcal{T}$, if $\mathcal{T}' \models \alpha$ iff $\mathcal{T} \models \alpha$ for each TBox axiom α in $\mathcal{L}$ over $\mathsf{sig}(\mathcal{T}) \setminus \Sigma$.

Definition 1. *For a DL $\mathcal{L}$, a (possible inconsistent) KB $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ in $\mathcal{L}$, a signature Σ and an inconsistency-tolerant semantics γ, a consistent KB $\mathcal{K}' = \mathcal{T}' \cup \mathcal{A}'$ is a result of γ-forgetting about Σ in $\mathcal{K}$ if (1) $\mathsf{sig}(\mathcal{K}') \subseteq \mathsf{sig}(\mathcal{K}) \setminus \Sigma$, (2) $\mathcal{T}'$ is a result of $\mathcal{L}$-forgetting about Σ in $\mathcal{T}$, and (3) for any BCQ q over $\mathsf{sig}(\mathcal{K}) \setminus \Sigma$, $\mathcal{K}' \models q$ iff $\mathcal{K} \models_\gamma q$.*

A result of γ-forgetting is a consistent KB that has exactly the same logical consequences (i.e., TBox axioms and BCQs) as the initial KB (under γ semantics) over the remaining signature. The definition does not suggest the existence of forgetting. In fact it is well known that $\mathcal{L}$-forgetting does not always exist for basic DLs such as $\mathcal{EL}$ and $\mathcal{ALC}$. Yet we show that γ-forgetting always exists for DL-Lite$_{core}$.

Theorem 1. *For a KB $\mathcal{K}$ in DL-lite$_{core}$ and a signature Σ, a result of γ-forgetting about Σ in $\mathcal{K}$ exists for any inconsistency-tolerant semantics γ.*

For DL-lite$_{core}$ KBs, it is efficient to compute their IAR and ICAR forgettings.

Theorem 2. *In DL-Lite$_{core}$, the computation of γ-forgetting is in P for γ being IAR or ICAR with respect to data complexity.*

The intuition behind this result is that forgetting under IAR and ICAR can be computed by removing all the conflicts (i.e., facts that lead to inconsistency) from the KB and then perform the standard definition of forgetting for consistent KBs. These two steps can be done in polynomial time.

4 Implementation and Experimental Results

Because of the linear property of axioms in DL-lite$_{core}$, DL-lite$_{core}$ KBs can be stored in directed binary graphs, thus some reasoning tasks can be naturally captured by traversals of such graphs. We have implemented a prototype system with Neo4j[1] for computing IAR concept forgetting in DL-lite$_{core}$, using a graph-based algorithm. Specifically, given a DL-lite$_{core}$ KB $\mathcal{K}$, and a set Σ of concepts to forget from $\mathcal{K}$, we first transfer $\mathcal{K}$ into a directed graph $G_{\mathcal{K}}$ where each node of $G_{\mathcal{K}}$ represents a general concept or an individual of $\mathcal{K}$, and each edge of $G_{\mathcal{K}}$ represents an inclusion or membership assertion of $\mathcal{K}$ [4]. Then the graph is processed in two steps:

1. For every pair of contradicting concepts in $G_{\mathcal{K}}$, we calculate individual nodes that have access to both of them, thus we obtain all the membership edges that represent conflicts of $\mathcal{K}$, which are then removed from $G_{\mathcal{K}}$.
2. For nodes representing concepts in Σ, we build edges between their predecessors and successors, and then remove all edges related to them.

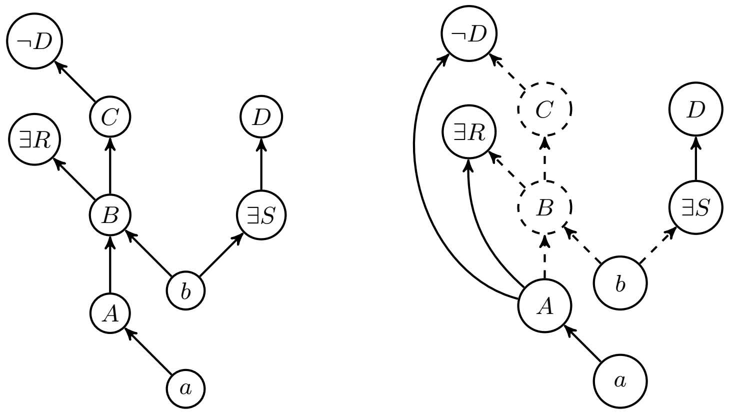

Fig. 1. (left) A DL-lite$_{core}$ KB stored in a graph $G_{\mathcal{K}}$ (right) $G_{\mathcal{K}}$ after IAR Forgetting about $\{B, C\}$

We use a modified version of LUBM with auto-generated inconsistent ABox data [3] as our benchmark. Different sets of concept names with size of 40 are randomly

[1] https://neo4j.com/

selected to be forgot from the KBs in the modified LUBM, average times that are taken by Step 1, Step 2 and the whole process, and sizes of the results of forgetting are recorded. All experiments are conducted on a server with Intel Xeon E5-1603 2.80 GHz CPU and 16GB of RAM, running Linux mint 17 OS, and Java 1.8 with 6GB.

Some experimental results are shown in Table 4, where $\mathcal{A}_n^m$ denotes an ABox containing the data of n universities and inconsistencies added with probability of m.

For the largest ABox with up to 2 million assertions, our algorithm can still solve the problem in around one minute. In all cases, the size of the resulting KB is considerably larger than the original one, which is not surprising because a large number of concepts related to facts are forgotten. We can also see that the ratio of inconsistencies contributes little to the cost of computation.

Abox data	original size	forgetting size	step 1 (s)	step 2 (s)	Total (s)
$\mathcal{A}_{20}^{15e-4}$	1982922	2630679	52.6	10.8	63.4
$\mathcal{A}_{20}^{5e-2}$	2014129	2651328	53.7	11.2	64.9
$\mathcal{A}_{20}^{2e-1}$	2103366	2736531	57.3	9.9	67.2

Table 2. IAR-forgetting of LUBM KBs

5 Future Issues

We plan to adapt the proposed approach to some other non-standard reasoning supports for ontologies, such as query explanation (abduction) [6].

Acknowledgement

This work was partially supported by the Australian Research Council (ARC) under grant DP130102302.

References

1. Wang, Z., Wang, K., Topor, R., Pan, J. Z. Forgetting concepts in DL-Lite. In *Proc. 5th ESWC*, pages 245-257, 2008.
2. Kontchakov, R., Wolter, F., Zakharyaschev, M. Logic-based ontology comparison and module extraction, with an application to DL-Lite. *Artif. Intell.*, 174:1093–1141, 2010.
3. Bienvenu, M., Bourgaux, C., Goasdoué, F. Querying inconsistent description logic knowledge bases under preferred repair semantics. In *Proc. 28th AAAI*, pages 996–1002, 2014.
4. Qi, G., Wang, Z., Wang, K., Fu, X., Zhuang, Z. Approximating model-based ABox revision in DL-Lite: Theory and Practice. In *Proc. 29th AAI*, pages 254–260, 2015.
5. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M. Inconsistency-tolerant semantics for description logics. In *Proc. 4th RR*, pages 103–117, 2010.
6. Du, J., Wang, K., Shen, Y. D. Towards tractable and practical ABox abduction over inconsistent description logic ontologies. In *Proc. 29th AAAI*, pages 1489–1495, 2015.