Forgetting Concept and Role Symbols in $\mathcal{ALCOIH}\mu^+(\nabla, \sqcap)$-Ontologies

Yizheng Zhao and **Renate A. Schmidt**
The University of Manchester, UK

Abstract

Forgetting is a non-standard reasoning problem concerned with creating restricted views for ontologies relative to subsets of their initial signatures while preserving all logical consequences up to the symbols in the restricted views. In this paper, we present an Ackermann-based approach for forgetting of concept and role symbols in ontologies expressible in the description logic $\mathcal{ALCOIH}\mu^+(\nabla, \sqcap)$. The method is one of only few approaches that can eliminate role symbols, that can handle role inverse, ABox statements, and is the only approach so far providing support for forgetting in description logics with nominals. Despite the inherent difficulty of forgetting for this level of expressivity, performance results with a prototypical implementation have shown very good success rates on real-world ontologies.

1 Introduction

This paper presents a practical forgetting method for creating restricted views of ontologies expressed in the language of the description logic $\mathcal{ALCOIH}\mu^+(\nabla, \sqcap)$. The work is motivated by the high demand for advanced techniques for ontology-based knowledge processing. Research of forgetting, often referred to as uniform interpolation (also known as variable elimination, projection or second-order quantifier elimination) has gained significant momentum since the work of various groups developing forgetting methods for the description logic $\mathcal{ALC}$ and logics weaker than $\mathcal{ALC}$, e.g., [Konev *et al.*, 2009b; Lutz *et al.*, 2012; Ludwig and Konev, 2014; Nikitina and Rudolph, 2014; Wang *et al.*, 2009; 2014], and the foundational studies of the properties of forgetting for description logics by, e.g., [Wang *et al.*, 2008; Konev *et al.*, 2009a; Lutz and Wolter, 2011; Lutz *et al.*, 2012; Konev *et al.*, 2013]. These works give arguments for the important role of forgetting in realizing various tasks crucial for effective processing and management of ontologies. For example, forgetting can be used for ontology analysis, for creating ontology summaries, for ontology reuse, for information hiding, for computing the logical difference between ontologies, for ontology debugging and repair, and for query answering.

Early work in the area primarily focused on forgetting concept symbols, as role forgetting was realized to be significantly harder than forgetting of concept symbols [Wang *et al.*, 2008], because the result of forgetting role symbols often requires more expressivity than is available in the target logic. Although the earliest work did study concept and role forgetting, e.g. [Wang *et al.*, 2009], most subsequent work considered only concept forgetting with the exception of [Koopmann and Schmidt, 2013a; 2013b; 2014; 2015]. Their method can perform both concept and role forgetting for description logics extending $\mathcal{ALC}$ up to and including description logics with the expressiveness of $\mathcal{SH}$, $\mathcal{SIF}$ and $\mathcal{SHQ}$. In addition they have extended their method to forgetting for description logics with ABoxes for logics between $\mathcal{ALC}$ and $\mathcal{SHI}$.

The work on forgetting for description logics is predated by work on second-order quantifier elimination [Gabbay and Ohlbach, 1992; Szałas, 1993; Nonnengart and Szałas, 1999; Doherty *et al.*, 1997; Gabbay *et al.*, 2008], which can be traced to questions posed by Boole and seminal work of [Ackermann, 1935]. These works triggered and influenced work in knowledge representation [Lin and Reiter, 1994; Wernhard, 2011], but also led to striking results in the automation of correspondence theory of modal logics [Gabbay and Ohlbach, 1992; Conradie *et al.*, 2006; Schmidt, 2012]. Because of the close relationship between description logics and modal logics, besides the work on modal correspondence theory, investigations of uniform interpolation for modal logics and the μ-calculus [Visser, 1996; Herzig and Mengin, 2008; D'Agostino and Hollenberg, 2000] are relevant. These are related to concept forgetting, but not to role forgetting.

The contribution of this paper is an approach for forgetting of concept and role symbols in expressive description logics not considered so far. The method accommodates ontologies expressible in the description logic $\mathcal{ALCOIH}$ and the extension allowing positive occurrences of the least fixpoint operator μ, the top role ∇ and role conjunction $\sqcap$. This means the method is one of only few approaches that can eliminate role symbols, while also handling role inverse, ABox statements, and the only approach so far providing support for forgetting in description logics with nominals. The added expressivity has the advantage that it reduces information loss; for instance, the solution of forgetting the role symbol r in the ontology $\{A \sqsubseteq \exists r.B, \exists r.B \sqsubseteq B\}$ is $\{A \sqsubseteq \exists\nabla.B, A \sqsubseteq B\}$, whereas in a description logic without the top role (or ABox axioms or nominals) the uniform interpolant is $\{A \sqsubseteq B\}$, which is weaker.

Being based on the Ackermann approach to second-order quantifier elimination [Doherty *et al.*, 1997; Szałas, 2006; Schmidt, 2012; Zhao and Schmidt, 2015] the method terminates always. We have shown that the method is correct, i.e., the forgetting solution computed is equivalent to the original ontology up to the symbols that have been forgotten. A general problem of forgetting is marking out a language expressive enough to allow for solutions to be given by finitely many formulas, while being not too expressive so that the forgetting solutions returned can be further processed by available tools. For example, including second-order quantifiers of concept and role symbols in the language solves the forgetting problem trivially, but tool support would be absent. Even though concept forgetting becomes possible with fixpoint operators, at present mainstream tools do not support fixpoints. We avoid this problem by restricting occurrences of fixpoints in our logic to cases that can be simulated without them. Despite our method not being complete, performance results of an evaluation with a prototypical implementation have shown very good success rates on real-world ontologies.

2 The Description Logic $\mathcal{ALCOIH}\mu^+(\nabla,\sqcap)$

Let $\mathsf{N_C}$, $\mathsf{N_R}$ and $\mathsf{N_O}$ be mutually disjoint sets of *concept symbols (names)*, *role symbols (names)* and *individuals*, respectively, and let N_μ be a set of *concept variables* disjoint with $\mathsf{N_C}$, $\mathsf{N_R}$ and $\mathsf{N_O}$. *Roles* in $\mathcal{ALCOIH}(\nabla,\sqcap)$ can be the top role ∇, a role symbol $r \in \mathsf{N_R}$, the inverse r^- of a role symbol r (an *inverted role symbol*), or can be formed with conjunction. *Concepts* in $\mathcal{ALCOIH}(\nabla,\sqcap)$ have one of the following forms: $a \mid \top \mid A \mid \neg C \mid C \sqcup D \mid \forall R.C$, where $a \in \mathsf{N_O}$, $A \in \mathsf{N_C}$, C and D are any concepts, and R is any role in $\mathcal{ALCOIH}(\nabla,\sqcap)$. We assume w.l.o.g. that concept and role expressions are equivalent relative to associativity and commutativity of $\sqcap$ and $\sqcup$, $\neg$ and $^-$ are involutions, and $\top$ and ∇ are units w.r.t. $\sqcap$.

We assume that a TBox $\mathcal{T}$ is a set of *concept axioms* of the form $C \sqsubseteq D$, where C and D are closed concepts, i.e., contain no concept variable not in the scope of a μ operator. An RBox $\mathcal{R}$ is a set of *role axioms* of the form $R \sqsubseteq S$, where R and S are role symbols or inverted role symbols. An ABox is a set of *concept assertions* of the form $C(a)$ and *role assertions* of the form $R(a,b)$. In description logics with nominals ABox assertions are superfluous, since they can be equivalently expressed as TBox axioms, namely, $C(a)$ as $a \sqsubseteq C$ and $R(a,b)$ as $a \sqsubseteq \exists R.b$. Hence, we assume an ontology is the union of a TBox and an RBox.

