Fine-Grained Forgetting for the Description Logic ALC

Mostafa Sakr, Renate A. Schmidt

The University of Manchester, Manchester, UK

Abstract

Forgetting is an important ontology extraction technique. A variant of forgetting which has received significant attention in the literature is *deductive forgetting* (or *uniform interpolation*). While *deductive forgetting* is attractive as it generates the forgetting view in a language with the same complexity as the language of the original ontology, it is known that with a slightly extended target language using definer symbols more information can be preserved. In this paper, we study *deductive forgetting* of concept names with the aim of understanding the unpreserved information. We present a system that performs *deductive forgetting* and produces a set Δ of axioms representing the unpreserved information in the forgetting view. Our system allows a new fine-grained ontology extraction process that gives the user the option to enhance the informativeness of the deductive forgetting view by appending to it axioms from Δ.

1. Introduction

Forgetting is an important ontology extraction technique. It eliminates from an ontology a given subset of its vocabulary. The result is a focused ontology, called *forgetting view*, which preserves the content of the ontology relative to the non-forgotten vocabulary. Forgetting offers solutions for many applications such as: *computing logical difference* [1], *information hiding* [2], *abduction* [3], *resolving conflicts* [4], *relevance* [5, 6, 7], and *forgetting actions in planning* [8].

A variant of forgetting that has been studied in the literature is *deductive forgetting* (or *uniform interpolation*). Given an $\mathcal{ALC}$ ontology and a forgetting signature, deductive forgetting produces a view of the ontology which preserves only the information expressible in $\mathcal{ALC}$ over the non-forgetting signature [9, 10, 11]. Deductive forgetting is however not precise, because information expressible with more expressivity may not be preserved. Yet, deductive forgetting remains an appealing variant of forgetting because when performed on $\mathcal{ALC}$ ontologies, the generated forgetting views are (infinitely) representable in $\mathcal{ALC}$, or finitely representable if fixpoint operators are allowed [12, 13].

When a deductive view is computed, the following questions arise: *Does the deductive view preserve all information of the non-forgotten vocabulary? If not, what information is not preserved and what does this information represent? Can some of this information be partially preserved, i.e., can a view more informative than the deductive forgetting view be computed?* In this paper, we

DL 2022: 35th International Workshop on Description Logics, August 7–10, 2022, Haifa, Israel

Mostafa.Sakr@manchester.ac.uk (M. Sakr); Renate.Schmidt@manchester.ac.uk (R. A. Schmidt)
0000-0003-4296-5831 (M. Sakr); 0000-0002-6673-3333 (R. A. Schmidt)

aim to address these questions, gain a better understanding of the information not preserved by deductive forgetting, and provide a practical tool to compute this information. We focus on concept forgetting for ontologies in the description logic $\mathcal{ALC}$, the basic logic in the family of expressive description logics [14].

The main contribution is a novel forgetting method. (1) The method converts the input ontology into an intermediate ontology in which the forgetting signature has been eliminated. This intermediate ontology is semantically equivalent to the input ontology with respect to the non-forgotten vocabulary. That is, the reducts of the models of both ontologies to the non-forgotten vocabulary coincide. The intermediate ontology may use foreign concept symbols, or definers, to represent subsets of role successors. (2) The method obtains from the intermediate ontology two sets $\mathcal{O}^{red}$ and Δ of axioms. The set $\mathcal{O}^{red}$ approximates the deductive view by allowing the use of foreign symbols. We present a method to eliminate these foreign symbols from $\mathcal{O}^{red}$ and obtain the final deductive view in $\mathcal{ALC}$. Complete elimination of the foreign symbols may not however succeed when the deductive view does not exist finitely in $\mathcal{ALC}$ due to cycles occurring over forgetting symbols. The set Δ represents the *information difference* between the intermediate ontology and $\mathcal{O}^{red}$. If all foreign symbols are successfully eliminated from $\mathcal{O}^{red}$, then Δ also represents the information difference between the input ontology and the deductive view.

Several benefits are obtained from our forgetting method. (1) In two different evaluations an implementation of our forgetting method was compared against the state-of-the-art deductive forgetting tool Lethe [15]. Our implementation was found to be faster than Lethe. Our analysis shows that this improvement can be attributed to a novelty of the language of the intermediate ontology as it allows for avoiding time-consuming operations performed by Lethe. (2) By inspecting the set Δ, we now understand the difference between the input ontology and the deductive view as information on the conjunctions of different subsets of role successors. (3) An empty Δ indicates that the deductive view coincides semantically with the input ontology with respect to the non-forgotten vocabulary. (4) By incrementing $\mathcal{O}^{red}$ with axioms from Δ, our method allows for a fine-grained forgetting framework where views of the original ontology that are more informative than the deductive view can be obtained based on user requirements.

All proofs are provided in the long version https://github.com/e73898ms/FineGrainedForgetting/blob/main/Fine_Grained_Forgetting-Long%20Version-DL22.pdf.

2. History and Related Work

Forgetting can be traced back to Boole who referred to it as *elimination of the middle terms*. In propositional logic, it was studied in relation to *relevance, independence,* and *variable elimination* [6, 7]. A variant of forgetting which preserves semantic equivalences is *semantic forgetting*. In the context of first-order logic (FOL), semantic forgetting was viewed as a *second-order quantifier elimination problem* [16, 17] finding that the semantic view of a FO theory is not in general expressible in FOL but is always expressible in second-order logic (SOL). A way to view the foreign symbols in the intermediate ontology created by our method is as second-order existentially quantified concept symbols. This is because they are used to represent subsets of role successors whose first-order definition, as we show, cannot be computed in general.

Therefore, our intermediate ontology adheres to results in the literature [9, 16], and can be viewed as an approximation to the semantic view of the input ontology. We show that standard reasoning operations can be performed on the intermediate ontology using standard $\mathcal{ALC}$ reasoning methods.

