On Concept Forgetting in Description Logics with Qualified Number Restrictions

Yizheng Zhao and Renate A. Schmidt
School of Computer Science, The University of Manchester, UK

Abstract

This paper presents a practical method for computing solutions of concept forgetting in the description logic $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$, basic $\mathcal{ALC}$ extended with nominals, qualified number restrictions, role negation, role conjunction and role disjunction. The method is based on a non-trivial generalisation of Ackermann's Lemma, and attempts to compute either semantic solutions of concept forgetting or uniform interpolants in $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$. It is so far the only approach to concept forgetting in description logics with number restrictions plus nominals, as well as in description logics with ABoxes. Results of an evaluation with a prototypical implementation have shown that the method was successful in more than 90% of the test cases from a large corpus of biomedical ontologies. In only 13.2% of these cases the solutions were semantic solutions.

1 Introduction

Forgetting is a non-standard reasoning service that seeks to create restricted views of ontologies by eliminating concept and role names from ontologies in a way such that all logical consequences are preserved up to the remaining signature. It has proved to be a very useful technique in ontology-based knowledge processing, as it allows users to focus on specific parts of (usually very large) ontologies for easy reuse, or to zoom in on (usually very complex) ontologies for in-depth analysis. Other uses of forgetting are information hiding, explanation generation (abduction), ontology debugging and repair, as well as computing the logical difference between ontology versions [Bicarregui *et al.*, 2001; Lang *et al.*, 2003; Konev *et al.*, 2009; Grau and Motik, 2012; Wernhard, 2013; Ludwig and Konev, 2014; Wang *et al.*, 2014].

Forgetting can be defined in two ways that are closely related; it can be defined syntactically as the dual of *uniform interpolation* [Visser, 1996] (related notions include weak forgetting [Zhang and Zhou, 2010], consequence-based inseparability [Lutz and Wolter, 2010] and consequence-based conservative extensions [Ghilardi *et al.*, 2006]) and it can be defined model-theoretically as *semantic forgetting* [Wang *et al.*, 2014] (related notions include strong forgetting [Lin and Reiter, 1994], model inseparability [Konev *et al.*, 2013], model conservative extensions [Lutz *et al.*, 2007] and second-order quantifier elimination [Gabbay *et al.*, 2008]). The two notions differ in the sense that uniform interpolation preserves all *logical consequences* up to certain names whereas semantic forgetting preserves *equivalence* up to certain names. In this sense, semantic solutions are in general stronger than the uniform interpolants; they often require more expressivity than is available in the source logic. For example, the semantic solution of forgetting the role name $\{r\}$ from the $\mathcal{ALC}$-ontology $\{A_1 \sqsubseteq \exists r.B, A_2 \sqsubseteq \forall r.\neg B\}$ is $\{A_1 \sqsubseteq \exists \nabla.B, A_1 \sqcap A_2 \sqsubseteq \bot\}$, whereas the uniform interpolant is $\{A_1 \sqcap A_2 \sqsubseteq \bot\}$, which is weaker. Observe in this case that the target language must include the universal role ∇ to represent the semantic solution. If a semantic solution is expressible in the source logic, then it is equivalent to the uniform interpolant, which means, in this case, the two notions coincide [Zhang and Zhou, 2010].

Despite the notable usefulness in ontology engineering as described above, forgetting is an inherently difficult problem; it is much harder than standard reasoning (satisfiability testing) and very few logics are known to have the uniform interpolation property or are complete for semantic forgetting. It is known that: (i) uniform interpolants and semantic solutions of forgetting do not always exist for $\mathcal{EL}$ and $\mathcal{ALC}$ [Konev *et al.*, 2008; Lutz and Wolter, 2011; Konev *et al.*, 2013], (ii) deciding the existence of uniform interpolants is EXPTIME-complete for $\mathcal{EL}$ [Lutz *et al.*, 2012] and 2EXPTIME-complete for $\mathcal{ALC}$ [Lutz and Wolter, 2011], (iii) the existence of semantic solutions of forgetting is undecidable for $\mathcal{EL}$ and $\mathcal{ALC}$ [Konev *et al.*, 2013; Botoeva *et al.*, 2016], and (iv) uniform interpolants can be triple exponential in size w.r.t. the input ontologies for $\mathcal{EL}$ and $\mathcal{ALC}$ [Lutz and Wolter, 2011; Nikitina and Rudolph, 2014].

