# 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 that is concerned with creating restricted views of ontologies relative to a subset of the initial signature and preserving pertinent 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 and ABox statements (via nominals), and 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 (UI) in this area (or, variable elimination, projection or second-order quantifier elimination in knowledge representation and logic) 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., [11, 18, 19, 21, 25–27], and the foundational studies of the properties of forgetting for description logics by, e.g., [10, 19, 20, 28]. These works give arguments for the important role of forgetting in realising various tasks that are crucial for effective processing and management of ontologies. For example, forgetting can be used for ontology analysis and summary, 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 realised to be significantly harder than forgetting of concept symbols [28]. The dominant reason is probably that the result of forgetting role symbols may not always be expressible in the source language. Although the earliest work did study concept and role forgetting, e.g. [25], most subsequent work considered only concept forgetting with the exception of [12–15]. 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 $\mathcal{SHI}$) [16].

The work on forgetting for description logics is predated by work on second-order quantifier elimination [5–7], which can be traced to questions posed by Boole and seminal work of [1]. These works triggered and influenced work in knowledge representation [17], but also led to striking results in the automation of correspondence theory of modal logics [3, 22]. Because of the close relationship between description logics and modal logics, besides the work on modal correspondence theory, investigations of UI for modal logics [4, 9, 24] are relevant. These are particularly 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 $\mu$, the top role $\nabla$ and role intersection $\sqcap$. This means the method is one of only few approaches that can eliminate role symbols, while also handling role inverse and ABox statements via nominals, and the only approach so far providing support for forgetting in description logics with nominals. Being based on the Ackermann approach to second-order quantifier elimination [5, 22, 29] the method terminates always. We have shown the method is correct, i.e., the forgetting solution computed is equivalent to the original ontology up to the symbols that have been eliminated. A general problem of forgetting is marking out a language that is expressive enough to allow for solutions to be expressed by finitely many formulas. Completeness results are rare and become harder to achieve with increased expressivity, unless one is willing to admit second-order quantification, for which the forgetting problem becomes trivially solvable. Despite our method not being complete, results of an evaluation with a prototypical implementation have shown high success rates on real-world ontologies of various sizes.

## 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. Let $\mathsf{N}_\mu$ be a set of *concept variables* disjoint with $\mathsf{N_C}$, $\mathsf{N_R}$ and $\mathsf{N_O}$. 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 $\mathcal{ALCOIH}(\nabla, \sqcap)$-*concepts*, and $R$ is any role expression. Role expressions in $\mathcal{ALCOIH}(\nabla, \sqcap)$ can be the top role $\nabla$, a role symbol $r \in \mathsf{N_R}$, the inverse $r^-$ of a role symbol $r$, or a conjunction of these. 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 negations). Moreover, $\mu$-expressions may occur only positively. Because of this restriction, $\mu$-expressions can be simulated in $\mathcal{ALCOIH}(\nabla, \sqcap)$ with auxiliary concept symbols as in [13].

The semantics of the $\mathcal{ALCOIH}(\nabla, \sqcap)$-fragment is as expected. Intuitively, $\mu X.C[X]$ denotes the least general concept $C_\mu$ w.r.t. a concept expression for

which $C_\mu \equiv C[C_\mu]$ holds, where $C[C_\mu]$ is a concept expression obtained from replacing every occurrence of $X$ in $C[X]$ by $C_\mu$. A formal definition of the semantics of fixpoint expressions can be found in [2].

We assume without loss of generality 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 $\mu$. An RBox $\mathcal{R}$ is a set of *role axioms* of the form $R \sqsubseteq S$, where $R$ and $S$ is a role symbol or an inverted role symbol. Nominals in a description logic make ABox assertions superfluous, since these can be captured by TBox inclusions via nominals. An $\mathcal{ALCOIH}\mu^+(\triangledown,\sqcap)$-ontology is therefore assumed to be the union of the TBox and the RBox in this paper.

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

Theorem 1 follows from the decidability results of guarded fixpoint logic [8] and the description logic $\mathcal{ALBO}^{\mathsf{id}}$ [23], which both subsume the languages of $\mathcal{ALCOIH}(\triangledown,\sqcap)$ and $\mathcal{ALCOIH}\mu^+(\triangledown,\sqcap)$.

In the sequel we describe the normal form on which our forgetting method works. A *TBox clause* is a disjunction of $\mathcal{ALCOIH}\mu^+(\triangledown,\sqcap)$-concepts, in which role expressions can be a role symbol, an inverted role symbol, or a conjunction of these. A *role literal* is either a role symbol or an inverted role symbol, or their negations. An *RBox clause* is a disjunction of two role literals of complementary polarity. RBox clauses are transformed from the clausification of role axioms, where role negation is introduced.