Deductive forgetting was considered in [18] under the name *weak forgetting*. The proposal in [18] builds on previous work in modal logics which views deductive forgetting as *uniform interpolation* [19, 20], i.e., forgetting a signature $\mathcal{F}$ from an ontology $\mathcal{O}$ is equivalent to computing the uniform interpolant over the remaining vocabulary of $\mathcal{O}$ after excluding $\mathcal{F}$. This allows characterizing the relationship between the original ontology and the deductive view in terms of *bisimulation* over the non-forgotten symbols [10, 9, 21]. Deductive and semantic forgetting are also closely related to the notions of *concept inseparability* and *model inseparability* [22, 23, 24, 25]. Several deductive forgetting methods were proposed in [13, 20, 1].

Deciding the existence of a finitely representable deductive view is 2ExpTime, and its size is, at most, triple exponential in the size of the original ontology [10]. For $\mathcal{ALC}$ ontologies, the deductive view can be captured, possibly infinitely, in $\mathcal{ALC}$. Infinite forgetting views occur when cycles over some forgetting symbols exist [13]. In this case, finite representations may be approximated by fixpoint operators [12, 13, 26, 27] or by using foreign symbols to witness these cycles [13]. The latter representation can be converted to the former [13].

3. Basic Definitions

Let N_c, N_r be two disjoint sets of concept symbols and role symbols. Concepts in $\mathcal{ALC}$ are of the following forms: $\bot \mid A \mid \neg C \mid C \sqcap D \mid \exists r.C$ where $A \in N_c, r \in N_r$ and C and D are $\mathcal{ALC}$ concepts. We also allow the following abbreviations: $\top \equiv \neg\bot, \forall r.C \equiv \neg\exists r.\neg C, C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$. An interpretation in $\mathcal{ALC}$ is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ where the domain $\Delta^{\mathcal{I}}$ is a nonempty set and $\cdot^{\mathcal{I}}$ is an interpretation function that assigns to each concept symbol $A \in N_c$ a subset of $\Delta^{\mathcal{I}}$ and to each $r \in N_r$ a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The language constructs are interpreted as follows: $\bot^{\mathcal{I}} := \emptyset, \quad (\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}, (C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}, (\exists r.C)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} | \exists y : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$.

A TBox, or an ontology, is a set of axioms of the form $C \sqsubseteq D$, where C and D are concepts. $\mathcal{I}$ is model of an ontology $\mathcal{O}$ if all axioms $C \sqsubseteq D \in \mathcal{O}$ are true in $\mathcal{I}$, in symbols $\mathcal{I} \models C \sqsubseteq D$. And, $\mathcal{I} \models C \sqsubseteq D$ if and only if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. We say that $C \sqsubseteq D$ is satisfiable with respect to $\mathcal{O}$ if and only if $\mathcal{I} \models C \sqsubseteq D$ for some model $\mathcal{I}$ of $\mathcal{O}$. We also say that $C \sqsubseteq D$ is a consequence of $\mathcal{O}$, in symbols $\mathcal{O} \models C \sqsubseteq D$, if and only if $\mathcal{I} \models C \sqsubseteq D$ for every model $\mathcal{I}$ of $\mathcal{O}$.

Let C be an $\mathcal{ALC}$ concept, we denote by $sig(C)$ the set of concept and role symbols appearing in C. For an ontology $\mathcal{O}$, $sig(\mathcal{O}) = \bigcup_{C \sqsubseteq D \in \mathcal{O}} sig(C) \cup sig(D)$. The size of an ontology is the number of axioms in it.

Definition 1. *Two models $\mathcal{I}$ and $\mathcal{J}$* Σ-coincide *iff $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$ and $p^{\mathcal{I}} = p^{\mathcal{J}}$ for every concept or role symbol $p \in \Sigma$.*

Definition 2. *Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be two ontologies and Σ a set of symbols where $\Sigma \subseteq N_c \cup N_r$. We say $\mathcal{O}_1$ and $\mathcal{O}_2$ are* semantically Σ-equivalent*, in symbols $\mathcal{O}_1 \equiv^{\mathcal{M}}_{\Sigma} \mathcal{O}_2$, iff for every model $\mathcal{I}_1$ of $\mathcal{O}_1$ there is a model $\mathcal{I}_2$ of $\mathcal{O}_2$, and vice versa, such that $\mathcal{I}_1$ and $\mathcal{I}_2$ Σ-coincide.*

Resolution (Res)

$$\frac{C_1 \sqcup A \quad C_2 \sqcup \neg A}{C_1 \sqcup C_2}$$

where A is a forgetting symbol and C_1, C_2 are general concept expressions.

Figure 1: Binary resolution rule

Definition 3. *Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be two $\mathcal{ALC}$ ontologies, and let Σ a set of symbols where $\Sigma \subseteq N_c \cup N_r$. We say $\mathcal{O}_1$ and $\mathcal{O}_2$ are* deductively Σ-equivalent*, in symbols $\mathcal{O}_1 \equiv^C_\Sigma \mathcal{O}_2$, iff for every $\mathcal{ALC}$ concept inclusion α, where $sig(\alpha) \subseteq \Sigma$, we have $\mathcal{O}_1 \models \alpha$ iff $\mathcal{O}_2 \models \alpha$.*

Deductive forgetting is defined using deductive equivalence [10].

Definition 4. *Let $\mathcal{O}$ be an $\mathcal{ALC}$ ontology, and let $\mathcal{F} \subseteq sig(\mathcal{O}) \cap N_c$ be a forgetting signature. An ontology $\mathcal{V}$ is a* deductive forgetting view *of $\mathcal{O}$ w.r.t. $\mathcal{F}$ iff $sig(\mathcal{V}) \subseteq sig(\mathcal{O}) \backslash \mathcal{F}$, and $\mathcal{O} \equiv^C_{sig(\mathcal{O})\backslash\mathcal{F}} \mathcal{V}$.*

4. Computing the Intermediate Ontology

The first stage of our method is to compute the intermediate ontology $\mathcal{O}^{int}$ of the input ontology $\mathcal{O}$ w.r.t. the given forgetting signature $\mathcal{F}$. The method applies resolution to the input ontology written in clausal form. $\mathcal{O}^{clausal}$ is computed by: (1) converting $\mathcal{O}$ into negation normal form (NNF), with negation applied only to concept names, (2) miniscoping, i.e., replacing $\exists r.C \sqcup \exists r.D$ with the semantically equivalent $\exists r.(C \sqcup D)$, and $\forall r.C \sqcap \forall r.D$ with the semantically equivalent $\forall r.(C \sqcap D)$, (3) applying structural transformations to extract the formulas under role restriction that contain the forgetting symbols by introducing fresh concept symbols (called definers) [28], and (4) converting the result to conjunctive normal form (CNF).

