Uniform Interpolation for $\mathcal{ALC}$ Revisited*

Zhe Wang[1], Kewen Wang[1], Rodney Topor[1], Jeff Z. Pan[2], and Grigoris Antoniou[3]

[1] Griffith University, Australia
[2] University of Aberdeen, UK
[3] University of Crete, Greece

Abstract. The notion of uniform interpolation for description logic $\mathcal{ALC}$ has been introduced in [9]. In this paper, we reformulate the uniform interpolation for $\mathcal{ALC}$ from the angle of forgetting and show that it satisfies all desired properties of forgetting. Then we introduce an algorithm for computing the result of forgetting in concept descriptions. We present a detailed proof for the correctness of our algorithm using the Tableau for $\mathcal{ALC}$. Our results have been used to compute forgetting for $\mathcal{ALC}$ knowledge bases.

1 Introduction

The Web Ontology Language (OWL) [15] provides a construct *owl:imports* for importing and merging Web ontologies by referencing axioms contained in another ontology that may be located somewhere else on the Web. This construct is very limited in that it can only merge some linked ontologies together but is unable to resolve conflicts among merged ontologies or to filter redundant parts from those ontologies. However, an ontology is often represented as a logical theory, and the removal of one term may influence other terms in the ontology. Thus, more advanced methods for dealing with large ontologies and reusing existing ontologies are desired.

It is well-known that OWL is built on description logics (DLs) [1]. Recent efforts show that the notions of *uniform interpolation* and *forgetting* are promising tools for extracting modular ontologies from a large ontology. In a recent experiment reported in [5], uniform interpolation and forgetting have been used for extracting modular ontologies from two large medical ontologies SNOMED CT [16] and NCI [17]. SNOMED CT contains around 375,000 concept definitions while NCI Thesaurus has 60,000 axioms. The experiment result is promising. For instance, if 2,000 concepts definitions are forgotten from SNOMED CT, the success rate is 93% and if 5,000 concepts definitions are forgotten from NCI, the success rate is 97%.

Originally, *interpolation* is proposed and investigated in pure mathematical logic, specifically, in proof theory. Given a theory T, *ordinary interpolation* for T says that if $T \vdash \phi \rightarrow \psi$ for two formulas ϕ and ψ, then there is a formula $I(\phi, \psi)$ in the language containing only the shared symbols, say S, such that $T \vdash \phi \rightarrow I(\phi, \psi)$ and $T \vdash I(\phi, \psi) \rightarrow \psi$. *Uniform interpolation* is a strengthening of ordinary interpolation in that the interpolant can be obtained from either ϕ and S or from ψ and S. Uniform

* This work was partially supported by the Australia Research Council (ARC) Discovery Project 0666107.

interpolation for various propositional modal logics have been investigated by Visser [10] and Ghilardi [3]. A definition of uniform interpolation for the description logic $\mathcal{ALC}$ is given in [9] and it is used in investigating the definability of TBoxes for $\mathcal{ALC}$.

On the other hand, (semantic) forgetting is studied by researchers in AI [8, 7, 2]. Informally, given a knowledge base K in classical logic or nonmonotonic logic, we may wish to forget about (or discard) some redundant predicates but still preserve certain forms of reasoning. Forgetting has been investigated for DL-Lite and extended $\mathcal{EL}$ in [11, 6, 5] but not for expressive DLs. Forgetting for modal logic is studied in [13].

Forgetting and uniform interpolation have different intuitions behind them and are introduced by different communities. Uniform interpolation is originally investigated as a syntactic concept and forgetting is a semantic one. However, if the axiom system is sound and complete, they can be characterized by each other.

In this paper, we first reformulate the notion of uniform interpolation for $\mathcal{ALC}$ studied in [9] from the angle of forgetting and show that all desired properties of forgetting are satisfied. We introduce an algorithm for computing the result of forgetting in concept descriptions and, a novel and detailed proof for the correctness of the algorithm is developed using the Tableau for $\mathcal{ALC}$. We note that a similar algorithm for uniform interpolation is provided in [9] in which it is mentioned that the correctness of their algorithm can be shown using a technique called *bisimulation* that is widely used in modal logic[4]. In a separate paper [12], we use the results obtained in this paper to compute forgetting for $\mathcal{ALC}$ knowledge bases.

Due to space limitation, proofs are omitted in this paper but can be found at `http://www.cit.gu.edu.au/~kewen/Papers/alc_forget_long.pdf`.

2 Preliminaries

We briefly recall some basics of $\mathcal{ALC}$. Further details of $\mathcal{ALC}$ and other DLs can be found in [1].

First, we introduce the syntax of *concept descriptions* for $\mathcal{ALC}$.