A clause that contains occurrences of a designated concept symbol and 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 negations. Otherwise it is *negative*. A role symbol that occurs immediately under a (negated) universal role restriction is assumed to be negative (positive). $C$ 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*).

By $\mathrm{sig}(E)$ we denote the concept and role symbols occurring in $E$, where $E$ ranges over concepts, clauses, and ontologies. $\Sigma$ is assumed to be the concept and role symbols to be forgotten in this paper. 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* $\Sigma$, or $\Sigma$-*equivalent*, if $\mathcal{I}$ and $\mathcal{I}'$ are the same but differ possibly in the interpretations of the symbols in $\Sigma$.

**Definition 1 (Forgetting in $\mathcal{ALCOIH}\mu^+(\triangledown,\sqcap)$-Ontologies).** *Let $\mathcal{O}$ and $\mathcal{O}'$ be $\mathcal{ALCOIH}\mu^+(\triangledown,\sqcap)$-ontologies and let the signature be $\Sigma \subseteq \mathrm{sig}(\mathcal{O})$. $\mathcal{O}'$ is the* solution of forgetting $\Sigma$-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 $\mathcal{I}'$ $\Sigma$-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 $\Sigma$. It follows that if both $\mathcal{O}'$ and $\mathcal{O}''$ are solutions of forgetting symbols in $\Sigma$ from $\mathcal{O}$, then they are equivalent.

## 3 Overview of the Forgetting Method

The forgetting process in our method consists mainly of three phases: the reduction to a set of $\mathcal{ALCOIH}\mu^+(\triangledown, \sqcap)$-clauses, the forgetting phase and the definer elimination phase (see Figure 1).

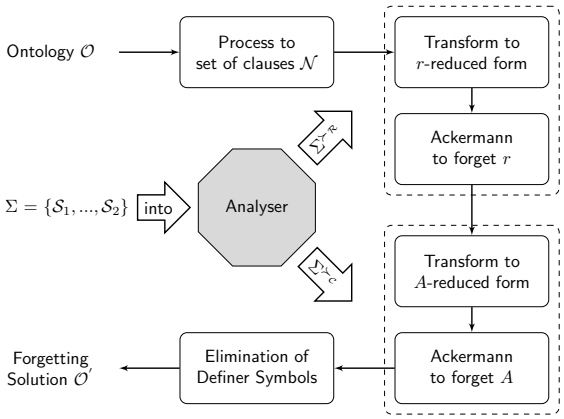


**Fig. 1.** Overview of the forgetting method

Initially given, as the input to the method, are an ontology $\mathcal{O}$ of TBox and RBox axioms expressible in $\mathcal{ALCOIH}\mu^+(\triangledown, \sqcap)$, and a set $\Sigma$ that contains the concept and role symbols to be forgotten. The $\Sigma$-symbols are entirely determined by the users and their application demands; $\Sigma$ can thus be any subset of the signature of the initial ontology as the user wishes. Once received by the system, $\mathcal{O}$ is transformed into a set $\mathcal{N}$ of clauses, i.e., the normal form of our method.

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, that is, the rules for concept forgetting and the rules for role forgetting are mutually independent; concept and role symbols can thus be forgotten in any desired order. In the forgetting phase, in order for the forgetting rules to be applied to eliminate $\mathcal{S} \in \Sigma$, the $\mathcal{S}$-clauses must be transformed into $\mathcal{S}$-*reduced form*. Thus, whether the $\Sigma$-symbols are eliminable depends entirely on whether the current set of clauses can be transformed into reduced form. This reduction is performed using two Ackermann-based calculi working independently, which also lead to goal-oriented elimination of the concept and role symbols. The calculi are described in the next sections. Equivalence-preserving simplification rules are applied throughout the forgetting process, ensuring that the current clauses are always in simpler representations for efficiency. What the method returns, if the forgetting is successful, is an ontology $\mathcal{O}'$ that does not contain the symbols in $\Sigma$, i.e., the returned ontology is formulated using only symbols in $\text{sig}(\mathcal{O})\backslash\Sigma$.

Previous research has shown that the success rates of forgetting depend very much on the order in which the $\Sigma$-symbols are forgotten [3, 5, 6, 14, 22], even

when forgetting only concept symbols, e.g., [29, 18]. An important challenge therefore is to find good orders of eliminating $\Sigma$-symbols. Our method either follows a user-specified order, or it uses a heuristic analysis based on the frequency counts of the $\Sigma$-symbols, where concept symbols and role symbols are analysed separately.