Example 1. *Consider the axiom $A \sqsubseteq \exists r.(B \sqcap C)$ where B is a forgetting symbol. It is first converted to NNF by eliminating the connective $\sqsubseteq$, giving $S_1 = \{\neg A \sqcup \exists r.(B \sqcap C)\}$. Structural transformation is applied to extract $B \sqcap C$, giving $S_2 = \{\neg A \sqcup \exists r.D_1, \neg D_1 \sqcup (B \sqcap C)\}$ where $D_1 \in N_d$ is a definer symbol. Finally, S_2 is converted to CNF, giving $S_3 = \{\neg A \sqcup \exists r.D_1, \neg D_1 \sqcup B, \neg D_1 \sqcup C\}$.*

The forgetting symbols in $\mathcal{F}$ are then eliminated from $\mathcal{O}^{clausal}$ by iteratively eliminating them using the *Resolution* rule in Figure 1. When all possible resolution inferences have been performed on a concept symbol in $\mathcal{F}$, clauses that contain this concept symbol are removed in a *purity deletion* step. Additionally, the following operations are applied eagerly: (1) Tautology deletion: clauses of the form $C \sqcup \neg C \sqcup E$ are deleted, where C and E are $\mathcal{ALC}$ concepts. (2) Purification: if a forgetting symbol A occurs only positively or only negatively in $\mathcal{O}$, then A is replaced everywhere by $\top$ and $\bot$ respectively. Assume $\mathcal{O}^{int}$ is the set of clauses that remain.

Example 2. *Let $\mathcal{O} = \{A \sqsubseteq \forall r.B \sqcap \forall s.\neg B, G \sqsubseteq \exists r.(\neg B \sqcup C), B \sqsubseteq H\}$, and $\mathcal{F} = \{B\}$. The method starts by generating $\mathcal{O}^{clausal} = \{\neg A \sqcup \forall r.D_1, \neg A \sqcup \forall s.D_2, \neg G \sqcup \exists r.D_3, \neg D_1 \sqcup B, \neg D_2 \sqcup \neg B, \neg D_3 \sqcup \neg B \sqcup C, \neg B \sqcup H\}$, where D_1, D_2, and D_3 are fresh definers. Then, it resolves on the concept symbol B using the* Resolution *rule in Figure 1 generating additionally the clauses $\{\neg D_1 \sqcup \neg D_2, \neg D_1 \sqcup \neg D_3 \sqcup C, \neg D_1 \sqcup H\}$. Finally, the clauses $\{\neg D_1 \sqcup B, \neg D_2 \sqcup \neg B, \neg D_3 \sqcup \neg B \sqcup C, \neg B \sqcup H\}$ are removed by purity deletion. So the intermediate ontology $\mathcal{O}^{int}$ is $\{\neg A \sqcup \forall r.D_1, \neg A \sqcup \forall s.D_2, \neg G \sqcup \exists r.D_3, \neg D_1 \sqcup \neg D_2, \neg D_1 \sqcup \neg D_3 \sqcup C, \neg D_1 \sqcup H\}$.*

Theorem 1. $\mathcal{O} \equiv^{\mathcal{M}}_{sig(\mathcal{O})\setminus\mathcal{F}} \mathcal{O}^{int}$.

Theorem 2. *The size of $\mathcal{O}^{int}$ is in the worst case exponential in the size of the given ontology $\mathcal{O}$ and double exponential in the number of forgetting symbols.*

Definers are used in the intermediate ontology $\mathcal{O}^{int}$ to represent subsets of role successors. Precise definitions of definers cannot always be given without knowledge of the forgetting symbols. For instance in Example 1, D_1 is interpreted as $D_1^{\mathcal{I}} = \{y \in B^{\mathcal{I}} \cap C^{\mathcal{I}} \mid (x, y) \in r^{\mathcal{I}}, x \in A^{\mathcal{I}}\}$ where $\mathcal{I}$ is a model of S. Since the user of the forgetting view may not be aware of B, a definition of D_1 in terms of B is not conveyed in $\mathcal{O}^{int}$. Thus, definers may be viewed as second-order existentially quantified concept symbols because they represent some subsets of the domain whose definitions cannot be precisely captured. One can observe that $\mathcal{O}^{int}$ can be viewed as an approximation to the semantic forgetting view of $\mathcal{O}$ with respect to $\mathcal{F}$.

5. Extracting Δ and $\mathcal{O}^{red}$

The second stage of the method is to obtain the sets Δ and $\mathcal{O}^{red}$ from $\mathcal{O}^{int}$, where $\mathcal{O}^{red}$ is an ontology approximating the deductive view and Δ is the information difference between $\mathcal{O}^{int}$ and $\mathcal{O}^{red}$. The *Reduction* rule in Figure 2 removes from $\mathcal{O}^{int}$ the clauses with two or more negative definers. These clauses constitute the set Δ.

The *Role Propagation* rule in Figure 2 computes the $\mathcal{ALC}$ consequences that are otherwise lost when removing the clauses of Δ from $\mathcal{O}^{int}$ by the *Reduction* rule. Therefore, we require it to be applied before removing these clauses. The premises of the *Role Propagation* rule start with the clause $P_0 \sqcup C_0$, where P_0 takes the form $\neg D_0 \sqcup \neg D_1 \sqcup \cdots \sqcup \neg D_n$. The second premise is a set of clauses $P_j \sqcup C_j$. Here, the concepts P_j takes the same form as the concept P_0, i.e., is a disjunction of negative definers, but also Definers$(P_j) \subseteq$ Definers(P_0) where Definers(P) denotes the set of definer symbols in $sig(P)$. The intuition here is that $P_j \sqsubseteq P_0$. Therefore, every domain element that is not in the interpretation of P_0, consequently P_j, must be in the interpretation of C_0 and C_j. The clauses in the third and the fourth premises take the same form, except that existential role restriction is only allowed in the third premise. By the third and fourth premises, every domain element must be in the interpretation of $\bigsqcup_{i=0}^{n} E_i$ or $\mathcal{Q}r.(\bigsqcap_{i=0}^{n} D_i)$. But the latter can be rewritten as $\mathcal{Q}r.\neg P_0$, which is subsumed by $\mathcal{Q}r.(\bigsqcap_{j=0}^{m} C_j)$ as concluded by the rule.