The language of $\mathcal{ALCOIH}\mu^+(\nabla,\sqcap)$ extends that of $\mathcal{ALCOIH}(\nabla,\sqcap)$ with atomic least fixpoint expressions of the form $\mu X.C[X]$, where $X \in \mathsf{N}_\mu$ and $C[X]$ is a concept expression in which X occurs only positively (under an even number of explicit and implicit negations) in place of a regular concept symbol. Moreover, *μ-expressions may occur only positively*. Because of this restriction, μ-expressions can be simulated in $\mathcal{ALCOIH}(\nabla,\sqcap)$ with auxiliary concept symbols as in [Koopmann and Schmidt, 2013b].

The semantics of $\mathcal{ALCOIH}\mu^+(\nabla,\sqcap)$ is defined in terms of an *interpretation* $\mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I}\rangle$, where $\Delta^\mathcal{I}$ is any nonempty set, and $\cdot^\mathcal{I}$ assigns to every nominal $a \in \mathsf{N_O}$ a singleton set $a^\mathcal{I} \subseteq \Delta^\mathcal{I}$, to every concept symbol $A \in \mathsf{N_C}$ a subset $A^\mathcal{I}$ of $\Delta^\mathcal{I}$, and to every role symbol $r \in \mathsf{N_R}$ a subset $r^\mathcal{I}$ of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$. The interpretation function $\cdot^\mathcal{I}$ is inductively extended to concept and role expressions as follows:

$$\top^\mathcal{I} = \Delta^\mathcal{I} \qquad \nabla^\mathcal{I} = \Delta^\mathcal{I} \times \Delta^\mathcal{I} \qquad (\neg C)^\mathcal{I} = \Delta^\mathcal{I} \backslash C^\mathcal{I}$$
$$(C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I} \qquad (R \sqcap S)^\mathcal{I} = R^\mathcal{I} \cap S^\mathcal{I}$$
$$(\forall R.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \forall y.(x,y) \in R^\mathcal{I} \to y \in C^\mathcal{I}\}$$
$$(R^-)^\mathcal{I} = \{(y,x) \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid (x,y) \in R^\mathcal{I}\}$$

The semantics of least fixpoint expressions takes additionally an assignment function $\cdot^\mathcal{V}$, mapping concept variables to subsets of $\Delta^\mathcal{I}$, which is defined by: $(\mu X.C)^{\mathcal{I},\mathcal{V}} = \bigcap\{\mathcal{E} \subseteq \Delta^\mathcal{I} \mid C^{\mathcal{I},\mathcal{V}[\mathcal{X}/\mathcal{E}]} \subseteq \mathcal{E}\}$, where $\mathcal{V}[X/\mathcal{E}]$ denotes an assignment function identical to $\cdot^\mathcal{V}$ except that $X^\mathcal{V} = \mathcal{E}$. $C^{\mathcal{I},\mathcal{V}}$ is the interpretation $C^\mathcal{I}$ of C that adopts the assignment function $\cdot^\mathcal{V}$, and $C^{\mathcal{I},\mathcal{V}} = C^\mathcal{I}$ when $\cdot^\mathcal{V}$ is defined for all variables in C.

A concept axiom $C \sqsubseteq D$ is *true* in an interpretation $\mathcal{I}$ iff $C^\mathcal{I} \subseteq D^\mathcal{I}$, and a role axiom $R \sqsubseteq S$ is *true* in $\mathcal{I}$ iff $R^\mathcal{I} \subseteq S^\mathcal{I}$. $\mathcal{I}$ is a *model* of an ontology $\mathcal{O}$ iff every axiom in $\mathcal{O}$ is *true*. In this case we write $\mathcal{I} \models \mathcal{O}$.

Theorem 1 *Reasoning in $\mathcal{ALCOIH}\mu^+(\nabla,\sqcap)$ is decidable.*

This follows, e.g., from the decidability of guarded fixpoint logic [Grädel and Walukiewicz, 1999] or the description logic $\mathcal{ALBO}^{\mathsf{id}}$ [Schmidt and Tishkovsky, 2014] (because positive occurrences of μ-expressions can be encoded). Reasoning in $\mathcal{ALCOIH}\mu^+(\nabla,\sqcap)$ can be performed with MetTeL, which decides $\mathcal{ALBO}^{\mathsf{id}}$ [Tishkovsky *et al.*, 2012].

The normal form of our method are axioms in clausal normal form. A *TBox clause* is a disjunction of a finite number of $\mathcal{ALCOIH}\mu^+(\nabla,\sqcap)$-concepts. A *role literal* is a role symbol, an inverted role symbol, or formed with negation. An *RBox clause* is a disjunction of exactly two role literals of complementary polarity. TBox and RBox clauses are obtained by clausification of the TBox and RBox axioms, where in the case of role axioms role negation is introduced. Nominals are treated like regular concept symbols in our method, except that they cannot be specified as symbols to be forgotten.

A clause that contains a designated concept (role) symbol $\mathcal{S}$ is called an $\mathcal{S}$-clause. Given an $\mathcal{S}$-clause C, an occurrence of $\mathcal{S}$ is *positive* in C if it is under an even number of explicit and implicit negations, otherwise it is *negative*. For instance, an occurrence of r is assumed to be negative in $\forall r.A$ and $r \sqsubseteq s$, and positive in $\neg\forall r.A$ and $s \sqsubseteq r$. C is *positive* (*negative*) w.r.t. $\mathcal{S}$ if every occurrence of $\mathcal{S}$ in C is positive (negative). A set $\mathcal{N}$ of clauses is *positive* (*negative*) w.r.t. $\mathcal{S}$ if every occurrence of $\mathcal{S}$ in $\mathcal{N}$ is positive (negative).

We now formalize our notion of forgetting. By $\mathrm{sig}(E)$ we denote the concept and role symbols occurring in E *excluding nominals*, where E ranges over all concepts, axioms, clauses, a set of clauses and ontologies. Let $\mathcal{S}$ be any concept or role symbol and let $\mathcal{I}$ and $\mathcal{I}'$ be interpretations. We say $\mathcal{I}$ and $\mathcal{I}'$ are *equivalent up to $\mathcal{S}$*, or *$\mathcal{S}$-equivalent*, if $\mathcal{I}$ and $\mathcal{I}'$ coincide but differ possibly in the interpretations of $\mathcal{S}$. More generally, $\mathcal{I}$ and $\mathcal{I}'$ are *equivalent up to a set* Σ, or *Σ-equivalent*, if $\mathcal{I}$ and $\mathcal{I}'$ are the same but differ possibly in the interpretations of the symbols in Σ. In this paper, Σ is assumed to be the set of the concept and role symbols to be forgotten.

Definition 1 (Forgetting for $\mathcal{ALCOIH}\mu^+(\triangledown,\sqcap)$) *Let $\mathcal{O}$ and $\mathcal{O}'$ be $\mathcal{ALCOIH}\mu^+(\triangledown,\sqcap)$-ontologies and let Σ be any subset of* $\mathrm{sig}(\mathcal{O})$. *$\mathcal{O}'$ is the* solution of forgetting the Σ-symbols *in $\mathcal{O}$, if the following conditions hold: (i) $\mathrm{sig}(\mathcal{O}') \subseteq \mathrm{sig}(\mathcal{O})$ and $\mathrm{sig}(\mathcal{O}') \cap \Sigma = \emptyset$, and (ii) for any interpretation $\mathcal{I}$: $\mathcal{I} \models \mathcal{O}'$ iff $\mathcal{I}' \models \mathcal{O}$, for some interpretation $\mathcal{I}'$ Σ-equivalent to $\mathcal{I}$.*

This states that the given ontology $\mathcal{O}$ and the forgetting solution $\mathcal{O}'$ are equivalent up to the interpretations of the symbols in Σ. Forgetting solutions are unique up to equivalence.

3 Overview of the Forgetting Method

The forgetting process in our method consists of three main phases: the reduction to a set of $\mathcal{ALCOIH}\mu^+(\triangledown,\sqcap)$-clauses, the forgetting phase and the definer elimination phase (see Figure 1). In the forgetting phase, an analyzer may be used to generate forgetting orderings, $\succ^{\mathcal{C}}$ and $\succ^{\mathcal{R}}$, of the concept symbols and role symbols in Σ.