We refer to the maximal symbol of $\Sigma$ w.r.t. the forgetting order $\succ$ as the *pivot* in our method. Following $\succ$ (and starting with the pivot), the method forgets the $\Sigma$-symbols one by one. If the pivot is successfully eliminated from $\mathcal{N}$, we attempt to forget the next symbol in $\succ$, which has become the new pivot. Otherwise the pivot remains in $\Sigma$, flagged as a currently non-forgettable symbol, and the next symbol in $\succ$ 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 the 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 $\Sigma$-symbols remaining, this means that the method fails to find a forgetting solution.

**Theorem 2.** *For any $\mathcal{ALCOIH}\mu^+(\triangledown, \sqcap)$-ontology $\mathcal{O}$ and any $\Sigma \subseteq \mathrm{sig}(\mathcal{O})$, the method always terminates and returns a set $\mathcal{O}'$ of clauses. If $\mathcal{O}'$ does not contain any $\Sigma$-symbols, the method was* successful. *$\mathcal{O}'$ is then a solution of forgetting the symbols in $\Sigma$ from $\mathcal{O}$. If neither $\mathcal{O}$ nor $\mathcal{O}'$ uses $\mu$, $\mathcal{O}'$ is $\Sigma$-equivalent to $\mathcal{O}$ in $\mathcal{ALCOIH}(\triangledown, \sqcap)$. Otherwise, it is $\Sigma$-equivalent to $\mathcal{O}$ in $\mathcal{ALCOIH}\mu^+(\triangledown, \sqcap)$.*

## 4 Calculus for Concept Forgetting

We first present the calculus for forgetting of *concept symbols* in ontologies expressible in $\mathcal{ALCOIH}\mu^+(\triangledown, \sqcap)$. The calculus extends the calculus of [29] for concept forgetting in $\mathcal{ALCOI}$. Since concept symbols occur only in TBox axioms, only the TBox needs to be processed for concept forgetting.

**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$^{\mathcal{C},p}$**

$$\frac{\mathcal{N}}{\mathcal{N}^A_{\top}}$$

$\mathcal{N}$ is positive w.r.t. $A$.

**Purify$^{\mathcal{C},n}$**

$$\frac{\mathcal{N}}{\mathcal{N}^A_{\neg\top}}$$

$\mathcal{N}$ is negative w.r.t. $A$.

**Fig. 2.** Forgetting concept symbol $A$ in a set $\mathcal{N}$ of $\mathcal{ALCOIH}\mu^+(\triangledown, \sqcap)$-clauses

**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 $\mathcal{ALCOIH}\mu^+(\triangledown, \sqcap)$-concept that **(Case I)** does not contain any occurrence of $A$, or **(Case 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 and Purify rules, given in Figure 2, are the *forgetting rules* that lead to the elimination of concept symbols in $\mathcal{ALCOIH}\mu^+(\triangledown, \sqcap)$-clauses. Suppose $A \in \mathsf{N_C}$ is the pivot and $C$ is a concept expression, then $\mathcal{N}_C^A$ denotes the set obtained from $\mathcal{N}$ by replacing every occurrence of $A$ by $C$. We refer to the clauses of the form $C_i \sqcup A$ $(1 \leq i \leq n)$ as *positive premises* of the rule. The Ackermann rule is applicable (to forget $A$ in $\mathcal{N}$) only if $\mathcal{N}$ is in $A$-reduced form.

**Theorem 3 (Ackermann Lemma for Concept Forgetting).** *Let $\mathcal{I}$ be an arbitrary $\mathcal{ALCOIH}\mu^+(\triangledown, \sqcap)$-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$^{\mathcal{C}}$ rule and the Purify$^{\mathcal{C},p(n)}$ rules.*

This states that eliminating the pivot symbol in $\mathsf{N_C}$ with the Ackermann$^{\mathcal{C}}$ and Purify$^{\mathcal{C}}$ rules preserves equivalence up to the pivot. Given the pivot $A$ and a set $\mathcal{N}$ of clauses in $A$-reduced form, the Non-Cyclic Ackermann rule is applied when $A$ does not occur in the $C_i$ $(1 \leq i \leq n)$ of the positive premises. The Cyclic Ackermann rule is applied when $A$ occurs in the $C_i$ but only negatively, e.g., $A$ occurs in a cyclic clause $\neg\forall r.A \sqcup A$. Fixpoints are introduced in this case in order to facilitate finite representation of the result, 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 rule is applied whenever $A$ is *pure* in $\mathcal{N}$, i.e., $\mathcal{N}$ is positive (or negative) w.r.t. $A$.