Example 3. *Continuing with Example 2, the* Role Propagation *rule applies with its four premises being:*

1. $P_0 \sqcup C_0 = \neg D_1 \sqcup \neg D_3 \sqcup C$

Role Propagation

$$\frac{P_0 \sqcup C_0,\ \bigcup_{j=1}^{m}\{P_j \sqcup C_j\}, E_0 \sqcup \mathcal{Q}r.D_0,\ \bigcup_{i=1}^{n}\{E_i \sqcup \forall r.D_i\}}{(\bigsqcup_{i=0}^{n} E_i) \sqcup \mathcal{Q}r.(\bigsqcap_{j=0}^{m} C_j)}$$

where $P_0 = \bigsqcup_{i=0}^{n} \neg D_i$, P_j is any sub-concept of P_0, $\mathcal{Q} \in \{\exists, \forall\}$, and C_0 and C_j do not contain a definer.

Reduction

$$\frac{\mathcal{O} \cup \{\neg D_1 \sqcup ... \sqcup \neg D_n \sqcup C\}}{\mathcal{O}}$$

where C is a concept expression that does not contain a negative definer, $D_1, ..., D_n$ are definer symbols, and $n \geq 2$.

Figure 2: $\mathcal{ALC}$ reduction rules.

2. $\bigcup_{j=1}^{m}\{P_j \sqcup C_j\} = \{\neg D_1 \sqcup H\}$
3. $E_0 \sqcup \mathcal{Q}r.D = \neg G \sqcup \exists r.D_3$
4. $\bigcup_{i=1}^{n}\{E_i \sqcup \forall r.D_i\} = \{\neg A \sqcup \forall r.D_1\}$

The conclusion is $\neg A \sqcup \neg G \sqcup \exists r.(C \sqcap H)$. Note that the generated conclusion preserves information that would otherwise be lost when the clause $\neg D_1 \sqcup \neg D_3 \sqcup C$ is removed by the Reduction *rule.*

After the *Role Propagation* rule has been exhaustively applied, the clauses of Δ are removed. The remaining clauses constitute $\mathcal{O}^{red}$.

Denote by $\mathcal{O}^{rp}$ the ontology obtained from $\mathcal{O}^{int}$ by applying the *Role Propagation* rule.

Example 4. *Continuing with Example 2. We have $\mathcal{O}^{rp} = \mathcal{O}^{int} \cup \{\neg A \sqcup \neg G \sqcup \exists r.(C \sqcap H)\}$ where the axiom on the right of the union operator is the conclusion of the* Role Propagation *rule obtained in Example 3. We also have $\Delta = \{\neg D_1 \sqcup \neg D_2, \neg D_1 \sqcup \neg D_3 \sqcup C\}$, and $\mathcal{O}^{red} = \{\neg A \sqcup \forall r.D_1, \neg A \sqcup \forall s.D_2, \neg G \sqcup \exists r.D_3, \neg D_1 \sqcup H, \neg A \sqcup \neg G \sqcup \exists r.(C \sqcap H)\}$.*

Theorem 3. *$\mathcal{O}^{rp} \equiv^{C}_{sig(\mathcal{O})\backslash\mathcal{F}} \mathcal{O}^{red}$.*

Theorem 3 proves a main contribution of the paper. First, observe that $\mathcal{O}^{rp} \equiv^{\mathcal{M}}_{sig(\mathcal{O})\backslash\mathcal{F}} \mathcal{O}^{int}$. It follows from this observation, and Theorems 1 and 3 that $\mathcal{O} \equiv^{C}_{sig(\mathcal{O})\backslash\mathcal{F}} \mathcal{O}^{red}$. Therefore, if $\mathcal{O}^{ui}$ is a deductive view of $\mathcal{O}^{red}$ with respect to $sig(\mathcal{O})\backslash\mathcal{F}$, then $\mathcal{O}^{red}$ is deductively equivalent to $\mathcal{O}^{ui}$ with respect to $sig(\mathcal{O})\backslash\mathcal{F}$. We shall strengthen this in the next section and show that if no cycles occur in $\mathcal{O}^{red}$ then it is semantically equivalent to the deductive view with respect to $sig(\mathcal{O})\backslash\mathcal{F}$.

Second, observe that $\mathcal{O}^{rp} = \mathcal{O}^{red} \cup \Delta$. Additionally, the clauses in Δ have been generated in $\mathcal{O}^{int}$ by resolution inferences over the forgetting symbols, and the premises of these inferences were removed by purity deletion. Therefore, we find in general that $\mathcal{O}^{red} \not\models \Delta$. This implies that Δ can be viewed as representing the information difference between $\mathcal{O}^{rp}$ and $\mathcal{O}^{red}$. Altogether we therefore conclude that Δ can be viewed as the information difference between $\mathcal{O}$ and the deductive view with respect to $sig(\mathcal{O})\backslash\mathcal{F}$.

We end this section with a discussion on the extracted set Δ which consists of clauses of the format $\neg D_1 \sqcup \cdots \sqcup \neg D_n \sqcup F$ where $n \geq 2$, or in axiom form, $D_1 \sqcap \cdots \sqcap D_n \sqsubseteq F$. Since we introduced the definer symbols to represent subsets of role successors, these clauses can be understood as information on the conjunctions of different subsets of role successors. For instance, in Example 2, the clause $\neg D_1 \sqcup \neg D_2 \in \Delta$ specifies the constraint that the subset of r-successors and the subset of s-successors of domain elements in the interpretation of A are disjoint. It was not a coincidence that we introduced definer symbols to represent subsets of role successors. Using them in this way and introducing them via structural transformation forces the clauses in Δ to be explicit members of $\mathcal{O}^{int}$ which simplifies their extraction, giving us a representation of the difference between $\mathcal{O}$ and the deductive view.