These not very encouraging results have not prevented the community from developing practical methods for computing uniform interpolants and semantic forgetting solutions for various description logics. For example, [Ludwig and Konev, 2014] have developed a resolution-based method for computing uniform interpolants for $\mathcal{ALC}$ TBoxes. Since $\mathcal{ALC}$ does not have the uniform interpolation property, the method introduces a depth-bounded version of the core algorithm to guarantee finite representations of uniform interpolants. Another resolution-based method for uniform interpolation for $\mathcal{ALC}$ TBoxes is by [Koopmann and Schmidt, 2013b]. This method introduces fixpoint operators in the target language to ensure that uniform interpolants can always be finitely represented. The method has been extended to the languages of $\mathcal{ALCH}$, $\mathcal{ALCQ}$ and $\mathcal{SIF}$ TBoxes [Koopmann and Schmidt, 2013a; 2014; 2015a], and $\mathcal{ALC}$ and $\mathcal{SHI}$ with ABoxes [Koopmann and Schmidt, 2015b; Koopmann, 2015]. Practical methods for computing semantic solutions of forgetting are developed, implemented and evaluated in [Zhao and Schmidt, 2015; 2016]. These methods are based on generalisations of Ackermann's Lemma [Ackermann, 1935], and attempt to eliminate concept and role names from ontologies expressible in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$. The methods are incomplete however; for example, they cannot handle the case of forgetting the concept name $\{B\}$ from $\{A_1 \sqsubseteq \exists r.B, A_2 \sqsubseteq \exists r.\neg B\}$, because the forgetting solution $\{A_1 \sqsubseteq \exists r.\top, A_2 \sqsubseteq \exists r.\top, A_1 \sqcap A_2 \sqsubseteq {\geq}2r.\top\}$ requires more expressivity than is available in the source logic (number restrictions). The methods have been extended to description logics with number restrictions [Zhao and Schmidt, 2017], but the work is focused only on role forgetting.

This paper presents a practical method for computing solutions of concept forgetting in description logics with number restrictions and nominals. While allowing the given example to be solved, admitting number restrictions and nominals significantly increases the difficulty of the problem. Our method handles in particular ontologies expressed in $\mathcal{ALCOQ}$ and the extension with role negation, role conjunction and role disjunction, which means that it can handle expressive description logics that cannot be handled by other methods at present. The method follows an Ackermann-based approach, it is terminating, sound and nearly concept forgetting complete for $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$. When it succeeds, the method outputs either a semantic solution of concept forgetting, or a uniform interpolant in $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$. Results of an evaluation with a prototypical implementation have shown that the method is computationally feasible and is able to find solutions of concept forgetting in more than 90% of the test cases from a large corpus of biomedical ontologies. In only 13.2% of these cases the solutions were semantic solutions.

2 The Description Logic $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$

Let $\mathsf{N_C}$, $\mathsf{N_R}$ and $\mathsf{N_I}$ be countably infinite and pairwise disjoint sets of *concept names*, *role names* and *individual names* (aka *nominals*), respectively. *Roles* in $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$ can be a role name $r \in \mathsf{N_R}$, or formed with negation $\neg$, conjunction $\sqcap$ and disjunction $\sqcup$. *Concepts* in $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$ have one of the following forms:

$$\top \mid \bot \mid a \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid {\geq}mR.C \mid {\leq}nR.C,$$

where $a \in \mathsf{N_I}$, $A \in \mathsf{N_C}$, C and D are any concepts, R is any role, and $m \geq 1$ and $n \geq 0$ are natural numbers. Additional concepts are represented as abbreviations: $\exists R.C = {\geq}1R.C$, $\forall R.C = {\leq}0R.\neg C$, $\neg{\geq}mR.C = {\leq}nR.C$ and $\neg{\leq}nR.C = {\geq}mR.C$, where $n = m - 1$. Concepts of the form ${\geq}mR.C$ and ${\leq}nR.C$ are called *(qualified) number restrictions*, which allow one to specify cardinality constraints on roles.

An $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$-ontology comprises of a TBox and an ABox. A TBox is a finite set of axioms of the form $C \sqsubseteq D$ (*concept inclusions*), where C and D are concepts. An ABox is a finite set of axioms of the form $C(a)$ (*concept assertions*), where $a \in \mathsf{N_I}$ and C is a concept. As this paper is only concerned with concept forgetting, not including role assertions or role inclusions in the language is without loss of generality, because concept names do not occur in them. Concept assertions are superfluous in description logics with nominals, because they can be expressed equivalently as concept inclusions via nominals: $C(a)$ as $a \sqsubseteq C$. Hence, in this paper, an $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$-ontology is assumed to contain only TBox axioms. The semantics of $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$ is as expected.

Theorem 1. *Reasoning in $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$ is decidable.*

Theorem 1 follows from the decidability of C^2 [Grädel *et al.*, 1997], which subsumes the logic $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$.

Our forgetting method works with TBox axioms in clausal form. A *TBox literal* in $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$ is a concept of the form a, $\neg a$, A, $\neg A$, ${\geq}mR.C$ or ${\leq}nR.C$. A *TBox clause* in $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$ is a disjunction of a finite number of TBox literals. TBox clauses are obtained from TBox axioms using the standard clausal form transformations.