**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: $A$ has positive occurrences in $\neg(D_1 \sqcup \ldots \sqcup D_n)$.

**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 $C_1 \sqcup \ldots \sqcup C_n$.

**Skolemization$^{\triangledown}$**

$$\frac{\mathcal{N}, C \sqcup \neg\forall\triangledown.D}{\mathcal{N}, \neg b \sqcup \neg D \sqcup \forall\triangledown.C}$$

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

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

**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 $\forall R.D$.

**Sign Switching**

$$\frac{\mathcal{N}}{(\mathcal{N}^A_{\neg A})^{\neg\neg A}_A}$$

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

**Fig. 3.** The rewrite rules for finding $A$-reduced form of $\mathcal{N}$

Clauses are usually not given in reduced form initially. Figure 3 lists the set of rewrite rules for finding the $A$-reduced form of a set of clauses for $A$ the pivot. The Case Splitting rule splits the derivation into several branches. The intermediate forgetting solution returned at the end of a symbol elimination round is the disjunction of the solutions of each branch in the derivation. New compared to [29], besides the Cyclic Ackermann rule, is the Skolemization$^{\triangledown}$ rule

that rewrites the existential quantification in the premise by the introduction of a fresh nominal. Crucial for the practicality of the method are a number of equivalence-preserving simplification rules, not described here though.

## 5 Calculus for Role Forgetting

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

**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 $\mathcal{ALCOIH}\mu^+(\triangledown, \sqcap)$-concepts that do not contain any occurrence of $r$ and $Q$ is an $\mathcal{ALCOIH}\mu^+(\triangledown, \sqcap)$ role expression that does not contain any occurrence of $r$; or it has the form $\neg S \sqcup r$ or $S \sqcup \neg r$, where $S \in \{s, s^-\}$ for $s$ ($\neq r$) 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. Finding pivot-reduced form of a clause is not always possible unless definer symbols are introduced. *Definer symbols* are specialised concept symbols that do not occur in the present ontology [13], 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 $C$ and $D$ do not contain any occurrences of $r$. A new clause $\neg\mathcal{D} \sqcup \mathcal{Q}$ is added to the clause set for each replaced sub-concept $\mathcal{Q}$, where $\mathcal{D}$ is a fresh definer symbol. For example, introducing a definer symbol $D_1$ leads to $A \sqcup \forall r.\neg\forall r.B$ being rewritten with $A \sqcup \forall r.D_1$ and $\neg D_1 \sqcup \neg\forall r.B$, where $A$ and $B$ are concept symbols. The definer symbols are eliminated using the rules for concept forgetting once all role symbols in $\Sigma$ have been forgotten. It is for this reason that our system defaults to forgetting role symbols first so that the definer symbols can be eliminated as part of subsequent concept forgetting.

| **Role Surfacing to TBox** | **Role Surfacing to RBox** |
|---|---|
| $\dfrac{\mathcal{N}, C \sqcup \forall r^-.D}{\mathcal{N}, D \sqcup \forall r.C}$ | $\dfrac{\mathcal{N}, \neg S \sqcup r^-}{\mathcal{N}, \neg S^- \sqcup r}$ $\quad$ $\dfrac{\mathcal{N}, S \sqcup \neg r^-}{\mathcal{N}, S^- \sqcup \neg r}$ |
| provided: $r$ is a role symbol that does not occur in $C$ and $D$. | provided: $S$ is either a role symbol or an inverted role symbol. |

**Fig. 4.** The rewrite rules for finding $r$-reduced form of $\mathcal{N}$

Since the underlying language accommodates role inverse, the calculus includes two Role Surfacing rules, shown in Figure 4, in order to reformulate expressions without occurrences of inverses of the pivot in the respective TBox and RBox clauses, so that the other rules in the calculus do not need to cater for role inverse. The Role Surfacing rules are applied after definer symbols have been introduced, and before the application of the forgetting rules.