6. Eliminating the Definer Symbols

We identified Δ as the axioms that contain two or more negative definers. $\mathcal{O}^{int}$ and $\mathcal{O}^{red}$ may contain definer symbols that appear negatively in clauses where no other negative definer is present. These definers can be eliminated safely while preserving the interpretations of the non-forgotten vocabulary. For this, we use the *Definer Elimination* rule in Figure 3.

The side conditions of the *Definer Elimination* rule exclude the elimination of definers that may appear both positively and negatively in a clause. We call such definer symbols *cyclic definers.* The existence of cyclic definers signifies cycles in the original ontology over some forgetting symbols. In this case the deductive view may not exist as it requires an infinite representation.

An approach to eliminate cyclic definers and obtain a finite approximation of the deductive view is using fixpoint operators [12]. As an alternative, cyclic definers can be left in the deductive view as witnesses of these cycles [13]. We find this the best option because it defers the decision of a suitable representation to a later stage.

For clauses that contain only one negative definer symbol, possibly with other positive definers, the *Definer Elimination* rule in Figure 3 is applied exhaustively. The rule replaces the definer symbol D with its super-concept $C_1 \sqcap ... \sqcap C_n$. Note that, in the *Definer Elimination* rule, C may be $\bot$. Besides the *Definer Elimination* rule, we also eagerly apply the *Tautology Deletion* and the *Purification* rules (see Section 4).

Example 5. *Continuing with Example 4, $\mathcal{V}$ is extracted from $\mathcal{O}^{red}$ as follows:*

1. *The definers D_2 and D_3 are eliminated using* Purification. *Since D_2 and D_3 appear only positively in $\mathcal{O}^{red}$, they are purified by replacing them with $\top$ which gives $\{\neg A \sqcup \forall r.D_1, \neg A \sqcup \forall s.\top, \neg G \sqcup \exists r.\top, \neg D_1 \sqcup H, \neg A \sqcup \neg G \sqcup \exists r.(C \sqcap H)\}$. As $\forall s.\top$ evaluates to $\top$, the result can be simplified further to $\{\neg A \sqcup \forall r.D_1, \neg G \sqcup \exists r.\top, \neg D_1 \sqcup H, \neg A \sqcup \neg G \sqcup \exists r.(C \sqcap H)\}$.*

Definer Elimination

$$\frac{\mathcal{O} \cup \{\neg D \sqcup C_1, ..., \neg D \sqcup C_n\}}{\mathcal{O}[D/C]}$$

where $C = \sqcap_{i=1}^{n} C_i$ and $D \notin sig(C)$, C does not contain any negative definers, and $\mathcal{O}$ does not contain D negatively.

Figure 3: Definer elimination rule

2. *The definer D_1 is eliminated by the* Definer Elimination *rule in Figure 3 giving $\{\neg A \sqcup \forall r.H, \neg G \sqcup \exists r.\top, \neg H \sqcup H, \neg A \sqcup \neg G \sqcup \exists r.(C \sqcap H)\}$. The clause $\neg H \sqcup H$ is then eliminated by* Tautology Deletion *giving $\mathcal{V} = \{\neg A \sqcup \forall r.H, \neg G \sqcup \exists r.\top \neg A \sqcup \neg G \sqcup \exists r.(C \sqcap H)\}$*

Theorem 4. *Let $\mathcal{V}$ be generated from $\mathcal{O}^{red}$ by applying the Definer Elimination rule from Figure 3 exhaustively. Then, (1) $\mathcal{O}^{red} \equiv^{\mathcal{M}}_{sig(\mathcal{O})\setminus\mathcal{F}} \mathcal{V}$; and (2) if $sig(\mathcal{V}) \cap N_d = \emptyset$ then $\mathcal{V}$ is a deductive forgetting view of $\mathcal{O}$ w.r.t. $\mathcal{F}$.*

7. Computing More Informative Forgetting Views

Our forgetting method allows customizing the informativeness of the final forgetting view according to user requirements, which reveals a spectrum of forgetting views that are more informative than the deductive view and at most as informative as the intermediate ontology. This can be done by overriding the *Reduction* rule as illustrated in the following example.

Example 6. *Consider $\mathcal{O}^{int}$ from Example 2. Applying the rules in Figure 2 gives $\mathcal{O}^{red}$ and Δ from Example 4. We may increase the informativeness of the final forgetting view by overriding the* Reduction *rule. We describe three different forgetting views that can be generated in this way.*

1. *If we want to preserve all the information about the r-successors of A, then we override the* Reduction *rule to retain the clauses where D_1 occurs. In this case, $\Delta_1 = \emptyset$ and the final forgetting view will be $\mathcal{O}_1^{red} = \mathcal{V}_1 = \{\neg A \sqcup \forall r.D_1, \neg A \sqcup \forall s.D_2, \neg G \sqcup \exists r.D_3, \neg D_1 \sqcup \neg D_2, \neg D_1 \sqcup \neg D_3 \sqcup C, \neg A \sqcup \neg G \sqcup \exists r.C\}$.*
2. *If we want to preserve the information about the r-successors of A in relation to the r-successors of G, we override the* Reduction *rule to retain the clauses where both D_1 and D_3 occur. That is, $\Delta_2 = \{\neg D_1 \sqcup \neg D_2\}$ and $\neg D_1 \sqcup \neg D_3 \sqcup C \notin \Delta$. Then, $\mathcal{O}_2^{red} = \{\neg A \sqcup \forall r.D_1, \neg A \sqcup \forall s.D_2, \neg G \sqcup \exists r.D_3, \neg D_1 \sqcup \neg D_3 \sqcup C, \neg A \sqcup \neg G \sqcup \exists r.C\}$, and $\mathcal{V}_2 = \{\neg A \sqcup \forall r.D_1, \neg G \sqcup \exists r.D_3, \neg D_1 \sqcup \neg D_3 \sqcup C, \neg A \sqcup \neg G \sqcup \exists r.C\}$ with D_2 purified away.*
3. *If we are interested in the relation between the r and s successors of A, then we override the* Reduction *rule to remove $\neg D_1 \sqcup \neg D_3 \sqcup C$ but not $\neg D_1 \sqcup \neg D_2$. That is $\Delta_3 = \{\neg D_1 \sqcup \neg D_3 \sqcup C\}$. Consequently, we get $\mathcal{O}_3^{red} = \{\neg A \sqcup \forall r.D_1, \neg A \sqcup \forall s.D_2, \neg G \sqcup \exists r.D_3, \neg A \sqcup \neg G \sqcup \exists r.C, \neg D_1 \sqcup \neg D_2\}$. The final forgetting view then becomes $\mathcal{V}_3 = \{\neg A \sqcup \forall r.D_1, \neg A \sqcup \forall s.D_2, \neg G \sqcup \exists r.\top, \neg A \sqcup \neg G \sqcup \exists r.C, \neg D_1 \sqcup \neg D_2\}$ with D_3 purified away.*