Elementary concept descriptions consist of both *concept names* and *role names*. So a concept name is also called *atomic concept* while a role name is also called *atomic role*. Complex concept descriptions are built inductively as follows: A (atomic concept); $\top$ (universal concept); $\bot$ (empty concept); $\neg C$ (negation); $C \sqcap D$ (conjunction); $C \sqcup D$ (disjunction); $\forall R.C$ (universal quantification) and $\exists R.C$ (existential quantification). Here, A is an (atomic) concept, C and D are concept descriptions, and R is a role.

An interpretation $\mathcal{I}$ of $\mathcal{ALC}$ 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 $\Delta^{\mathcal{I}}$ and each atomic role R with a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The function $\cdot^{\mathcal{I}}$ can be naturally extended to complex descriptions:

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}} \qquad \bot^{\mathcal{I}} = \emptyset$$
$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} - C^{\mathcal{I}} \qquad (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$$
$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$

[4] An email communication with one of the authors of [9] shows that they have not got a complete proof yet.

$(\forall R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \ : \ \forall b.(a,b) \in R^{\mathcal{I}} \text{ implies } b \in C^{\mathcal{I}}\}$
$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \ : \ \exists b.(a,b) \in R^{\mathcal{I}} \text{ and } b \in C^{\mathcal{I}}\}$

An interpretation $\mathcal{I}$ satisfies an *inclusion* (or *subsumption*) of the form $C \sqsubseteq D$ where C, D are concept descriptions, denoted $\mathcal{I} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. We write $\models C \sqsubseteq D$ if $\mathcal{I} \models C \sqsubseteq D$ for all $\mathcal{I}$. Similarly, $\models C \equiv D$ is an abbreviation of $\models C \sqsubseteq D$ and $\models D \sqsubseteq C$. $\mathcal{I}$ satisfies an assertion of the form form $C(a)$ or $R(a,b)$, where a and b are individuals, C is a concept description and R is a role name, if, respectively, $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

The signature of a concept description C, written $\mathsf{sig}(C)$, is the set of all concept and role names in C.

3 Forgetting in Concept Descriptions

In this section, we discuss the problem of forgetting about a concept/role in description logic $\mathcal{ALC}$. In the rest of this paper, by a *variable* we mean either a concept name or a role name. Intuitively, the result C' of forgetting about a variable from a concept description C may be weaker than C (w.r.t. subsumption) but it should be as close to C as possible. For example, after the concept $Male$ is forgotten from a concept description for "Male Australian student" $C = Australians \sqcap Students \sqcap Male$, then we should obtain a concept description $C' = Australians \sqcap Students$ for "Australian student".

3.1 Semantic Forgetting

Let C be a concept description that contains a variable V. If we want to forget (or discard) V from C, intuitively, the result of forgetting about V in C will be a concept description C' that satisfies the condition: C' defines a minimal concept among all concepts that subsumes C and is irrelevant to V (i.e. V does not appear in the concept description).

Definition 3.1. *(Forgetting for concept descriptions) Let C be a concept description in $\mathcal{ALC}$ and V be a variable. A concept description C' on the signature $\mathsf{sig}(C) \setminus \{V\}$ is a* result of forgetting *about V in C if the following conditions are satisfied:*

(RF1) $\models C \sqsubseteq C'$.

(RF2) *For all concept description C'' on $\mathsf{sig}(C) \setminus \{V\}$, $\models C \sqsubseteq C''$ and $\models C'' \sqsubseteq C'$ implies $\models C'' \equiv C'$, i.e., C' is the strongest concept description weaker than C that does not contain V.*

Notice that we can forget about a set of concept names and role names in a concept description by a straightforward generalization of Definition 3.1. The above (RF1) and (RF2) correspond to the conditions (2) and (3) of Theorem 8 in [9] [5], and generalize them by allowing V to be a role name.

A fundamental property of forgetting in $\mathcal{ALC}$ concept descriptions is that the result of forgetting is unique under concept description equivalence.

[5] The correspondence between (RF2) and (3) of Theorem 8 can be seen from this: $\models C \sqsubseteq C'$ and $\models C \sqsubseteq C''$ implies $\models C \sqsubseteq C' \sqcap C''$. It implies by (RF2), $\models C' \sqcap C'' \equiv C'$, which equals $\models C' \sqsubseteq C''$ in (3).

Theorem 3.1. *For any concept description C and variable V, if C_1 and C_2 are results of forgetting about V in C, then $\models C_1 \equiv C_2$.*

As all results of forgetting are equivalent, we write $\mathsf{forget}(C, V)$ to denote an arbitrary result of forgetting about V in C.