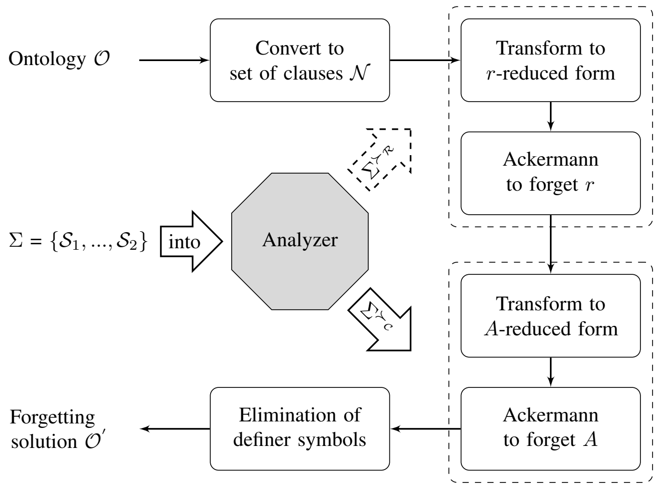
Ontology $\mathcal{O}$
Convert to set of clauses $\mathcal{N}$
Transform to r-reduced form
Ackermann to forget r
$\Sigma^{\succ^{\mathcal{R}}}$
$\Sigma = \{\mathcal{S}_1, ..., \mathcal{S}_2\}$
into
Analyzer
$\Sigma^{\succ^{\mathcal{C}}}$
Transform to A-reduced form
Ackermann to forget A
Elimination of definer symbols
Forgetting solution $\mathcal{O}'$

Figure 1: Overview of the forgetting method

The input to the method is an ontology $\mathcal{O}$ of TBox and RBox axioms expressible in $\mathcal{ALCOIH}\mu^+(\triangledown,\sqcap)$, and a set Σ with the concept and role symbols to be forgotten. The Σ-symbols are entirely determined by the user and their application demands; Σ can thus be any subset of the signature of the initial ontology as the user wishes. The first phase transforms the input ontology $\mathcal{O}$ into a set $\mathcal{N}$ of clauses.

The forgetting phase is an iteration of several rounds in which individual concept and role symbols are eliminated. An important feature of the method is that concept symbols and role symbols are forgotten in a focused way, i.e., the rules for concept forgetting and the rules for role forgetting are mutually independent; concept and role symbols can therefore be forgotten in any order. In the forgetting phase, if $\mathcal{S} \in \Sigma$ is a symbol to be forgotten, the idea is to transform the $\mathcal{S}$-clauses into *$\mathcal{S}$-reduced form*, so that the forgetting rules, specifically the Ackermann rules, can be applied to eliminate $\mathcal{S}$. Thus, whether the Σ-symbols are eliminable depends entirely upon whether the current set of clauses can be transformed into corresponding reduced forms. For A a concept symbol, the conversion to A-reduced form is performed using the rewrite rules in the calculus for concept forgetting, while for r a role symbol, the conversion to r-reduced form is realized using the rewrite rules in the calculus for role forgetting. The calculi are described in the next sections. To provide crucial control and flexibility in how the steps are performed, auxiliary concept symbols, called *definer symbols*, are introduced in the role forgetting rounds. The final phase in the method attempts to eliminate these using concept forgetting.

Previous research has shown that the success rates of forgetting depend very much upon the order in which the Σ-symbols are forgotten [Gabbay and Ohlbach, 1992; Doherty *et al.*, 1997; Conradie *et al.*, 2006; Schmidt, 2012; Koopmann and Schmidt, 2014], even when forgetting only concept symbols, e.g., [Ludwig and Konev, 2014; Zhao and Schmidt, 2015]. An important challenge therefore is to find appropriate orderings for eliminating Σ-symbols. Our method either follows the user-specified ordering, or it uses a heuristic analysis based on frequency counts of the Σ-symbols, where concept symbols and role symbols are analyzed separately.

We refer to the maximal symbol of Σ w.r.t. the forgetting ordering $\succ$ as the *pivot* in our method. Following $\succ$ (and starting with the pivot), the method forgets the Σ-symbols one by one. If the pivot is successfully eliminated from $\mathcal{N}$, we attempt to forget the next symbol in the ordering $\succ$, which has become the new pivot. Otherwise the pivot is flagged as a currently non-forgettable symbol and remains in Σ. The next symbol in $\succ$ then becomes the pivot. When the end of the ordering $\succ$ has been reached, the calculus is applied to the flagged symbols, attempting again to eliminate them. Success will always be pursued until a forgetting solution is found. Though the ordering $\succ$ provides useful guidance during the forgetting process, it does not generally guarantee success of forgetting. On the symbols not eliminated, a different ordering not attempted before, will be used. When all possible orderings have been attempted and there are still Σ-symbols remaining, our method has been unable to find a forgetting solution (because it is unable to find a suitable reduced-form for the non-forgettable symbols).

Inexpensive equivalence-preserving simplification rules are applied throughout the forgetting process, ensuring that the current clauses are always simple representations for efficiency. Our experimental results suggest that the simplification rules are also crucial to the success of concept forgetting, as they can help conversion of clauses to reduced form, when the pivot is a concept symbol. What the method returns, if forgetting is successful, is an ontology $\mathcal{O}'$ that does not contain the symbols in Σ.

Theorem 2 *For any $\mathcal{ALCOIH}\mu^+(\triangledown,\sqcap)$-ontology $\mathcal{O}$ and any set $\Sigma \subseteq \mathrm{sig}(\mathcal{O})$ of symbols to be forgotten, the method always terminates and returns a set $\mathcal{O}'$ of clauses. If $\mathcal{O}'$ does not contain any Σ-symbols, the method was* successful. *$\mathcal{O}'$ is then a solution of forgetting the symbols in Σ from $\mathcal{O}$.*

If neither $\mathcal{O}$ nor $\mathcal{O}'$ uses fixpoints, $\mathcal{O}'$ is Σ-equivalent to $\mathcal{O}$ in $\mathcal{ALCOIH}(\nabla,\sqcap)$. Otherwise, it is Σ-equivalent to $\mathcal{O}$ in $\mathcal{ALCOIH}\mu^+(\nabla,\sqcap)$.

4 Calculus for Concept Forgetting

This section presents the calculus for forgetting *concept symbols* from ontologies expressible in $\mathcal{ALCOIH}\mu^+(\nabla,\sqcap)$. The calculus extends the calculus of [Zhao and Schmidt, 2015] for concept forgetting in $\mathcal{ALCOI}$-ontologies by accommodating the top role in the calculus and allowing the least fixpoint operator to occur in the forgetting solutions for problems that contain cyclic dependencies. Since concept symbols occur only in TBox axioms, only the TBox needs to be processed for concept forgetting. The RBox remains unchanged.

Non-Cyclic Ackermann$^{\mathcal{C}}$

$$\frac{\mathcal{N}, C_1 \sqcup A, \ldots, C_n \sqcup A}{\mathcal{N}^A_{\neg C_1 \sqcup \ldots \sqcup \neg C_n}}$$

provided: (i) A does not occur in the C_i, and (ii) $\mathcal{N}$ is negative w.r.t. A.

Cyclic Ackermann$^{\mathcal{C}}$

$$\frac{\mathcal{N}, C_1[A] \sqcup A, \ldots, C_n[A] \sqcup A}{\mathcal{N}^A_{\mu X.(\neg C_1 \sqcup \ldots \sqcup \neg C_n)[X]}}$$

provided: (i) the C_i are negative w.r.t. A, and (ii) $\mathcal{N}$ is negative w.r.t. A.

Purify$^{\mathcal{C}}$

$$\frac{\mathcal{N}}{\mathcal{N}^A_{(\neg)\top}}$$ provided: $\mathcal{N}$ is positive (negative) w.r.t. A.