Observe that in $\mathcal{V}_1$ and $\mathcal{V}_2$ the clause $\neg A \sqcup \neg G \sqcup \exists r.C$ is redundant. We can eliminate this redundancy by applying the *Role Propagation* rule only when a premise of the rule occurs in Δ.

Since the final forgetting views may use definers, the following question may be asked: *Are definers in forgetting views limiting?* We argue that since definers are existentially quantified and the forgetting view is expressed using $\mathcal{ALC}$ syntax, standard reasoning tasks such as *satisfiability checking*, and *query answering*, can be performed with respect to the non-forgotten vocabulary using the existing $\mathcal{ALC}$ methods. The following example explains the idea.

Example 7. *Let ontology $\mathcal{O} = \{A_1 \sqsubseteq \forall r.B, A_2 \sqsubseteq \forall r.\neg B\}$ be an ontology, and $\mathcal{O}^{int} = \{\neg A_1 \sqcup \forall r.D_1, \neg A_2 \sqcup \forall r.D_2, \neg D_1 \sqcup \neg D_2 \sqsubseteq \bot\}$ the intermediate ontology of $\mathcal{O}$ with respect to $\mathcal{F} = \{B\}$ where D_1 and D_2 are definers. Assume $\Delta = \emptyset$, then $\mathcal{O}^{int}$ is the final forgetting view. Both $\mathcal{O}$ and $\mathcal{O}^{int}$ model the information that the r-successors of the elements in the interpretation of A_1 are disjoint from the r-successors of the elements in the interpretation of A_2. Suppose we additionally have a database $\mathcal{A} = \{A_1(a_1), A_2(a_2), r(a_1, b)\}$, and we want to prove the unsatisfiability of $r(a_2, b)$ with respect to the knowledge base consisting of $\mathcal{O}$ and $\mathcal{A}$. This can be done by using a standard $\mathcal{ALC}$ reasoner to show that $\mathcal{O}, \mathcal{A}, r(a_2, b) \models \bot$. Replacing $\mathcal{O}$ with $\mathcal{O}^{int}$, would still prove the unsatisfiability of $r(a_2, b)$. Moreover, the reasoner does not require the full interpretation of D_1 and D_2 to prove that $\mathcal{O}^{int}, \mathcal{A}, r(a_2, b) \models \bot$.*

8. Evaluation

We implemented a prototype of our method based on *Java 12* and the *OWL API 5.1.11*. We refer to our prototype as *SeD*. We used a random corpus of 50 ontologies from the NCBO Bioportal repository to perform the evaluation. Details of the corpus are given in the long version.

We performed two evaluations, each corresponding to a different selection of the forgetting signature $\mathcal{F}$. *Evaluation 1* selected $\mathcal{F}$ as a segment of the N_c sorted by name (recall that N_c is the set of concept names of the input ontology), so $\mathcal{F}$ contained related concept names. E.g., 'Abdomen' and 'Abdomen-pain' were likely to be together in $\mathcal{F}$ as they would be adjacent in the sorted N_c. The intuition was to simulate the use case of extracting the knowledge of a single topic, e.g., the *digestive system* from a large biomedical ontology.

Evaluation 2 selected the concept names that occurred most frequently under role restrictions, aiming for more definers being introduced in $\mathcal{O}^{int}$ and bigger Δ sets. The intuition was to simulate the worst case when the information difference between the intermediate ontology and the deductive views would be large, and the performance of our two-stage forgetting method would be expected to degrade.

In every evaluation and for each ontology in the corpus, three forgetting experiments were performed to forget 10%, 30%, and 50% of concept symbols in N_c giving a total of 150 experiments in each evaluation. Each experiment compared SeD and *Lethe* (version 2.11-0.026)[1] [15]. Lethe is an implementation of the deductive forgetting method in [13]. Each tool was allocated 2GB of memory and five hours time out. All experiments were run on a x64-based processor Intel(R) Core(TM) i5 CPU @ 2.7GHz with a 64-bit operating system (macOS Catalina 10.15.7).

[1]http://www.cs.man.ac.uk/~koopmanp/lethe/index.html

Table 1
Timeouts of SeD and Lethe

	Evaluation 1			Evaluation 2		
	10%	30%	50%	10%	30%	50%
SeD	2	3	5	2	3	3
Lethe	2	7	8	1	6	7