We use the following examples of concept descriptions to illustrate our semantic definitions of forgetting for $\mathcal{ALC}$. We will introduce an algorithm later and explain how we can compute a result of forgetting through a series of syntactic transformations of concept descriptions.

Example 3.1. Suppose the concept "Research Student" is defined by $C = Student \sqcap (Master \sqcup PhD) \sqcap \exists supervised.Professor$ where "Master", "PhD" and "Professor" are all concepts; "supervised" is a role and $supervised(x, y)$ means that x is supervised by y. If the concept description C is used only for students, we may wish to forget about $Student$: $\mathsf{forget}(C, Student) = (Master \sqcup PhD) \sqcap \exists supervised.Professor$. If we do not require that a supervisor for a research student must be a professor, then the filter "Professor" can be forgotten: $\mathsf{forget}(C, Professor) = Student \sqcap (Master \sqcup PhD) \sqcap \exists supervised.\top$.

3.2 Properties of Semantic Forgetting

The semantic forgetting for description logic possesses several important properties. The following result, which is not obvious, shows that forgetting distributes over union $\sqcup$.

Proposition 3.1. *Let $C_1, \ldots, C_n$ be concept descriptions in $\mathcal{ALC}$. For any variable V, we have*

$$\mathsf{forget}(C_1 \sqcup \cdots \sqcup C_n, V) = \mathsf{forget}(C_1, V) \sqcup \cdots \sqcup \mathsf{forget}(C_n, V)$$

However, forgetting for $\mathcal{ALC}$ does not distribute over intersection $\sqcap$. For example, if the concept description $C = A \sqcap \neg A$, then $\mathsf{forget}(C, A) = \bot$, since $\models C \equiv \bot$. But $\mathsf{forget}(A, A) \sqcap \mathsf{forget}(\neg A, A) = \top$. So $\mathsf{forget}(A \sqcap \neg A, A) \neq \mathsf{forget}(A, A) \sqcap \mathsf{forget}(\neg A, A)$.

On the other hand, forgetting for $\mathcal{ALC}$ does preserve the subsumption relation between concept descriptions.

Proposition 3.2. *Let C_1 and C_2 be concept descriptions in $\mathcal{ALC}$, and $\models C_1 \sqsubseteq C_2$. For any variable V, we have $\models \mathsf{forget}(C_1, V) \sqsubseteq \mathsf{forget}(C_2, V)$.*

As a corollary of Proposition 3.2, it is straightforward to show that semantic forgetting also preserves the equivalence of concept descriptions.

Proposition 3.3. *Let C_1 and C_2 be concept descriptions in $\mathcal{ALC}$, and $\models C_1 \equiv C_2$. For any variable V, we have $\mathsf{forget}(C_1, V) \equiv \mathsf{forget}(C_2, V)$.*

Satisfiability is key reasoning task in description logics. We say a concept C is satisfiable if $C^{\mathcal{I}} \neq \emptyset$ for some interpretation $\mathcal{I}$. C is unsatisfiable if $\models C \equiv \bot$. Forgetting also preserves satisfiability of concept descriptions.

Proposition 3.4. *Let C be a concept description in $\mathcal{ALC}$, and V be a variable. Then C is satisfiable iff* $\mathsf{forget}(C, V)$ *is satisfiable.*

When we want to forget about a set of variables, they can be forgotten one by one since the ordering of forgetting is irrelevant to the result.

Proposition 3.5. *Let C be a concept description in $\mathcal{ALC}$ and let $\mathcal{V} = \{V_1, \ldots, V_n\}$ be a set of variables. Then* $\mathsf{forget}(C, \mathcal{V}) = \mathsf{forget}(\mathsf{forget}(\mathsf{forget}(C, V_1), V_2), \ldots), V_n)$.

The next result shows that forgetting distributes over quantifiers.

Proposition 3.6. *Let C be a concept description in $\mathcal{ALC}$, R be a role name and V be a variable. Then*
(1) $\mathsf{forget}(\forall V.C, V) = \top$ *where V is a role name, and* $\mathsf{forget}(\forall R.C, V) = \forall R.\mathsf{forget}(C, V)$ *if $V \neq R$;*
(2) $\mathsf{forget}(\exists V.C, V) = \top$ *where V is a role name, and* $\mathsf{forget}(\exists R.C, V) = \exists R.\mathsf{forget}(C, V)$. *if $V \neq R$.*

4 Computing the Result of Forgetting

In this section we introduce an intuitive algorithm for computing the result of forgetting through rewriting of concept descriptions (syntactic concept transformations). Given a concept description C and a variable V, this algorithm consists of two stages: (1) C is first transformed into an equivalent disjunctive normal form (DNF), which is a disjunction of conjunctions of simple concept descriptions; (2) the result of forgetting about V in each such simple concept description is obtained by removing some parts of the conjunct.