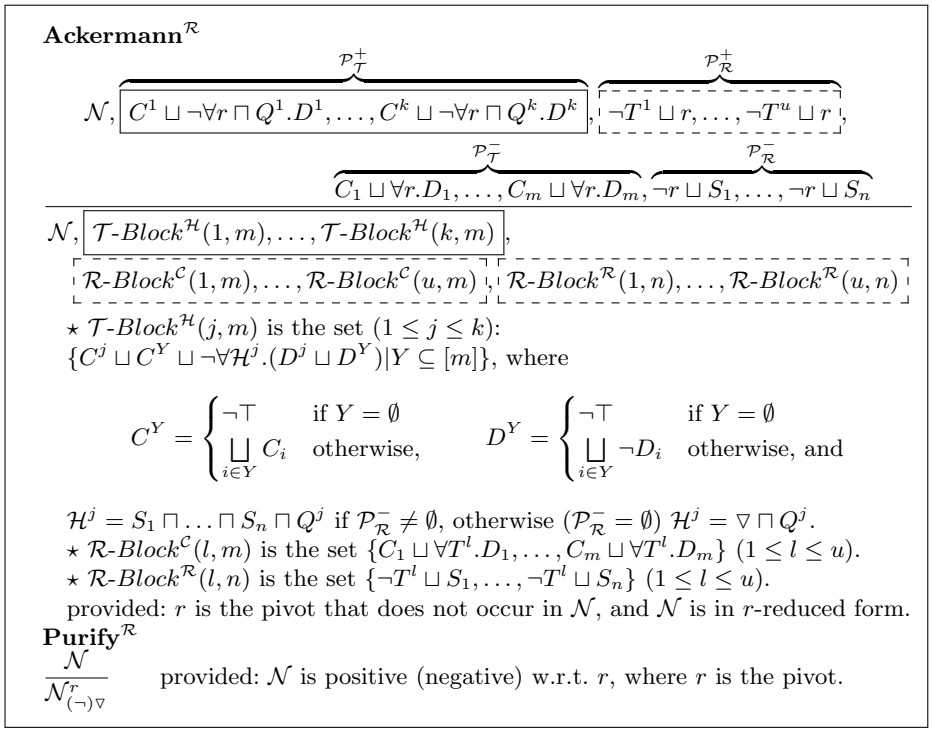


**Fig. 5.** Forgetting role symbol $r$ in a set $\mathcal{N}$ of $\mathcal{ALCOIH}\mu^+(\triangledown, \sqcap)$-clauses

**Theorem 4 (Ackermann Lemma for Role Forgetting).** *Let $\mathcal{I}$ be an arbitrary $\mathcal{ALCOIH}\mu^+(\triangledown, \sqcap)$-interpretation. For $r$ the pivot, when the Ackermann$^{\mathcal{R}}$ or Purify$^{\mathcal{R}}$ rule in Figure 5 is applicable then 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 and the Purify$^{\mathcal{R}}$ rule given in Figure 5 to eliminate $r$. 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 $\neg T^l \sqcup r$ $(1 \leq l \leq n)$ as *positive RBox premises* (denoted by $\mathcal{P}^+_{\mathcal{R}}$) and clauses of the form $\neg r \sqcup S_i$ $(1 \leq i \leq u)$ 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 wholly 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 $\mathcal{O}'$ where $r \notin \text{sig}(\mathcal{O}')$. Specifically, combining $\mathcal{P}^-_{\mathcal{T}}$ and $\mathcal{P}^-_{\mathcal{R}}$ with one of the positive

TBox premises ($\mathcal{P}^+_\mathcal{T}$) leads to a set of TBox clauses (denoted by $\mathcal{T}$-$Block^\mathcal{H}(j,m)$), where $\mathcal{H}^j$ is the conjunction of $Q^j$ and the $S_i$ ($1 \leq i \leq n$) occurring in $\mathcal{P}^-_\mathcal{R}$ ($\mathcal{H}^j := Q^j \sqcap S_1 \sqcap \ldots \sqcap S_n$), $m$ 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. 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 $A \sqcup \neg\forall r.B \in \mathcal{P}^+_\mathcal{T}$, combining $\mathcal{P}^-_\mathcal{T}$ and $\mathcal{P}^-_\mathcal{R}$ with $A \sqcup \neg\forall r.B$ leads to $\mathcal{T}$-$Block^\mathcal{H}(j,m) = \{A \sqcup C^Y \sqcup \neg\forall(s_1 \sqcap s_2^-).(B \sqcup D^Y) | Y \subseteq [2]\}$ that consists of the following clauses:

1. $A \sqcup \underbrace{\neg\top}_{C^Y} \sqcup \neg\forall(s_1 \sqcap s_2^-).(B \sqcup \underbrace{\neg\top}_{D^Y})$ when $Y = \emptyset$
2. $A \sqcup \underbrace{A_1}_{C^Y} \sqcup \neg\forall(s_1 \sqcap s_2^-).(B \sqcup \underbrace{\neg B_1}_{D^Y})$ when $Y = \{1\}$
3. $A \sqcup \underbrace{A_2}_{C^Y} \sqcup \neg\forall(s_1 \sqcap s_2^-).(B \sqcup \underbrace{\neg B_2}_{D^Y})$ when $Y = \{2\}$
4. $A \sqcup \underbrace{A_1 \sqcup A_2}_{C^Y} \sqcup \neg\forall(s_1 \sqcap s_2^-).(B \sqcup \underbrace{\neg B_1 \sqcup \neg B_2}_{D^Y})$ when $Y = \{1,2\}$