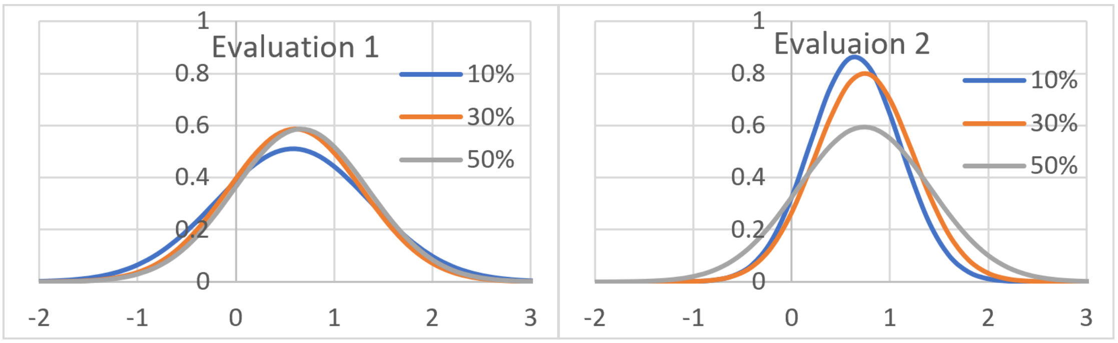

Figure 4: Normal Distribution of the *Gain* values. Range of X-axes is Average $\pm$ 3 Standard Deviations

Table 1 shows the timeouts of SeD and Lethe. SeD appeared to be more reliable than Lethe in both evaluations. Also, SeD was less affected by increasing the size of $\mathcal{F}$ from 10% to 30% to 50% of N_c, suggesting that SeD is more scalable to harder problems than Lethe.

Next we compared the execution times of SeD and Lethe to compute the deductive view. We first computed the time gained by using SeD over Lethe with the formula *Gain* = $(T_L - T_S)/T_L$, where T_L and T_S are the times consumed by Lethe and SeD respectively. Second, we computed the averages and standard deviations of the *Gain* values. In line with standard data analysis methods, *outliers* were excluded. These were experiments with extreme *Gain* values compared to the rest of the experiments. In *Evaluation 1* we excluded two experiments in the 10% setting and one in the 50% setting, whereas in *Evaluation 2* we excluded one experiment in the 10% setting and one from the 50% setting.

The averages and standard deviations in *Evaluation 1* were: (0.58, 0.78), (0.60, 0.68), and (0.66, 0.68) in the 10%, 30%, and 50% settings respectively. In *Evaluation 2* they were: (0.65, 0.46), (0.74, 0.50), and (0.74, 0.67) in the 10%, 30%, and 50% settings respectively. Figure 4 shows the normal distributions of the *Gain* values in the three settings in the two evaluations. The graphs reflect the attained positive averages stated above, and compare the gain values across the three settings in each evaluation, also allowing the two evaluations to be compared. Surprisingly, a better and more consistent performance was found in *Evaluation 2* over *Evaluation 1*, against our expectation that *Evaluation 2* would represent the worst case scenario for SeD. This is indicated by the higher peaks indicating higher probability of achieving the average *Gain*, and the narrower curves indicating less variation in the results.

The performance improvement happens due to the forgetting method itself not the corpus. While Lethe translates the input ontology to a clausal form that is similar to ours, it disallows clauses with two or more negative definers. To compensate for this restriction, Lethe introduces definer symbols as part of the forgetting calculus, and builds a subsumption hierarchy between the definers. This hierarchy forces extra resolution inferences to be performed. The following example illustrates dynamic introduction of definers in Lethe.

Example 8. *Let $\mathcal{O} = \{A_1 \sqsubseteq \exists r.\neg B, A_2 \sqsubseteq \forall r.B\}$, and $\mathcal{F} = \{B\}$. Lethe generates $\mathcal{O}^{clausal}$ as $\{\neg A_1 \sqcup \exists r.D_1, \neg A_2 \sqcup \forall r.D_2, \neg D_1 \sqcup \neg B, \neg D_2 \sqcup B\}$. Instead of resolving B directly to compute the clause $\neg D_1 \sqcup \neg D_2$, Lethe introduces a new definer D_3 and generates an intermediate ontology $\mathcal{O}_1 = \mathcal{O}^{clausal} \cup \{\neg A_1 \sqcup \neg A_2 \sqcup \exists r.D_3, \neg D_3 \sqcup D_1, \neg D_3 \sqcup D_2\}$. The last two clauses on $\mathcal{O}_1$ are resolved with $\neg D_1 \sqcup \neg B$ and $\neg D_2 \sqcup B$ to give $\neg D_3 \sqcup \neg B$ and $\neg D_3 \sqcup B$ which in turn are resolved together to give $\neg D_3$. All clauses where B occurs are then removed to give $\mathcal{O}_2 = \{\neg A_1 \sqcup \exists r.D_1, \neg A_2 \sqcup \forall r.D_2, \neg A_1 \sqcup \neg A_2 \sqcup \exists r.D_3, \neg D_3\}$. The definers D_1, D_2, and D_3 are eliminated in a similar way to the method described in Section 6 giving $\mathcal{V} = \{\neg A_1 \sqcup \exists r.\top, \neg A_1 \sqcup \neg A_2\}$.*

Our normal form is more flexible as it allows several negative definers to appear in a clause, thus avoids the extra resolution inferences performed by Lethe.

We measured the size of the extracted Δ set. In *Evaluation 1*, Δ was on average 0.01%, 0.66%, and 18.3% of the size of the original ontology, in the 10%, 30%, and 50% settings respectively. In *Evaluation 2*, Δ was on average 0.23%, 0.88%, and 7.13% of the size of the original ontology, in the 10%, 30%, and 50% settings respectively.

We also measured the number of definers in Δ as a ratio to the forgetting signature. In *Evaluation 1*, they were on average 0.03%, 0.63%, and 1.5% in the 10%, 30%, and 50% settings respectively. In *Evaluation 2*, they were on average 0.04%, 0%, and 1.4% in the 10%, 30%, and 50% settings respectively. These ratios indicate that our fine-grained method was feasible since appending $\mathcal{O}^{red}$ with axioms from Δ introduced few definers relative to the size of $\mathcal{F}$.

Finally, we measured the size of the deductive view relative to the original ontology. In *Evaluation 1* it was on average 114%, 103%, and 117% of the size of the original ontology in the 10%, 30%, and 50% settings respectively. In *Evaluation 2*, it was on average 113%, 95%, and 103% of the size of the original ontology in the 10%, 30%, and 50% settings respectively.