We call an atomic concept A or its negation $\neg A$ a *literal concept* or simply *literal*. A *pseudo-literal* with role R is a concept description of the form $\exists R.F$ or $\forall R.F$, where R is a role name and F is an arbitrary concept description. A *generalized literal* is either a literal or a pseudo-literal.

First, each arbitrary concept description can be equivalently transformed into a disjunction of conjunctions of generalized literals. This basic DNF for $\mathcal{ALC}$ is well known in the literature [1] and thus the details of the transformation are omitted here.

Definition 4.1. *(Basic DNF) A concept description D is in* basic disjunctive normal form *or* basic DNF *if $D = \bot$ or $D = \top$ or D is a disjunction of conjunctions of generalized literals $D = D_1 \sqcup \cdots \sqcup D_n$, where each $D_i \not\equiv \bot$ and D_i is of the form*

$\bigsqcap L \sqcap \bigsqcap_{R \in \mathcal{R}} [\forall R.U_R \sqcap \bigsqcap_k \exists R.E_R^{(k)}]$

where each L is a literal, $\mathcal{R}$ is the set of role names that occur in D_i, $k \geq 0$, and each U_R or $E_R^{(k)}$ is a concept description in basic DNF.

The reason for transforming a concept description into its basic DNF is that forgetting distributes over $\sqcup$ (Proposition 3.1). When a concept description is in its basic DNF, we only need to compute the result of forgetting in a conjunction of generalized literals. It can be shown that the result of forgetting about an atomic concept A in a conjunction B of literals can be obtained just by extracting A (or $\neg A$) from the

conjuncts (extracting a conjunct equals replacing it by $\top$). Unlike classical logics and DL-Lite, the basic DNF in $\mathcal{ALC}$ is not clean in that a generalized literal (i.e. pseudo-literal) can still be very complex. When C is a conjunction containing pseudo-literals, it is not straightforward to compute the result of forgetting about A in C. For example, let $C = \forall R.A \sqcap \forall R.\neg A$. Through simple transformation we can see $C \equiv \forall R.\bot$ is the result of forgetting about A in C, while simply extracting A and $\neg A$ results in $\top$. A similar example is when $C = \forall R.(A \sqcup B) \sqcap \exists R.(\neg A \sqcup B)$, the result of forgetting is $\exists R.B$ rather than $\exists R.\top$ (note that $\forall R.C_1 \sqcap \exists R.C_2 \sqsubseteq \exists R.(C_1 \sqcap C_2)$). For this reason, the following key step in obtaining a DNF is required for computing forgetting:

$$\forall R.C_1 \sqcap \exists R.C_2 \rightsquigarrow \forall R.C_1 \sqcap \exists R.(C_1 \sqcap C_2)$$

By applying this transformations to a concept description in its basic DNF, each $\mathcal{ALC}$ concept description can be transformed into the DNF as defined below.

Definition 4.2. *(Disjunctive Normal Form or DNF)*

A concept description D is in disjunctive normal form *if $D = \bot$ or $D = \top$ or D is a disjunction of conjunctions of generalized literals $D = D_1 \sqcup \cdots \sqcup D_n$, where each $D_i \not\equiv \bot$ ($1 \leq i \leq n$) is a conjunction $\bigsqcap L$ of literals or in the form of*

$$\bigsqcap L \sqcap \bigsqcap_{R \in \mathcal{R}} \Big[\forall R.U_R \sqcap \bigsqcap_{k} \exists R.(E_R^{(k)} \sqcap U_R) \Big]$$

where each L is a literal, $\mathcal{R}$ is the set of role names that occur in D_i, $k \geq 0$, and each U_R or $E_R^{(k)} \sqcap U_R$ is a concept description in DNF.

For convenience, each D_i is called a *normal conjunction* in this paper. The disjunctive normal form is a bit different from the normal form in [1] but they are essentially the same.

Once a concept D is in the normal form, the result of forgetting about a variable V in D can be obtained from D by simple symbol manipulations.

Obviously, the major cost of Algorithm 1 is from transforming the given concept description into its normal form. For this reason, the algorithm is exponential time in the worst case. However, if the input concept description C is in DNF, Algorithm 1 takes only linear time (w.r.t. the size of C).