Combining $\mathcal{P}^-_\mathcal{T}$ with one of the positive RBox premises leads to a set of TBox clauses (denoted by $\mathcal{R}$-$Block^\mathcal{C}(l,m)$), where $m$ are 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 leads to a set of RBox clauses (denoted by $\mathcal{R}$-$Block^\mathcal{R}(l,n)$), where $n$ are 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 leads to the same result as in Case (I) ($\mathcal{T}$-$Block^\mathcal{H}(j,m)$), only that a top role ($\triangledown$) replaces $S_1 \sqcap \ldots \sqcap S_n$, i.e., $\mathcal{H} = \triangledown$. Combining $\mathcal{P}^-_\mathcal{T}$ with one of the positive RBox premises leads to the same result as in Case (I), i.e., $\mathcal{R}$-$Block^\mathcal{C}(l,m)$ ($1 \leq l \leq u$). **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 leads to the same results as in Case (I), i.e., $\mathcal{T}$-$Block^\mathcal{H}(j,m)$ ($1 \leq j \leq k$) and $\mathcal{R}$-$Block^\mathcal{R}(l,n)$ ($1 \leq l \leq u$) 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. The pivot is eliminated using the Purify$^\mathcal{R}$ rule in this case.

The pivot is forgotten in the ontology once every positive premise (in $\mathcal{P}^+_\mathcal{T}$ and $\mathcal{P}^+_\mathcal{R}$) has 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 with $m$ 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 a set of $k \cdot 2^m$ clauses (exponential growth); combining $\mathcal{P}^-_\mathcal{T}$ and $\mathcal{P}^-_\mathcal{R}$ with all positive RBox premises yields a set of $um + un$ clauses (polynomial growth). The size of the forgetting solution therefore depends largely on the number of the negative TBox premises ($m$).

## 6 Evaluation and Empirical Results

We implemented a prototype of the forgetting method in Java using the OWL-API[1], and conducted a number of experiments on real-world ontologies that are taken from the NCBO BioPortal[2] and Oxford Ontology[3] repositories 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 restricted to $\mathcal{ALCOIH}(\triangledown, \sqcap)$-fragments, and any sub-concepts beyond the scope of $\mathcal{ALCOIH}(\triangledown, \sqcap)$ were replaced by $\top$. Consequently, 180 and 200 ontologies of various sizes were selected from the NCBO BioPortal and Oxford Ontology Library, 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 that were randomly selected 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 [12–15]. We therefore set up a series of experiments with a large number of concept and role symbols to be forgotten (80% of the concept symbols and 50% of the role symbols in the initial signature). The heuristic for determining the order of eliminating concept and roles symbols ($\succ_{\mathcal{C}}$ and $\succ_{\mathcal{R}}$) was also tested. The $\Sigma$-symbols were eliminated in the order as returned by the OWL-API function that gets all concept symbols in the ontology, when the heuristic was not applied. We started with the evaluation of concept symbol forgetting, where a timeout of 15 minutes was imposed.