9. Conclusions and Future Work

We presented a new forgetting method that performs deductive forgetting, and extracts a set Δ of axioms representing the information difference between the original ontology and the deductive view. Not only does this give a clearer understanding, in terms of the modelled information, on the difference between the input ontology and the deductive view, but also it allows a fine-grained forgetting system that gives control over the information modelled in the forgetting view. Empirical evaluation suggests that our forgetting method is faster than the state-of-the-art forgetting tool Lethe despite computing more information. Nevertheless, our evaluation suggested that appending the deductive forgetting view with information from Δ introduces few foreign symbols compared to the forgotten symbols. The final forgetting view therefore remains a compact extract of the original ontology for the use in applications.

Future work will study in greater depth the newly revealed spectrum of forgetting variants, and their intersections with other forgetting variants in the literature.

References

[1] M. Ludwig, B. Konev, Practical uniform interpolation and forgetting for $\mathcal{ALC}$ TBoxes with applications to logical difference, in: Proc. KR 2014, AAAI Press, 2014.

[2] B. C. Grau, B. Motik, Pushing the limits of reasoning over ontologies with hidden content, in: Proc. KR 2010, AAAI Press, 2010, pp. 214–224. URL: http://www.comlab.ox.ac.uk/people/Boris.Motik/pubs/gm10pushing-ibq.pdf.

[3] W. Del-Pinto, R. A. Schmidt, ABox abduction via forgetting in ALC, in: Proc. AAAI 2019, AAAI Press, 2019, pp. 2768–2775.

[4] J. Lang, P. Marquis, Reasoning under inconsistency: A forgetting-based approach, Artificial Intelligence 174 (2010) 799–823. URL: https://www.sciencedirect.com/science/article/pii/S0004370210000676. doi:https://doi.org/10.1016/j.artint.2010.04.023.

[5] D. Subramanian, M. R. Genesereth, The relevance of irrelevance., in: Proc. IJCAI, 1987.

[6] G. Lakemeyer, Relevance from an Epistemic Perspective, Artif. Intell. 97 (1997) 137–167. doi:10.1016/S0004-3702(97)00038-6.

[7] J. Lang, P. Liberatore, P. Marquis, Propositional independence: Formula-variable independence and forgetting, Artificial Intelligence Research 18 (2003) 391–443.

[8] E. Erdem, P. Ferraris, Forgetting Actions in Domain Descriptions, in: Proc. AAAI 2007, AAAI Press, 2007, pp. 409–414.

[9] Y. Zhang, Y. Zhou, Knowledge forgetting: Properties and applications, Artificial Intelligence 173 (2009) 1525–1537. doi:10.1016/j.artint.2009.07.005.

[10] C. Lutz, F. Wolter, Foundations for uniform interpolation and forgetting in expressive description logics, in: Proc. IJCAI 2011, 2011, p. 989–995.

[11] J. Delgrande, A Knowledge Level Account of Forgetting, Artificial Intelligence Research 60 (2017) 1165–1213. doi:10.1613/jair.5530.

[12] A. Nonnengart, A. Szalas, A fixpoint approach to second-order quantifier elimination with applications to correspondence theory, in: Logic at Work, Springer, 1998.

[13] P. Koopmann, R. A. Schmidt, Uniform interpolation of $\mathcal{ALC}$-ontologies using fixpoints, in: Proc. FroCoS 2013, volume 8152 of *LNAI*, Springer, 2013, pp. 87–102.

[14] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider, The Description Logic Handbook, Cambridge Univ. Press, 2003.

[15] P. Koopmann, R. A. Schmidt, Implementation and evaluation of forgetting in ALC-ontologies, in: Proc. WoMo 2013, volume 1081, CEUR-WS.org, 2013.

[16] F. Lin, R. Reiter, Forget it, in: AAAI Fall Symp. on Relevance, 1994, pp. 154–159.

[17] D. M. Gabbay, H. J. Ohlbach, Quantifier elimination in second-order predicate logic, in: Proc. KR 1992, Morgan Kaufmann, 1992, pp. 425–435.

[18] Y. Zhang, Y. Zhou, Forgetting revisited, in: Proc. KR, AAAI Press, 2010.

[19] S. Ghilardi, An algebraic theory of normal forms, Ann. Pure Appl. Logic 71 (1995) 189 – 245.

[20] A. Herzig, J. Mengin, Uniform interpolation by resolution in modal logic, in: Proc. JELIA 2008, volume 5293 LNAI, Springer, 2008, pp. 219–231.

[21] H. Ditmarsch, A. Herzig, J. Lang, P. Marquis, Introspective forgetting, in: Proc. AI 2008, Springer, 2008, pp. 18–29.

[22] B. Konev, D. Walther, F. Wolter, Forgetting and uniform interpolation in large-scale

description logic terminologies, in: Proc. IJCAI 2009, Morgan Kaufmann, 2009, p. 830–835.

[23] B. Konev, C. Lutz, D. Walther, F. Wolter, Model-theoretic inseparability and modularity of description logic ontologies, Artificial Intelligence 203 (2013) 66–103. doi:10.1016/j.artint.2013.07.004.

[24] E. Botoeva, B. Konev, C. Lutz, V. Ryzhikov, F. Wolter, M. Zakharyaschev, Inseparability and conservative extensions of description logic ontologies: A survey, in: Reasoning Web 2016, Springer, 2017, pp. 27–89. URL: https://doi.org/10.1007/978-3-319-49493-7_2. doi:10.1007/978-3-319-49493-7_2.

[25] C. Lutz, F. Wolter, Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$, Symbolic Computation 45 (2010) 194–228. doi:10.1016/j.jsc.2008.10.007.

[26] G. D'Agostino, M. Hollenberg, Uniform interpolation, automata and the modal μ-calculus, Logic Group Preprint Series 165 (1996).

[27] D. Calvanese, G. De Giacomo, M. Lenzerini, Reasoning in Expressive Description Logics with Fixpoints Based on Automata on Infinite Trees, in: Proc. IJCAI 1999, Morgan Kaufmann, 1999, pp. 84–89.

[28] A. Nonnengart, C. Weidenbach, Computing small clause normal forms, in: Handbook of Automated Reasoning, North-Holland, 2001, pp. 335 – 367.