Example 4.1. Given a concept $D = (A \sqcup \exists R.\neg B) \sqcap \forall R.(B \sqcup C)$, we want to forget about concept name B in D. In Step 1 of Algorithm 1, D is firstly transformed into its DNF $D' = [A \sqcap \forall R.(B \sqcup C)] \sqcup [\forall R.(B \sqcup C) \sqcap \exists R.(\neg B \sqcap C)]$. Note that $\exists R.(\neg B \sqcap C)$ is transformed from $\exists R.[\neg B \sqcap (B \sqcup C)]$. Then in Step 2, each occurrence of B in D' is replaced by $\top$, and $\forall R.(\top \sqcup F)$ is replaced with $\top$. We obtain $\mathsf{CForget}(D, B) = A \sqcup \exists R.C$. To forget about role R in D, Algorithm 1 replaces each pseudo-literals in D' of the form $\forall R.F$ or $\exists R.F$ with $\top$, and returns $\mathsf{CForget}(D, R) = \top$.

Indeed we can prove that Algorithm 1 is sound and complete w.r.t. the semantic definition of forgetting for $\mathcal{ALC}$ in Definition 3.1.

Theorem 4.1. *Let V be a variable and C a concept description in $\mathcal{ALC}$. Then*

$$\models \mathsf{CForget}(C, V) \equiv \mathsf{forget}(C, V).$$

Algorithm 1
Input: A concept description C in $\mathcal{ALC}$ and a variable V in C.
Output: The result of forgetting about V in C.
Method $\mathsf{CForget}(C, V)$:
Step 1. Transform C into its DNF D. If D is $\top$ or $\bot$, return D; otherwise, let $D = D_1 \sqcup \cdots \sqcup D_n$ as in Definition 4.2.
Step 2. For each conjunct E in each D_i, perform the following transformations:

- if (V is a concept name and) E is a literal equals V or $\neg V$, replace E with $\top$;
- if (V is a role name and) E is a pseudo-literal of the form $\forall V.F$ or $\exists V.F$, replace E with $\top$;
- if E is a pseudo-literal in the form of $\forall R.F$ or $\exists R.F$ where $R \neq V$, replace F with $\mathsf{CForget}(F, V)$, and replace each resulting $\forall S.(\top \sqcup G)$ with $\top$.

Step 3. Return the resulting concept description as $\mathsf{CForget}(C, V)$.

Fig. 1. Forgetting in concept descriptions.

Theorem 4.1 and Proposition 3.6 can be immediately derived from some lemmas in the next section and the validity of these lemmas is established by using the Tableau for $\mathcal{ALC}$.

5 Proof of Theorem 4.1

Proofs of Proposition 3.6 and Theorem 4.1 (i. e. the correctness of Algorithm 1) heavily rely on the Tableau theory for $\mathcal{ALC}$ and thus we first briefly introduce it.

Given two concept descriptions C_0 and D_0, the Tableau theory states that $\models C_0 \sqsubseteq D_0$ iff no (finite) interpretation $\mathcal{I}$ can be constructed such that $\mathcal{I}$ satisfies concept $C_0 \sqcap \neg D_0$, i.e., there is an individual x_0 with $x_0^{\mathcal{I}} \in (C_0 \sqcap \neg D_0)^{\mathcal{I}}$. Equivalently, ABox $\mathcal{A}_0 = \{(C_0 \sqcap \neg D_0)(x_0)\}$ must be inconsistent for an arbitrary individual x_0. And by transforming $C_0 \sqcap \neg D_0$ into its Negation Normal Form (NNF) and applying Tableau transformation rules to the ABox, *clashes* of the form $A(x), \neg A(x)$ must occur.

The $\mathcal{ALC}$ Tableau transformation rules are as follows:
- $\{(C_1 \sqcap C_2)(x), \ldots\} \rightarrow_{\sqcap} \{(C_1 \sqcap C_2)(x), C_1(x), C_2(x), \ldots\}$.
- $\{(C_1 \sqcup C_2)(x), \ldots\} \rightarrow_{\sqcup} \{(C_1 \sqcup C_2)(x), C_1(x), \ldots\}$ and $\{(C_1 \sqcup C_2)(x), C_2(x), \ldots\}$.
- $\{(\exists R.C)(x), \ldots\} \rightarrow_{\exists} \{(\exists R.C)(x), R(x, y), C(y), \ldots\}$.
- $\{(\forall R.C)(x), R(x, y)\} \rightarrow_{\forall} \{(\forall R.C)(x), R(x, y), C(y)\}$.

Conversely, if clashes occur in each resulting ABox after applying Tableau rules, then $\mathcal{A}_0$ must be inconsistent and $\models C_0 \sqsubseteq D_0$ holds.

Before presenting the proofs, we first show a useful lemma, whose correctness can be immediately obtained using the Tableau.