| Input | Setting | | Results | | | | Setting | | Results | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ontology | $\lvert\Sigma\rvert$ (%) | $\succ_{\mathcal{C}}$ | Time (sec.) | Timeout | Success R. | Fixp. | $\lvert\Sigma\rvert$ (%) | $\succ_{\mathcal{R}}$ | Definer in | Time (sec.) | Success R. | Clause ↑ |
| **BioPortal** (354 Axioms on Avg.) | 20 (10%) | ✗ | 4.890 | 0.0% | 100.0% | 0.0% | 4 (10%) | ✗ | 1.1/onto. | 2.120 sec. | 100.0% | 4.1% |
| | | ✓ | 3.260 | 0.0% | 100.0% | 0.0% | | ✓ | 10 onto. | 2.101 sec. | 100.0% | 4.1% |
| | 60 (30%) | ✗ | 18.672 | 4.4% | 94.4% | 7.2% | 12 (30%) | ✗ | 1.9/onto. | 8.658 sec. | 100.0% | 14.5% |
| | | ✓ | 9.336 | 1.1% | 98.3% | 7.8% | | ✓ | 13 onto. | 8.314 sec. | 100.0% | 14.5% |
| | 160 (80%) | ✗ | 70.416 | 13.8% | 83.3% | 13.3% | 20 (50%) | ✗ | 3.0/onto. | 20.913 sec. | 100.0% | 26.5% |
| | | ✓ | 29.340 | 5.6% | 91.7% | 17.2% | | ✓ | 16 onto. | 20.566 sec. | 100.0% | 26.5% |
| | **80 Avg.** | ✗ | **31.326** | **6.3%** | **92.6%** | **6.8%** | **Avg.** | ✗ | **2.0/onto.** | **10.564 sec.** | **100.0%** | **15.0%** |
| | | ✓ | **13.979** | **2.4%** | **96.7%** | **8.3%** | | ✓ | **13 onto.** | **10.327 sec.** | **100.0%** | **15.0%** |
| **Oxford** (875 Axioms on Avg.) | 36 (10%) | ✗ | 44.392 | 3.0% | 97.0% | 0.5% | 5 (10%) | ✗ | 2.4/onto. | 3.187 sec. | 100.0% | 2.2% |
| | | ✓ | 27.745 | 1.5% | 98.5% | 0.5% | | ✓ | 25 onto. | 3.072 sec. | 100.0% | 2.2% |
| | 108 (30%) | ✗ | 193.106 | 17.0% | 79.5% | 11.5% | 15 (30%) | ✗ | 3.7/onto. | 18.537 sec. | 100.0% | 6.9% |
| | | ✓ | 80.461 | 9.5% | 88.5% | 14.5% | | ✓ | 28 onto. | 17.998 sec. | 100.0% | 6.9% |
| | 288 (80%) | ✗ | 412.852 | 34.5% | 61.5% | 19.0% | 25 (50%) | ✗ | 5.7/onto. | 36.292 sec. | 100.0% | 14.6% |
| | | ✓ | 166.270 | 17.5% | 78.5% | 26.5% | | ✓ | 39 onto. | 34.117 sec. | 100.0% | 14.6% |
| | **144 Avg.** | ✗ | **216.783** | **18.2%** | **79.3%** | **10.3%** | **Avg.** | ✗ | **3.9/onto.** | **19.339 sec.** | **100.0%** | **7.9%** |
| | | ✓ | **91.492** | **9.5%** | **88.5%** | **13.8%** | | ✓ | **30 onto.** | **18.396 sec.** | **100.0%** | **7.9%** |

**Fig. 6.** Results for concept forgetting

**Fig. 7.** Results for role forgetting

The results (of forgetting only concept symbols) obtained from forgetting 10%, 30% and 80% of the concept symbols from the respective BioPortal and Oxford ontologies are shown in Figure 6, which is quite revealing in several ways. The most notable observation to emerge from the results is that, with the

[1] http://owlapi.sourceforge.net/
[2] http://bioportal.bioontology.org/
[3] http://www.cs.ox.ac.uk/isg/ontologies/

heuristically determined elimination order (indicated by ✓), our implementation was successful (forgot all $\Sigma$-symbols) in 96.7% of the BioPortal-cases within a limited period of time, with 8.3% of them using fixpoints in the result. In the case of 10% of the concept symbols specified to be forgotten, the success rate rose to 100%. Even without the heuristic (✗), the forgetting solution could be found by our implementation 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 cases 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.

We also evaluated the performance of forgetting different number of role symbols with the same ontologies used for the evaluation of concept forgetting (using a timeout of 5 minutes). The results (of forgetting only role symbols) are shown in Figure 7, from which it can be seen that our method successfully eliminated the role symbols in $\Sigma$ in all cases. The time used for forgetting role 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 symbol forgetting leads to growth of clauses in the forgetting solution, which was however modest (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 that there were $m$ definer symbols introduced in each ontology, and by $n$ *onto* we mean that $n$ ontologies out of the total introduced the definer symbol.

## 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 has 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 the interaction between transitive roles and role hierarchy inclusion statements can lead to results where it is not clear how to represent them finitely. The problem of extending the method to handle number restrictions remains completely open; possibly the techniques of [14, 15] can help extend the method.