Figure 2: Rules for forgetting pivot $A \in \mathsf{N_C}$

Definition 2 (A-Reduced Form) *Suppose A is a concept symbol. A clause is in* A-reduced form *if it is negative w.r.t. A, or it has the form $A \sqcup C$, where C is an $\mathcal{ALCOIH}\mu^+(\nabla,\sqcap)$-concept that (i) does not have any occurrences of A, or (ii) is negative w.r.t. A. A set $\mathcal{N}$ of clauses is in* A-reduced form *if every A-clause in $\mathcal{N}$ is in A-reduced form.*

The Ackermann$^{\mathcal{C}}$ rules and the Purify$^{\mathcal{C}}$ rule, given in Figure 2, are the *forgetting rules* that lead to the elimination of concept symbols in a set of clauses. For $A \in \mathsf{N_C}$ the pivot and C a concept expression, $\mathcal{N}^A_C$ denotes the set obtained from $\mathcal{N}$ by replacing every occurrence of A by C. We refer to the clauses of the form $C_i \sqcup A$ and $C_i[A] \sqcup A$ ($1 \leq i \leq n$) as *positive premises* of the Non-Cyclic Ackermann$^{\mathcal{C}}$ rule and the Cyclic Ackermann$^{\mathcal{C}}$ rule, respectively. The Ackermann$^{\mathcal{C}}$ rules are applicable to forget A in $\mathcal{N}$ only if $\mathcal{N}$ is in A-reduced form.

Theorem 3 (Ackermann Lemma, Concept Forgetting) *Let $\mathcal{I}$ be any $\mathcal{ALCOIH}\mu^+(\nabla,\sqcap)$-interpretation. For A the pivot, when the Non-Cyclic Ackermann$^{\mathcal{C}}$ rule is applicable, the conclusion of the rule is true in $\mathcal{I}$ iff for some interpretation $\mathcal{I}'$ A-equivalent to $\mathcal{I}$, the premises are true in $\mathcal{I}'$. The same is true for the Cyclic Ackermann$^{\mathcal{C}}$ and the Purify$^{\mathcal{C}}$ rules.*

Concept Clausify $$\frac{\mathcal{N}, C \sqcup \neg(D_1 \sqcup \ldots \sqcup D_n)}{\mathcal{N}, C \sqcup \neg D_1, \ldots, C \sqcup \neg D_n}$$

provided: A occurs positively in $\neg(D_1 \sqcup \ldots \sqcup D_n)$.

Concept Surfacing $$\frac{\mathcal{N}, C \sqcup \forall R.D}{\mathcal{N}, (\forall R^-.C) \sqcup D}$$

provided: (i) A does not occur positively in C, and (ii) A occurs positively in $\forall R.D$.

Skolemization^R $$\frac{\mathcal{N}, \neg a \sqcup \neg\forall R.C}{\mathcal{N}, \neg a \sqcup \neg\forall R.\neg b, \neg b \sqcup \neg C}$$

provided: (i) A occurs positively in $\neg\forall R.C$, and (ii) b is a fresh nominal.

Skolemization$^\nabla$ $$\frac{\mathcal{N}, C \sqcup \neg\forall\nabla.D}{\mathcal{N}, \neg b \sqcup \neg D \sqcup \forall\nabla.C}$$

provided: (i) A occurs positively in $\neg\forall\nabla.D$, and (ii) b is a fresh nominal.

Case Splitting $$\frac{\mathcal{N}, \neg a \sqcup C_1 \sqcup \ldots \sqcup C_n}{\mathcal{N}, \neg a \sqcup C_1 \mid \ldots \mid \neg a \sqcup C_n}$$

provided: A occurs positively in $C_1 \sqcup \ldots \sqcup C_n$.

Sign Switching $$\frac{\mathcal{N}}{\mathcal{N}^A_{\neg A}}$$

provided: (i) $\mathcal{N}$ is closed w.r.t. the other rewrite rules, and (ii) Sign Switching has not been performed on A.

Figure 3: The rewrite rules for finding A-reduced form

This states that eliminating the pivot symbol with the Ackermann$^{\mathcal{C}}$ and Purify$^{\mathcal{C}}$ rules preserves equivalence up to the pivot. Given a pivot $A \in \mathsf{N_C}$ and a set $\mathcal{N}$ of clauses in A-reduced form, the Ackermann$^{\mathcal{C}}$ and Purify$^{\mathcal{C}}$ rules are applied in different situations. Specifically, the Non-Cyclic Ackermann$^{\mathcal{C}}$ rule is applied when A does not occur in the concepts C_i of the positive premises ($1 \leq i \leq n$). The Cyclic Ackermann$^{\mathcal{C}}$ rule is applied when A occurs in the C_i but only negatively (e.g., A in the cyclic clause $A \sqcup \neg\forall r.A$). Fixpoints are introduced in this case in order to facilitate finite representation of the forgetting solution, where every occurrence of A in $\neg C_1 \sqcup \ldots \sqcup \neg C_n$ is replaced by X, a fresh concept variable, and every occurrence of A in $\mathcal{N}$ is replaced by $\mu X.(\neg C_1 \sqcup \ldots \sqcup \neg C_n)[X]$. The Purify$^{\mathcal{C}}$ rule can be applied any time provided that A is *pure* in $\mathcal{N}$, i.e., $\mathcal{N}$ is positive (or negative) w.r.t. A.

Figure 3 lists a set of rewrite rules for finding the pivot-reduced form of a clause for the pivot a concept symbol. New compared to [Zhao and Schmidt, 2015], besides the Cyclic Ackermann$^{\mathcal{C}}$ rule, is the Skolemization$^\nabla$ rule that rewrites existential restrictions over the top role by the introduction of fresh nominals. The Case Splitting rule splits the derivation into several branches. The intermediate result returned at the end of a symbol elimination round is the disjunction of the solutions of each branch in the derivation. Crucial to the practicality of our method are a number of equivalence-preserving simplification rules, not described here.

It follows from [Schmidt, 2012; Zhao and Schmidt, 2015], which this work extends, that our method for concept forgetting succeeds on problems corresponding to the modal classes of $\mathcal{C}$ and $\mathcal{C}^{\succ}$ [Schmidt, 2012], which subsume the class of Sahlqvist formulas [Sahlqvist, 1975] and the class of monadic inductive formulas [Goranko and Vakarelov, 2006].

5 Calculus for Role Forgetting

The main contribution of this paper is a calculus for goal-oriented forgetting of *role symbols* in ontologies expressible in $\mathcal{ALCOIH}\mu^+(\nabla,\sqcap)$. Since role symbols also occur in the RBox, both the TBox and the RBox need to be processed when role symbols are to be forgotten.

Role Surfacing to TBox clauses

$$\frac{\mathcal{N}, C \sqcup \forall r^-.D}{\mathcal{N}, D \sqcup \forall r.C}$$

provided: r does not occur in C and D.

Role Surfacing to RBox clauses

$$\frac{\mathcal{N}, \neg S \sqcup r^-}{\mathcal{N}, \neg S^- \sqcup r} \qquad \frac{\mathcal{N}, S \sqcup \neg r^-}{\mathcal{N}, S^- \sqcup \neg r}$$

provided: S is a role symbol or an inverted role symbol.