Lemma 5.1. *Let C_i's be concepts and R a role name. Then, for every $j (1 \leq j \leq m)$ or $C_j = \bot$, $\models \forall R.(\bigsqcup_{i=1}^{m} C_i) \sqsubseteq \forall R.C_j \sqcup \bigsqcup_{i \neq j}^{m} \exists R.C_i$.*

Lemma 5.2. *Let U, E_i's be concepts with $\models E_i \sqsubseteq U$, and R a role name. Denote $D = \forall R.U \sqcap \bigsqcap \exists R.E_i$ with $\not\models D \equiv \bot$. Suppose $D' = \bigsqcup \forall R.C_j \sqcup \exists R.C \sqcup \bigsqcup L_k$, where C and C_j's are concepts, and L_k's are generalized literals not containing R.*

Then $\models D \sqsubseteq D'$ implies that at least one of the following three holds:
(1) $\models D' \equiv \top$; or
(2) $\models E_i \sqsubseteq C$ for some i, and $\models D \sqsubseteq \exists R.C \sqsubseteq D'$; or
(3) $\models U \sqsubseteq C \sqcup C_j$, and $\models D \sqsubseteq \forall R.(C \sqcup C_j) \sqsubseteq D'$ for some j.

Proof For simplicity, we discuss the case of $m = 2$ and $n = 2$, that is, $D = \forall R.U \sqcap \exists R.E_1 \sqcap \exists R.E_2$ and $D' = \forall R.C_1 \sqcup \forall R.C_2 \sqcup \exists R.C \sqcup \bigsqcup L_k$. Other cases can be proved in the same way.

According to the Tableau, the ABox $\{D(x), \neg D'(x)\}$ is inconsistent, which can be transformed through Tableau rules into $\{\ (\forall R.U)(x),\ (\exists R.E_1)(x),\ (\exists R.E_2)(x),\ (\exists R.\neg C_1)(x),\ (\exists R.\neg C_2)(x),\ (\forall R.\neg C)(x),\ \neg(\bigsqcup L_k)(x)\ \}$

It can be further transformed into

$$\begin{aligned}\mathcal{A} = \{\ & R(x, y_1),\ E_1(y_1),\ R(x, y_2),\ E_2(y_2),\ R(x, z_1),\ \neg C_1(z_1),\ R(x, z_2),\\ & \neg C_2(z_2),\ (\forall R.U)(x),\ U(y_1),\ U(y_2),\ U(z_1),\ U(z_2),\ (\forall R.\neg C)(x),\\ & \neg C(y_1),\ \neg C(y_2),\ \neg C(z_1),\ \neg C(z_2),\ \neg(\textstyle\bigsqcup L_k)(x),\ \dots\ \}.\end{aligned}$$

Note that L_k's do not contain any pseudo-literal of the form $\forall R.F$ or $\exists R.F$. Thus there is no way to generate any new assertions about y_i or z_j from $\neg(\bigsqcup L_k)(x)$ ($i, j = 1, 2$). Neither can $R(x, v)$ with $v \neq y_i$ and $v \neq z_j$ be generated from $\mathcal{A}$. This means no Tableau rule is applicable to $R(x, y_i)$, $R(x, z_j)$, $(\forall R.U)(x)$ or $(\forall R.\neg C)(x)$ any more. Thus we can ignore those assertions.

According to the Tableau, $\mathcal{A}$ must be inconsistent for arbitrary instances x, y_1, y_2, z_1, z_2. Thus it is safe to assume that x, y_1, y_2, z_1, z_2 represent different individuals. Thus $\mathcal{A}$ can be written as

$$\begin{aligned}&\{\ \neg(\textstyle\bigsqcup L_k)(x)\ \} \cup \{\ E_1(y_1),\ U(y_1),\ \neg C(y_1)\ \} \cup \{\ E_2(y_2),\ U(y_2),\ \neg C(y_2)\ \}\\ &\cup\{\ U(z_1),\ \neg C_1(z_1),\ \neg C(z_1)\ \} \cup \{\ U(z_2),\ \neg C_2(z_2),\ \neg C(z_2)\ \}.\end{aligned}$$

Consider three cases:
Case 1. $\{\neg(\bigsqcup L_k)(x)\}$ is inconsistent, then we have $\models \bigsqcup L_k \equiv \top$. That is, $\models D' \equiv \top$.
Case 2. $\{E_i(y_i), U(y_i), \neg C(y_i)\}$ ($i = 1$ or 2) is inconsistent, then $\models E_i \sqcap U \sqsubseteq C$. From $\models E_i \sqsubseteq U$, it follows that $\models E_i \sqsubseteq C$. Thus $\models D \sqsubseteq \exists R.E_i \sqsubseteq \exists R.C \sqsubseteq D'$.
Case 3. $\{U(z_j), \neg C_j(z_j), \neg C(z_j)\}$ ($j = 1$ or 2) is inconsistent, then $\models U \sqsubseteq C \sqcup C_j$. Thus, by Lemma 5.1, we have $\models D \sqsubseteq \forall R.(C \sqcup C_j) \sqsubseteq \forall R.C_j \sqcup \exists R.C \sqsubseteq D'$. ■

Now we can show a general property of forgetting w.r.t. quantifiers. Proposition 3.6 is just a special case of the following lemma.