## References


1. W. Ackermann. Untersuchungen über das Eliminationsproblem der mathematischen Logik. *Mathematische Annalen*, 110(1):390–413, 1935.
2. D. Calvanese, G. De Giacomo, and M. Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *Proc. IJCAI'99*, pages 84–89. Morgan Kaufmann, 1999.
3. 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.
4. G. D'Agostino and M. Hollenberg. Logical questions concerning the $\mu$-Calculus: Interpolation, Lyndon and Los-Tarski. *J. Symb. Log.*, 65(1):310–332, 2000.
5. P. Doherty, W. Łukaszewicz, and A. Szałas. Computing circumscription revisited: A reduction algorithm. *Journal of Automated Reasoning*, 18(3):297–336, 1997.
6. D. M. Gabbay and H. J. Ohlbach. Quantifier elimination in second–order predicate logic. In *Proc. KR'92*, pages 425–435. Morgan Kaufmann, 1992.
7. D. M. Gabbay, R. A. Schmidt, and A. Szałas. *Second Order Quantifier Elimination: Foundations, Computational Aspects and Applications*. College Publications, 2008.
8. E. Grädel and I. Walukiewicz. Guarded fixed point logic. In *Proc. LICS'99*, pages 45–54. IEEE Computer Society, 1999.
9. A. Herzig and J. Mengin. Uniform interpolation by resolution in modal logic. In *Proc. JELIA'08*, volume 5293 of *Lecture Notes in Computer Science*, pages 219–231. Springer, 2008.
10. B. Konev, C. Lutz, D. Walther, and F. Wolter. Model-theoretic inseparability and modularity of description logic ontologies. *Artificial Intelligence*, 203:66–103, 2013.
11. B. Konev, D. Walther, and F. Wolter. Forgetting and uniform interpolation in extensions of the description logic $\mathcal{EL}$. In *Proc. DL'09*, volume 477 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.
12. P. Koopmann and R. A. Schmidt. Forgetting concept and role symbols in $\mathcal{ALCH}$-ontologies. In *Proc. LPAR'13*, volume 8312 of *Lecture Notes in Computer Science*, pages 552–567. Springer, 2013.
13. P. Koopmann and R. A. Schmidt. Uniform interpolation of $\mathcal{ALC}$-ontologies using fixpoints. In *Proc. FroCos'13*, volume 8152 of *Lecture Notes in Computer Science*, pages 87–102. Springer, 2013.
14. P. Koopmann and R. A. Schmidt. Count and forget: Uniform interpolation of $\mathcal{SHQ}$-ontologies. In *Proc. IJCAR'14*, volume 8562 of *Lecture Notes in Computer Science*, pages 434–448. Springer, 2014.
15. P. Koopmann and R. A. Schmidt. Saturated-based forgetting in the description logic $\mathcal{SIF}$. In *Proc. DL'15*, volume 1350 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.
16. P. Koopmann and R. A. Schmidt. Uniform interpolation and forgetting for $\mathcal{ALC}$-ontologies with ABoxes. In *Proc. AAAI'15*, pages 175–181. AAAI Press, 2015.
17. F. Lin and R. Reiter. Forget it! In *Proc. AAAI Fall Symposium on Relevance*, pages 154–159, 1994.
18. 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.
19. 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*, pages 286–297. AAAI Press, 2012.

20. C. Lutz and F. Wolter. Foundations for uniform interpolation and forgetting in expressive description logics. In *Proc. IJCAI'11*, pages 989–995. IJCAI/AAAI, 2011.
21. 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.
22. R. A. Schmidt. The Ackermann approach for modal logic, correspondence theory and second-order reduction. *Journal of Applied Logic*, 10(1):52–74, 2012.
23. R. A. Schmidt and D. Tishkovsky. Using tableau to decide description logics with full role negation and identity. *ACM Trans. Comput. Log.*, 15(1):7:1–7:31, 2014.
24. A. Visser. *Bisimulations, Model Descriptions and Propositional Quantifiers*. Logic Group Preprint Series. Department of Philosophy, Utrecht Univ., 1996.
25. K. Wang, Z. Wang, R. Topor, J. Z. Pan, and G. Antoniou. Concept and role forgetting in $\mathcal{ALC}$ ontologies. In *Proc. ISWC'09*, volume 5823 of *Lecture Notes in Computer Science*, pages 666–681. Springer, 2009.
26. 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.
27. Z. Wang, K. Wang, R. Topor, and J. Z. Pan. Forgetting for knowledge bases in DL-Lite. *Annals of Mathematics and Artificial Intelligence*, 58(1-2):117–151, 2010.
28. Z. Wang, K. Wang, R. W. Topor, and J. Z. Pan. Forgetting concepts in DL-Lite. In *Proc. ESWC'08*, volume 5021 of *Lecture Notes in Computer Science*, pages 245–257. Springer, 2008.
29. Y. Zhao and R. A. Schmidt. Concept Forgetting in $\mathcal{ALCOI}$-Ontologies Using an Ackermann Approach. In *Proc. ISWC'15*, volume 9366 of *Lecture Notes in Computer Science*, pages 587–602. Springer, 2015.