Figure 4: The rewrite rules for finding r-reduced form

Definition 3 (r-Reduced Form) *Suppose r is a role symbol. A clause is in* r-reduced form *if it has the form $C \sqcup \forall r.D$ or $C \sqcup \neg\forall r \sqcap Q.D$, where C and D are $\mathcal{ALCOIH}\mu^+(\nabla,\sqcap)$-concepts that do not contain any occurrence of r and Q is a role that does not contain any occurrence of r; or it has the form $\neg S \sqcup r$ or $S \sqcup \neg r$, where $S \in \{s, s^-\}$ and s ($\neq r$) is a role symbol. A set $\mathcal{N}$ of clauses is in* r-reduced form *if every r-clause in $\mathcal{N}$ is in r-reduced form.*

As in concept forgetting, the pivot-clauses are first transformed into pivot-reduced form, so that the Ackermann rule for role forgetting becomes applicable (to eliminate the pivot). Finding the pivot-reduced form of a clause is however not always possible unless definer symbols are introduced. *Definer symbols* are specialized concept symbols that do not occur in the present ontology [Koopmann and Schmidt, 2013b], and are introduced as follows: given a clause of the form $C \sqcup \forall r^{(-)}.D$ or $C \sqcup \neg\forall r^{(-)}.D$, with r being the pivot and occurring in $\mathcal{Q} \in \{C, D\}$, the definer symbols are used as substitutes, incrementally replacing C and D until neither contains any occurrences of r. A new clause $\neg\mathcal{D} \sqcup \mathcal{Q}$ is added to the clause set for each replaced subconcept $\mathcal{Q}$, where $\mathcal{D}$ is a fresh definer symbol. For example, introducing a definer symbol D_1 leads to $A \sqcup \neg\forall r.\forall r.B$ being rewritten with $A \sqcup \neg\forall r.D_1$ and $\neg D_1 \sqcup \forall r.B$, where A and B are concept symbols. The definer symbols are eliminated (if they were introduced in the conversion of clauses to the pivot-reduced form) using the rules for concept forgetting once all role symbols in Σ have been forgotten. It is for this reason that our implementation defaults to forgetting role symbols first so that the definer symbols can be eliminated as part of subsequent concept forgetting.

Ackermann$^{\mathcal{R}}$

$$\frac{\mathcal{N}, \boxed{C^1 \sqcup \neg\forall r \sqcap Q^1.D^1, \ldots, C^k \sqcup \neg\forall r \sqcap Q^k.D^k}, \boxed{\neg T^1 \sqcup r, \ldots, \neg T^u \sqcup r}, C_1 \sqcup \forall r.D_1, \ldots, C_m \sqcup \forall r.D_m, \neg r \sqcup S_1, \ldots, \neg r \sqcup S_n}{\mathcal{N}, \boxed{\mathcal{T}\text{-}Block^{\mathcal{H}}(1,m), \ldots, \mathcal{T}\text{-}Block^{\mathcal{H}}(k,m)}, \boxed{\mathcal{R}\text{-}Block^{\mathcal{C}}(1,m), \ldots, \mathcal{R}\text{-}Block^{\mathcal{C}}(u,m)}, \boxed{\mathcal{R}\text{-}Block^{\mathcal{R}}(1,n), \ldots, \mathcal{R}\text{-}Block^{\mathcal{R}}(u,n)}}$$

provided: (i) r does not occur in $\mathcal{N}$, and (ii) $\mathcal{N}$ is in r-reduced form.

Purify$^{\mathcal{R}}$

$$\frac{\mathcal{N}}{\mathcal{N}^r_{(\neg)\nabla}}$$ provided: $\mathcal{N}$ is positive (negative) w.r.t. r.

Notation in the Ackermann$^{\mathcal{R}}$ rule ($1 \leq j \leq k$, $1 \leq l \leq u$):
$\mathcal{T}\text{-}Block^{\mathcal{H}}(j,m)$ denotes the set

$$\{C^j \sqcup C^Y \sqcup \neg\forall\mathcal{H}^j.(D^j \sqcup D^Y) | Y \subseteq [m]\}, \text{ where}$$

$$C^Y = \begin{cases} \neg\top & \text{if } Y = \emptyset \\ \bigsqcup_{i \in Y} C_i & \text{otherwise,} \end{cases} \quad D^Y = \begin{cases} \neg\top & \text{if } Y = \emptyset \\ \bigsqcup_{i \in Y} \neg D_i & \text{otherwise,} \end{cases}$$

$$\text{and} \quad \mathcal{H}^j = \begin{cases} S_1 \sqcap \ldots \sqcap S_n \sqcap Q^j & \text{if } \mathcal{P}^-_{\mathcal{R}} \neq \emptyset \\ \nabla \sqcap Q^j & \text{otherwise.} \end{cases}$$

$\mathcal{R}\text{-}Block^{\mathcal{C}}(l,m)$ denotes the set

$$\{C_1 \sqcup \forall T^l.D_1, \ldots, C_m \sqcup \forall T^l.D_m\}.$$

$\mathcal{R}\text{-}Block^{\mathcal{R}}(l,n)$ denotes the set $\{\neg T^l \sqcup S_1, \ldots, \neg T^l \sqcup S_n\}$.

Figure 5: Rules for forgetting pivot $r \in \mathsf{N_R}$

Since the underlying language accommodates role inverse, the calculus includes two Role Surfacing rules (shown in Figure 4) to reformulate expressions without occurrences of inverses of the pivot in the TBox and RBox clauses, and to free the other rules in the calculus from needing to cater for role inverse. The Role Surfacing rules are applied after definer symbols have been introduced (if necessary), and before the application of the forgetting rules.

Theorem 4 (Ackermann Lemma, Role Forgetting) *Let $\mathcal{I}$ be any $\mathcal{ALCOIH}\mu^+(\nabla,\sqcap)$-interpretation. For r the pivot, when the Ackermann$^{\mathcal{R}}$ or Purify$^{\mathcal{R}}$ rule is applicable, the conclusion of the rule is true in $\mathcal{I}$ iff for some interpretation $\mathcal{I}'$ r-equivalent to $\mathcal{I}$, the premises are true in $\mathcal{I}'$.*

Given a set $\mathcal{N}$ of clauses with $r \in \mathsf{N_R}$ being the pivot, once $\mathcal{N}$ has been transformed into r-reduced form, we apply the Ackermann$^{\mathcal{R}}$ rule given in Figure 5 to eliminate r. The Purify$^{\mathcal{R}}$ rule can be applied any time provided that r is pure. By definition, clauses in r-reduced form have four distinct forms. We refer to the clauses of the form $C^j \sqcup \neg\forall r \sqcap Q^j.D^j$ ($1 \leq j \leq k$) as *positive TBox premises* (denoted by $\mathcal{P}^+_{\mathcal{T}}$)

and clauses of the form $C_i \sqcup \forall r.D_i$ $(1 \leq i \leq m)$ as *negative TBox premises* (denoted by $\mathcal{P}_{\mathcal{T}}^-$) of the rule. We refer to the clauses of the form $\neg T^l \sqcup r$ $(1 \leq l \leq u)$ as *positive RBox premises* (denoted by $\mathcal{P}_{\mathcal{R}}^+$) and clauses of the form $\neg r \sqcup S_i$ $(1 \leq i \leq n)$ as *negative RBox premises* (denoted by $\mathcal{P}_{\mathcal{R}}^-$) of the rule. Since it is possible for r to occur in any of these premises (and the premises do not necessarily all exist), there are several situations where the Ackermann$^{\mathcal{R}}$ rule is applicable. For different $\mathcal{P}_{\mathcal{T}}^-$ and $\mathcal{P}_{\mathcal{R}}^-$ the Ackermann$^{\mathcal{R}}$ rule is performed as follows.

Case (I): If $\mathcal{P}_{\mathcal{T}}^- \neq \emptyset$ and $\mathcal{P}_{\mathcal{R}}^- \neq \emptyset$, then $\mathcal{P}_{\mathcal{T}}^-$ and $\mathcal{P}_{\mathcal{R}}^-$ (the negative premises) are combined with every clause in $\mathcal{P}_{\mathcal{T}}^+$ (if $\mathcal{P}_{\mathcal{T}}^+ \neq \emptyset$) and every clause in $\mathcal{P}_{\mathcal{R}}^+$ (if $\mathcal{P}_{\mathcal{R}}^+ \neq \emptyset$), which leads to the elimination of r, and a forgetting solution $\mathcal{O}'$ such that $r \notin \text{sig}(\mathcal{O}')$. Specifically, combining $\mathcal{P}_{\mathcal{T}}^-$ and $\mathcal{P}_{\mathcal{R}}^-$ with one of the positive TBox premises ($\mathcal{P}_{\mathcal{T}}^+$) yields a set of TBox clauses (denoted by $\mathcal{T}\text{-}Block^{\mathcal{H}}(j,m)$), where $\mathcal{H}^j$ is the conjunction of Q^j and the S_i $(1 \leq i \leq n)$ in $\mathcal{P}_{\mathcal{R}}^-$ ($\mathcal{H}^j = Q^j \sqcap S_1 \sqcap \ldots \sqcap S_n$), and m denotes the number of negative TBox premises ($|\mathcal{P}_{\mathcal{T}}^-|$), and j refers to the positive TBox premise with which $\mathcal{P}_{\mathcal{T}}^-$ and $\mathcal{P}_{\mathcal{R}}^-$ combine. $[m]$ denotes the set $\{1,\ldots,m\}$, and $\{C^j \sqcup C^Y \sqcup \neg\forall\mathcal{H}^j.(D^j \sqcup D^Y) | Y \subseteq [m]\}$ is the set that contains all clauses corresponding to every assignment of Y. The following example illustrates this case.