Lemma 5.3. *Let U, E_i's be concepts with $\models E_i \sqsubseteq U$, R be a role name, and V be a variable. Denote $C = \forall R.U \sqcap \bigsqcap_{i \in M} \exists R.E_i$ where $M = \{1, \dots, m\}$ is a set of natural numbers.*

Suppose $\not\models C \equiv \bot$. If $V = R$, then $\mathsf{forget}(C, V) = \top$. Otherwise, we have $\mathsf{forget}(C, V) = \forall R.\mathsf{forget}(U, V) \sqcap \bigsqcap_{i \in M} \exists R.\mathsf{forget}(E_i, V)$.

Proof Set $C' = \forall R.\mathsf{forget}(U, V) \sqcap \bigsqcap_{i \in M} \exists R.\mathsf{forget}(E_i, V)$.
(CF1): Obviously, $\models C \sqsubseteq \top$. And we have $\models C \sqsubseteq C'$, since $\models \forall R.U \sqsubseteq \forall R.\mathsf{forget}(U, V)$ and $\models \exists R.E_i \sqsubseteq \exists R.\mathsf{forget}(E_i, V)$ for all i.
(CF2): Suppose that D is a concept not containing V and $\models C \sqsubseteq D$. We want to prove $\models \top \sqsubseteq D$ for $V = R$, and otherwise, $\models C' \sqsubseteq D$.

Let $D = \bigsqcap_{k \in N} D_k$, and every D_k is of the form $\bigsqcup \forall R.U'_j \sqcup \exists R.E' \sqcup B'$ where E' and U'_j's are concepts, B' is a disjunction of generalized literals not containing R. From $\models C \sqsubseteq D$, we have $\models C \sqsubseteq D_k$ for all k.

If $V = R$, then each D_k contains no disjunct of $\forall R.U'_j$, and $\models E' \equiv \bot$. By Lemma 5.2, we have $\models D_k \equiv \top$ for each k. In this case, $\models \top \sqsubseteq D$. We have shown in this case, $\mathsf{forget}(C, V) = \top$.

Otherwise, suppose for some D_k, it does not contain any occurrence of R. In this case, $\models D_k \equiv \top$, and we can remove D_k from the conjunction. In what follows, we assume D_k contains R and $\not\models D_k \equiv \top$ for each $k \in N$.

By Lemma 5.2, for some D_k's in D (denoted as $k \in K \subseteq N$), we always have some E_i ($i \in M$) in C such that $\models E_i \sqsubseteq F_{\mathcal{E}_k}$ where $\exists R.F_{\mathcal{E}_k}$ is the existential quantified disjunct of D_k, and thus $\models C \sqsubseteq \exists R.F_{\mathcal{E}_k} \sqsubseteq D_k$. For the other D_k's with $k \in N - K$, we always have $\models U \sqsubseteq F_{\mathcal{U}_k}$ and $\models C \sqsubseteq \forall R.F_{\mathcal{U}_k} \sqsubseteq D_k$ for some concept $F_{\mathcal{U}_k}$ not containing V. This is to say, we can always find
$D' = \bigsqcap_{k \in K} \exists R.F_{\mathcal{E}_k} \sqcap \bigsqcap_{l \in N-K} \forall R.F_{\mathcal{U}_l}$
such that D' does not contain V, and $\models C \sqsubseteq D' \sqsubseteq D$.

By the definition of K, for each $F_{\mathcal{E}_k}$ ($k \in K$), we always have some E_i ($i \in M$) in C such that $\models E_i \sqsubseteq F_{\mathcal{E}_k}$. By the definition of c-forgetting, $\models \mathsf{forget}(E_i, V) \sqsubseteq F_{\mathcal{E}_k}$. That is, for each $k \in K$, there always exists some $i \in M$ such that $\models \exists R.\mathsf{forget}(E_i, V) \sqsubseteq \exists R.F_{\mathcal{E}_k}$. This implies $\models \bigsqcap_{i \in M} \exists R.\mathsf{forget}(E_i, V) \sqsubseteq \bigsqcap_{k \in K} \exists R.F_{\mathcal{E}_k}$. Similarly, we have $\models \forall R.\mathsf{forget}(U, V) \sqsubseteq \forall R.\bigsqcap_{l \in N-K} F_{\mathcal{U}_l}$. Thus, we can conclude that
$\models \forall R.\mathsf{forget}(U, V) \sqcap \bigsqcap_{i \in M} \exists R.\mathsf{forget}(E_i, V) \sqsubseteq \forall R.\bigsqcap_{l \in N-K} F_{\mathcal{U}_l} \sqcap \bigsqcap_{k \in K} \exists R.F_{\mathcal{E}_k}$.
which is, $\models C' \sqsubseteq D'$. Hence, we have $\models C' \sqsubseteq D$. ■

Similar to the above lemma, we can show the following result. For a literal L, we use L^+ to denote the concept name in L.