Example 1 *Given $\Sigma = \{r\}$ and a set $\mathcal{N}$ of clauses in r-reduced form with $\mathcal{P}_{\mathcal{T}}^- = \{A_1 \sqcup \forall r.B_1, A_2 \sqcup \forall r.B_2\}$, $\mathcal{P}_{\mathcal{R}}^- = \{\neg r \sqcup s_1, \neg r \sqcup s_2^-\}$, and $\neg a \sqcup \neg\forall r.B \in \mathcal{P}_{\mathcal{T}}^+$, combining $\mathcal{P}_{\mathcal{T}}^-$ and $\mathcal{P}_{\mathcal{R}}^-$ with $\neg a \sqcup \neg\forall r.B$ leads to $\mathcal{T}\text{-}Block^{\mathcal{H}}(j,m) = \{\neg a \sqcup C^Y \sqcup \neg\forall(s_1 \sqcap s_2^-).(B \sqcup D^Y) | Y \subseteq [2]\}$ that consists of the following clauses:*

$$\neg a \sqcup \underbrace{\neg\top}_{C^Y} \sqcup \neg\forall(s_1 \sqcap s_2^-).(B \sqcup \underbrace{\neg\top}_{D^Y}) \quad \textit{if } Y = \emptyset$$

$$\neg a \sqcup \underbrace{A_1}_{C^Y} \sqcup \neg\forall(s_1 \sqcap s_2^-).(B \sqcup \underbrace{\neg B_1}_{D^Y}) \quad \textit{if } Y = \{1\}$$

$$\neg a \sqcup \underbrace{A_2}_{C^Y} \sqcup \neg\forall(s_1 \sqcap s_2^-).(B \sqcup \underbrace{\neg B_2}_{D^Y}) \quad \textit{if } Y = \{2\}$$

$$\neg a \sqcup \underbrace{A_1 \sqcup A_2}_{C^Y} \sqcup \neg\forall(s_1 \sqcap s_2^-).(B \sqcup \underbrace{\neg B_1 \sqcup \neg B_2}_{D^Y}) \quad \textit{if } Y = \{1,2\}.$$

Combining $\mathcal{P}_{\mathcal{T}}^-$ with one of the positive RBox premises yields a set of TBox clauses (denoted by $\mathcal{R}\text{-}Block^{\mathcal{C}}(l,m)$), where m ranges over the negative TBox premises and l refers to the positive RBox premise with which $\mathcal{P}_{\mathcal{T}}^-$ combines; combining $\mathcal{P}_{\mathcal{R}}^-$ with one of the positive RBox premises yields a set of RBox clauses (denoted by $\mathcal{R}\text{-}Block^{\mathcal{R}}(l,n)$), where n ranges over the negative RBox premises and l refers to the positive RBox premise with which $\mathcal{P}_{\mathcal{R}}^-$ combines.

Case (II): If $\mathcal{P}_{\mathcal{T}}^- \neq \emptyset$ and $\mathcal{P}_{\mathcal{R}}^- = \emptyset$, then combining $\mathcal{P}_{\mathcal{T}}^-$ with one of the positive TBox premises yields the same result as in Case (I) ($\mathcal{T}\text{-}Block^{\mathcal{H}}(j,m)$), only that ∇ replaces $S_1 \sqcap \ldots \sqcap S_n$ in $\mathcal{H}^j$, i.e., $\mathcal{H}^j = \nabla \sqcap Q^j$ $(1 \leq j \leq k)$. Combining $\mathcal{P}_{\mathcal{T}}^-$ with one of the positive RBox premises yields the same result as in Case (I), i.e., $\mathcal{R}\text{-}Block^{\mathcal{C}}(l,m)$ $(1 \leq l \leq u)$. **Case (III)**: If $\mathcal{P}_{\mathcal{T}}^- = \emptyset$ and $\mathcal{P}_{\mathcal{R}}^- \neq \emptyset$, then combining $\mathcal{P}_{\mathcal{R}}^-$ with one of the positive TBox premises and with one of the positive RBox premises yields the same results as in Case (I), i.e., $\mathcal{T}\text{-}Block^{\mathcal{H}}(j,m)$ $(1 \leq j \leq k)$ and $\mathcal{R}\text{-}Block^{\mathcal{R}}(l,n)$ $(1 \leq l \leq u)$, respectively. **Case (IV)**: The case $\mathcal{P}_{\mathcal{T}}^- = \emptyset$ and $\mathcal{P}_{\mathcal{R}}^- = \emptyset$ can be seen as an instance of the case where r is pure and then eliminated using the Purify$^{\mathcal{R}}$ rule.

The pivot is forgotten in the ontology once every premise in $\mathcal{P}_{\mathcal{T}}^+$ and every premise in $\mathcal{P}_{\mathcal{R}}^+$, i.e., the positive premises, have been combined with $\mathcal{P}_{\mathcal{T}}^-$ (if $\mathcal{P}_{\mathcal{T}}^- \neq \emptyset$) and $\mathcal{P}_{\mathcal{R}}^-$ (if $\mathcal{P}_{\mathcal{R}}^- \neq \emptyset$). Given a set of clauses in pivot-reduced form and m negative TBox premises ($|\mathcal{P}_{\mathcal{T}}^-| = m$), n negative RBox premises ($|\mathcal{P}_{\mathcal{R}}^-| = n$), k positive TBox premises ($|\mathcal{P}_{\mathcal{T}}^+| = k$), and u positive RBox premises ($|\mathcal{P}_{\mathcal{R}}^+| = u$), combining $\mathcal{P}_{\mathcal{T}}^-$ and $\mathcal{P}_{\mathcal{R}}^-$ with all positive TBox premises yields $k2^m$ clauses (exponential growth); combining $\mathcal{P}_{\mathcal{T}}^-$ and $\mathcal{P}_{\mathcal{R}}^-$ with all positive RBox premises yields $um + un$ clauses (polynomial growth). The size of the forgetting solution therefore depends largely upon the number of the negative TBox premises (m).

6 Evaluation and Empirical Results

We have implemented a prototype of the forgetting method in Java using the OWL-API[1] and conducted two series of experiments on real-world ontologies to evaluate the practicality of the method. The experiments were run on a desktop with an Intel® Core™ i7-4790 processor, and four cores running at up to 3.60 GHz and 8 GB of DDR3-1600 MHz RAM. The ontologies used for our evaluation were taken from the NCBO BioPortal[2] and Oxford Ontology[3] repositories and were restricted to the $\mathcal{ALCOIH}(\nabla,\sqcap)$-fragments; any subconcepts beyond the scope of $\mathcal{ALCOIH}(\nabla,\sqcap)$ were replaced by $\top$. Consequently, 180 and 200 ontologies of various sizes were selected from the NCBO BioPortal and Oxford Ontology repositories, respectively. We repeated the experiments 100 times on each ontology and averaged the results to verify the accuracy of our findings.

To fit in with real-world applications such as computing logical difference between ontologies and predicate hiding, where forgetting a small number of symbols is in demand, we set up a series of experiments where we forgot 10% and 30% of concept and role symbols in each ontology. There are also situations where it would be of interest to forget a large number of symbols; ontology reuse is such an example [Koopmann and Schmidt, 2013a; 2013b; 2014; 2015]. We therefore set up another series of experiments where we forgot 80% of the concept symbols and 50% of the role symbols in each ontology. The symbols to be forgotten were randomly selected in the experiments. The heuristic for determining the orders of eliminating concept and role symbols ($\succ_{\mathcal{C}}$ and $\succ_{\mathcal{R}}$) was also tested. When the heuristic was not applied, the Σ-symbols were eliminated in the order as returned by the OWL-API function that gets all concept and role symbols in the ontology. We started the evaluation with concept forgetting, where a timeout of 15 minutes was used.

The results obtained from forgetting 10%, 30% and 80%

[1]http://owlapi.sourceforge.net/

[2]http://bioportal.bioontology.org/

[3]http://www.cs.ox.ac.uk/isg/ontologies/

Input	Setting		Results			
Ontology	$\|\Sigma\|$ (%)	$\succ_{\mathcal{C}}$	Time (sec.)	Timeout	Success R.	Fixp.
BioPortal (354 Axioms on Avg.)	20 (10%)	✗	4.890	0.0%	100.0%	0.0%
		✓	3.260	0.0%	100.0%	0.0%
	60 (30%)	✗	18.672	4.4%	94.4%	7.2%
		✓	9.336	1.1%	98.3%	7.8%
	160 (80%)	✗	70.416	13.8%	83.3%	13.3%
		✓	29.340	5.6%	91.7%	17.2%
	80 Avg.	✗	**31.326**	**6.3%**	**92.6%**	**6.8%**
		✓	**13.979**	**2.4%**	**96.7%**	**8.3%**
Oxford (875 Axioms on Avg.)	36 (10%)	✗	44.392	3.0%	97.0%	0.5%
		✓	27.745	1.5%	98.5%	0.5%
	108 (30%)	✗	193.106	17.0%	79.5%	11.5%
		✓	80.461	9.5%	88.5%	14.5%
	288 (80%)	✗	412.852	34.5%	61.5%	19.0%
		✓	166.270	17.5%	78.5%	26.5%
	144 Avg.	✗	**216.783**	**18.2%**	**79.3%**	**10.3%**
		✓	**91.492**	**9.5%**	**88.5%**	**13.8%**

Figure 6: Evaluation results for concept forgetting

of the concept symbols from the respective BioPortal and Oxford ontologies are shown in Figure 6, which is revealing in several ways. The most notable observation to emerge from the results is that, with the heuristically determined forgetting ordering (indicated by ✓), our implementation was successful (forgot all the concept symbols in Σ) in 96.7% of the BioPortal-cases within a short period of time, with 8.3% of them using fixpoints in the result. In the experiments of 10% of the concept symbols specified to be forgotten, the success rate rose to 100%. Even when not using the heuristic (✗), the forgetting solution could be found in 92.6% of the cases. Since the Oxford ontologies were more than twice as large as the BioPortal ontologies and a larger set of concept symbols was specified to be forgotten, a reduction in the performance was expected. The implementation was unable to compute the solution in 11.5% of the Oxford-cases, and in 13.8% of the solved ones fixpoints occurred in the result. The use of the heuristic boosted the overall success rate by 4% and 9.2%, and improved the time efficiency by 124.1% and 136.9% in the BioPortal and Oxford ontologies, respectively.

Setting		Results				
$\|\Sigma\|$ (%)	$\succ_{\mathcal{R}}$	Definer in	Time (sec.)	Success R.	D. Left	Clause ↑
4 (10%)	✗	1.1/onto.	2.120 sec.	100.0%	0	4.1%
	✓	10 onto.	2.101 sec.	100.0%	0	4.1%
12 (30%)	✗	1.9/onto.	8.658 sec.	100.0%	0	14.5%
	✓	13 onto.	8.314 sec.	100.0%	0	14.5%
20 (50%)	✗	3.0/onto.	20.913 sec.	100.0%	0	26.5%
	✓	16 onto.	20.566 sec.	100.0%	0	26.5%
Avg.	✗	**2.0/onto.**	**10.564 sec.**	**100.0%**	**0**	**15.0%**
	✓	**13 onto.**	**10.327 sec.**	**100.0%**	**0**	**15.0%**
5 (10%)	✗	2.4/onto.	3.187 sec.	100.0%	0	2.2%
	✓	25 onto.	3.072 sec.	100.0%	0	2.2%
15 (30%)	✗	3.7/onto.	18.537 sec.	100.0%	0	6.9%
	✓	28 onto.	17.998 sec.	100.0%	0	6.9%
25 (50%)	✗	5.7/onto.	36.292 sec.	100.0%	0	14.6%
	✓	39 onto.	34.117 sec.	100.0%	0	14.6%
Avg.	✗	**3.9/onto.**	**19.339 sec.**	**100.0%**	**0**	**7.9%**
	✓	**30 onto.**	**18.396 sec.**	**100.0%**	**0**	**7.9%**

Figure 7: Evaluation results for role forgetting

We evaluated the performance of forgetting different numbers of role symbols with the same ontologies used for the evaluation of concept forgetting (using a timeout of 5 minutes). The results are shown in Figure 7, from which it can be seen that our implementation was successful (forgot all the role symbols in Σ) in all cases. The time used for forgetting role symbols (including the time for the elimination of definer symbols), as expected, was significantly longer than forgetting the same number of concept symbols, despite the 100% success rate. Because of the nature of the Ackermann$^{\mathcal{R}}$ rule, role forgetting leads to growth of clauses in the forgetting solution, which was however modest (see Clause ↑) compared to the theoretical worst case. Definer symbols were introduced only in a small proportion of the ontologies to help conversion to reduced form. This indicates that most clauses in the ontologies were flat. By $m/onto$, we mean there were on average m definer symbols introduced in each ontology, and by n $onto$, we mean that n of all ontologies introduced definer symbols. These definer symbols were successfully eliminated in the subsequent definer elimination phase (D. Left).

Suppose that $r \in \mathsf{N_R}$ is the pivot and a set of clauses is in r-reduced form with $|\mathcal{P}^+_{\mathcal{T}}| \geq 1$ (since role conjunction can only occur in the positive TBox premises). Role forgetting can lead to role conjunctions occurring in forgetting solutions in these situations: (i) Role conjunction already occurs in at least one of the positive TBox premises, i.e., $C \sqcup \neg\forall r \sqcap Q.D \in \mathcal{P}^+_{\mathcal{T}}$, where Q is a role. (ii) If none of the positive TBox premises includes role conjunction, then having at least two negative RBox premises ($|\mathcal{P}^-_{\mathcal{R}}| \geq 2$) yields a forgetting solution including role conjunction. In our evaluation it was however found that neither of these cases happened often; in most cases, rather the opposite occurred: the positive TBox premises were in the simple form $C \sqcup \neg\forall r.D$ and often $|\mathcal{P}^-_{\mathcal{R}}| \leq 1$. Only in 8.9% of the tested cases role conjunction occurred in the forgetting solutions.

7 Conclusion and Future Work

In this paper we have developed a method of concept and role forgetting for expressive description logics with nominals, non-empty TBoxes and ABoxes, and a rich language for expressing properties of roles. The method can handle role inclusion statements, role conjunctions and role inverses. This is extremely useful from the perspective of ontology engineering as it increases the arsenal of tools available to create restricted views of ontologies. The results of the evaluation on real-world ontologies have shown that often fixpoints and role conjunction are not needed to express forgetting solutions, and overall the performance results are very positive.

Currently beyond the scope of our method are forgetting transitive roles. We can extend the method to eliminate concept symbols in the presence of transitive roles, but when forgetting role symbols the interaction between transitive roles and role inclusion statements can lead to results where it is not clear how to represent them finitely; see [Koopmann, 2015] for an example. The problem of extending the method to handle role functionality and numerical quantifier restrictions remains completely open; possibly the techniques of [Koopmann and Schmidt, 2014; 2015] can help extend the method.

Acknowledgments

We thank the anonymous reviewers for useful comments.