Lemma 5.4. *Let C be a disjunct in DNF such that $C = \bigsqcap L_i \sqcap \bigsqcap_{R \in \mathcal{R}} C_R$, where each L is a literal, concept C_R is of the form $\forall R.U \sqcap \bigsqcap \exists R.E_k$ with $\models E_k \sqsubseteq U$ for each k. Then we have,* $\mathsf{forget}(C, V) = \bigsqcap_{L_i^+ \neq V} L_i \sqcap \bigsqcap_{R \in \mathcal{R}} \mathsf{forget}(C_R, V)$.

Since forgetting is distributive over disjunction, Theorem 4.1 is proven.

6 Conclusion

We have looked into the concept of uniform interpolation for $\mathcal{ALC}$ from the angle of variable forgetting. As a result, a theory of forgetting in $\mathcal{ALC}$ concept descriptions is developed, in which forgetting can be done for both concepts and roles. As well as several important properties, we have developed algorithms for computing results of

forgetting and provide a novel proof for the correctness of the algorithm w.r.t. the semantic definition of forgetting. Forgetting for $\mathcal{ALC}$ concept descriptions has been implemented in C^{++} as a new component of the DL reasoner FaCT++ [14] and it is available at `http://www.cit.gu.edu.au/~kewen/DLForget/`. Such a forgetting component can be used by an ontology editor to enhance its ability to partially reuse existing ontologies and thus provides a flexible tool for tailoring large ontologies. Although semantic forgetting can be easily adapted to most DLs, it is not straightforward to generalize the algorithms for computing forgetting to other expressive DLs.

References

1. F. Baader, D. Calvanese, D.McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2002.
2. T. Eiter and K. Wang. Semantic forgetting in answer set programming. *Artificial Intelligence*, 172(14): 1644–1672, 2008.
3. S. Ghilardi. An algebraic theory of normal forms. In *Annals Pure Appl. Logic*, 71(3):189–245, 1995.
4. B. Konev, D. Walther, and F. Wolter. The logical difference problem for description logic terminologies. In *Proc. IJCAR'08*, pp.259-274, 2008.
5. B. Konev, D. Walther, and F. Wolter. Forgetting and uniform interpolation in large-scale description logic terminologies. In *Proc. IJCAI'09*, 2009.
6. R. Kontchakov, F. Wolter, and M. Zakharyaschev. Can you tell the difference between DL-Lite ontologies? In *Proc. KR'08*, pp.285–295, 2008.
7. J. Lang, P. Liberatore, and P. Marquis. Propositional independence: Formula-variable independence and forgetting. *J. Artif. Intell. Res.*, 18:391–443, 2003.
8. F. Lin and R. Reiter. Forget it. In *Proc. AAAI Fall Symposium on Relevance*, pp.154–159, 1994.
9. B. ten Cate, W. Conradie, M. Marx, and Y. Venema. Definitorially complete description logics. In *Proc. KR'06*, pp.79–89, 2006.
10. A. Visser. Uniform interpolation and layered bisimulation. In *Proc. of Gödel'96*, pp.139–164, 1996.
11. Z. Wang, K. Wang, R. Topor, and J. Z. Pan. Forgetting concepts in DL-Lite. In *Proc. ESWC'08*, pp.245–257, 2008.
12. Z. Wang, K. Wang, R. Topor, J. Z. Pan, and G. Antoniou. Concept and Role Forgetting in $\mathcal{ALC}$ Ontologies. In *Proc. ISWC'09*, pp.666–681, 2009.
13. Y. Zhang and Y. Zhou. Knowledge Forgetting: Properties and Applications. In *Artificial Intelligence*, 173(2009): 1525–1537, 2009.
14. D. Tsarkov and I. Horrocks. Description logic reasoner: System description. In *Proc. IJCAR'06*, pp.292–297, 2006.
15. S. Bechhofer *et al*. OWL (Web Ontology Language) Reference, 2004. `http://www.w3.org/TR/2004/REC-owl-ref-20040210/`
16. Systematized Nomenclature of Medicine-Clinical Terms. `http://www.ihtsdo.org/snomed-ct/`
17. NCI Wiki. `http://ncicb.nci.nih.gov/`