References

[Ackermann, 1935] W. Ackermann. Untersuchungen über das Eliminationsproblem der mathematischen Logik. *Mathematische Annalen*, 110(1):390–413, 1935.

[Conradie *et al.*, 2006] W. Conradie, V. Goranko, and D. Vakarelov. Algorithmic correspondence and completeness in modal logic. I. The core algorithm SQEMA. *Logical Methods in Computer Science*, 2(1), 2006.

[D'Agostino and Hollenberg, 2000] G. D'Agostino and M. Hollenberg. Logical questions concerning the μ-Calculus: Interpolation, Lyndon and Los-Tarski. *Journal of Symbolic Logic*, 65(1):310–332, 2000.

[Doherty *et al.*, 1997] P. Doherty, W. Łukaszewicz, and A. Szałas. Computing circumscription revisited: A reduction algorithm. *Journal of Automated Reasoning*, 18(3):297–336, 1997.

[Gabbay and Ohlbach, 1992] D. M. Gabbay and H. J. Ohlbach. Quantifier elimination in second–order predicate logic. In *Proc. KR'92*, pp. 425–435. Morgan Kaufmann, 1992.

[Gabbay *et al.*, 2008] D. M. Gabbay, R. A. Schmidt, and A. Szałas. *Second Order Quantifier Elimination*. College Publ., 2008.

[Goranko and Vakarelov, 2006] V. Goranko and D. Vakarelov. Elementary canonical formulae: extending Sahlqvist's theorem. *Annals of Pure and Applied Logic*, 141(1-2):180–217, 2006.

[Grädel and Walukiewicz, 1999] E. Grädel and I. Walukiewicz. Guarded fixed point logic. In *Proc. LICS'99*, pp. 45–54. IEEE Computer Society, 1999.

[Herzig and Mengin, 2008] A. Herzig and J. Mengin. Uniform interpolation by resolution in modal logic. In *Proc. JELIA'08*, vol. 5293 of *LNCS*, pp. 219–231. Springer, 2008.

[Konev *et al.*, 2009a] B. Konev, C. Lutz, D. Walther, and F. Wolter. Formal properties of modularisation. In *Modular Ontologies*, vol. 5445 of *LNCS*, pp. 25–66. Springer, 2009.

[Konev *et al.*, 2009b] B. Konev, D. Walther, and F. Wolter. Forgetting and uniform interpolation in extensions of the description logic $\mathcal{EL}$. In *Proc. DL'09*, vol. 477 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[Konev *et al.*, 2013] B. Konev, C. Lutz, D. Walther, and F. Wolter. Model-theoretic inseparability and modularity of description logic ontologies. *Artificial Intelligence*, 203:66–103, 2013.

[Koopmann and Schmidt, 2013a] P. Koopmann and R. A. Schmidt. Forgetting concept and role symbols in $\mathcal{ALCH}$-ontologies. In *Proc. LPAR'13*, vol. 8312 of *LNCS*, pp. 552–567. Springer, 2013.

[Koopmann and Schmidt, 2013b] P. Koopmann and R. A. Schmidt. Uniform interpolation of $\mathcal{ALC}$-ontologies using fixpoints. In *Proc. FroCoS'13*, vol. 8152 of *LNCS*, pp. 87-102. Springer, 2013.

[Koopmann and Schmidt, 2014] P. Koopmann and R. A. Schmidt. Count and forget: Uniform interpolation of $\mathcal{SHQ}$-ontologies. In *Proc. IJCAR'14*, vol. 8562 of *LNCS*, pp. 434–448. Springer, 2014.

[Koopmann and Schmidt, 2015] P. Koopmann and R. A. Schmidt. Saturated-based forgetting in the description logic $\mathcal{SIF}$. In *Proc. DL'15*, vol. 1350 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[Koopmann, 2015] P. Koopmann. *Practical uniform interpolation for expressive description logics*. PhD thesis, University of Manchester, 2015.

[Lin and Reiter, 1994] F. Lin and R. Reiter. Forget it! In *Proc. AAAI Fall Symposium on Relevance*, pp. 154–159, 1994.

[Ludwig and Konev, 2014] M. Ludwig and B. Konev. Practical uniform interpolation and forgetting for $\mathcal{ALC}$ TBoxes with applications to logical difference. In *Proc. KR'14*. AAAI Press, 2014.

[Lutz and Wolter, 2011] C. Lutz and F. Wolter. Foundations for uniform interpolation and forgetting in expressive description logics. In *Proc. IJCAI'11*, pp. 989–995. IJCAI/AAAI, 2011.

[Lutz *et al.*, 2012] C. Lutz, I. Seylan, and F. Wolter. An automata-theoretic approach to uniform interpolation and approximation in the description logic $\mathcal{EL}$. In *Proc. KR'12*, pp. 286–297. AAAI Press, 2012.

[Nikitina and Rudolph, 2014] N. Nikitina and S. Rudolph. (Non-)Succinctness of Uniform Interpolants of General Terminologies in the Description Logic $\mathcal{EL}$. *Artificial Intelligence*, 215:120–140, 2014.

[Nonnengart and Szałas, 1999] A. Nonnengart and A. Szałas. A fixpoint approach to second-order quantifier elimination with applications to correspondence theory. In E. Orlowska, editor, *Logic at Work*, pp. 307–328. Springer, 1999.

[Sahlqvist, 1975] H. Sahlqvist. Completeness and correspondence in the first and second order semantics for modal logics. In *Proc. 3rd Scandinavian Logic Symposium, 1973*, pp. 110–143. North-Holland, 1975.

[Schmidt and Tishkovsky, 2014] R. A. Schmidt and D. Tishkovsky. Using tableau to decide description logics with full role negation and identity. *ACM Transactions of Computational Logic*, 15(1):7:1–7:31, 2014.

[Schmidt, 2012] R. A. Schmidt. The Ackermann approach for modal logic, correspondence theory and second-order reduction. *Journal of Applied Logic*, 10(1):52–74, 2012.

[Szałas, 1993] A. Szałas. On the correspondence between modal and classical logic: An automated approach. *Journal of Logic and Computation*, 3:605–620, 1993.

[Szałas, 2006] A. Szałas. Second-order reasoning in description logics. *Journal of Applied Non-Classical Logics*, 16(3-4):517–530, 2006.

[Tishkovsky *et al.*, 2012] D. Tishkovsky, R. A. Schmidt, and M. Khodadadi. The tableau prover generator MetTeL2. In *Proc. JELIA'12*, vol. 7519 of *LNCS*, pp. 492–495. Springer, 2012.

[Visser, 1996] A. Visser. *Bisimulations, Model Descriptions and Propositional Quantifiers*. Logic Group Preprint Series. Department of Philosophy, Utrecht Univ., 1996.

[Wang *et al.*, 2008] Z. Wang, K. Wang, R. W. Topor, and J. Z. Pan. Forgetting concepts in DL-Lite. In *Proc. ESWC'08*, vol. 5021 of *LNCS*, pp. 245–257. Springer, 2008.

[Wang *et al.*, 2009] K. Wang, Z. Wang, R. Topor, J. Z. Pan, and G. Antoniou. Concept and role forgetting in $\mathcal{ALC}$ ontologies. In *Proc. ISWC'09*, vol. 5823 of *LNCS*, pp. 666–681. Springer, 2009.

[Wang *et al.*, 2014] K. Wang, Z. Wang, R. Topor, J. Z. Pan, and G. Antoniou. Eliminating concepts and roles from ontologies in expressive description logics. *Computational Intelligence*, 30(2):205–232, 2014.

[Wernhard, 2011] C. Wernhard. Computing with logic as operator elimination: The ToyElim system. In *Proc. INAP/WLP'11*, vol. 7773 of *LNCS*, pp. 289–296. Springer, 2011.

[Zhao and Schmidt, 2015] Y. Zhao and R. A. Schmidt. Concept Forgetting in $\mathcal{ALCOI}$-Ontologies Using an Ackermann Approach. In *Proc. ISWC'15*, vol. 9366 of *LNCS*, pp. 587–602. Springer, 2015.