Let $A \in \mathsf{N_C}$ be a designated concept name. An occurrence of A is said to be *positive* (*negative*) in an axiom (clause) if it is under an *even* (*odd*) number of explicit and implicit negations, e.g., A is positive in ${\geq}mr.A$ and ${\leq}nr.\neg A$, and negative in ${\geq}mr.\neg A$ and ${\leq}nr.A$. An axiom or clause is called an *A-axiom* or *A-clause* if it contains an occurrence of A. An axiom or clause is called an *A^+-axiom* or *A^+-clause* (*A^--axiom* or *A^--clause*) if it contains a positive (negative) occurrence of A. A set $\mathcal{N}$ of clauses is said to be *positive* (*negative*) w.r.t. A if every occurrence of A in $\mathcal{N}$ is positive (negative).

By $\mathsf{sig_C}(X)$, $\mathsf{sig_R}(X)$ and $\mathsf{sig_I}(X)$, we denote respectively sets of the concept names, role names and individual names occurring in X, where X ranges over concepts, roles, axioms, clauses, sets of axioms and sets of clauses. Let $A \in \mathsf{N_C}$ be any concept name, and let $\mathcal{I}$ and $\mathcal{I}'$ be two interpretations. We say that $\mathcal{I}$ and $\mathcal{I}'$ are *equivalent up to A*, or *A-equivalent*, if $\mathcal{I}$ and $\mathcal{I}'$ coincide but differ possibly in the interpretation of A. More generally, $\mathcal{I}$ and $\mathcal{I}'$ are *equivalent up to a set $\mathcal{F}$ of concept names*, or *$\mathcal{F}$-equivalent*, if $\mathcal{I}$ and $\mathcal{I}'$ coincide but differ possibly in the interpretations of the names in $\mathcal{F}$.

Definition 1 (**Semantic Forgetting**). *Let $\mathcal{O}$ be an ontology and let $\mathcal{F} \subseteq \mathsf{sig_C}(\mathcal{O})$ be a forgetting signature. An ontology $\mathcal{O}'$ is a* semantic solution *of forgetting $\mathcal{F}$ from $\mathcal{O}$ iff the following conditions hold: (i) $\mathsf{sig_C}(\mathcal{O}') \subseteq \mathsf{sig_C}(\mathcal{O}) \backslash \mathcal{F}$, and (ii) for any interpretation $\mathcal{I}$: $\mathcal{I} \models \mathcal{O}'$ iff $\mathcal{I}' \models \mathcal{O}$, for some interpretation $\mathcal{I}'$ $\mathcal{F}$-equivalent to $\mathcal{I}$.*

Definition 2 (**Uniform Interpolation**). *Let $\mathcal{O}$ be an ontology and let $\mathcal{F} \subseteq \mathsf{sig_C}(\mathcal{O})$ be a forgetting signature. An ontology $\mathcal{O}'$ is a* uniform interpolant *of $\mathcal{O}$ for the signature $\mathsf{sig_C}(\mathcal{O}) \backslash \mathcal{F}$ iff the following conditions hold: (i) $\mathsf{sig_C}(\mathcal{O}') \subseteq \mathsf{sig_C}(\mathcal{O}) \backslash \mathcal{F}$, and (ii) for any axiom α with $\mathsf{sig}(\alpha) \subseteq \mathsf{sig}(\mathcal{O}) \backslash \mathcal{F}$, $\mathcal{O}' \models \alpha$ iff $\mathcal{O} \models \alpha$, i.e., $\mathcal{O}'$ preserves all logical consequences up to the names in $\mathsf{sig_C}(\mathcal{O}) \backslash \mathcal{F}$.*

In this paper, we use the notation $\mathcal{F}$ to denote the forgetting signature, i.e., the set of concept names to be forgotten. The concept name in $\mathcal{F}$ under current consideration for forgetting is referred to as the *pivot* in our method.

3 Major Obstacles to Semantic Concept Forgetting in Description Logics with Qualified Number Restrictions

Solutions of forgetting the names in $\mathcal{F}$ are computed by successively eliminating single names in $\mathcal{F}$. For most cases the elimination can be based on generalisations of *Ackermann's Lemma* [Ackermann, 1935], which allow one to eliminate a single concept or role name from an ontology in a way such that the original ontology and the resulting one are equivalent up to the interpretations of the eliminated name (i.e., Conditions (i) and (ii) of Definition 1 hold). The proof is an easy adaptation of the Ackermann's original result [Gabbay *et al.*, 2008]. In Ackermann-based approaches, e.g. [Szałas, 2006; Schmidt, 2012; Zhao and Schmidt, 2015; 2016; 2017], the lemma is the basis for the following rules:

$$\frac{\mathcal{O}(\mathcal{S}^-), \alpha_1 \sqsubseteq \mathcal{S}, ..., \alpha_n \sqsubseteq \mathcal{S} \quad \text{(premises)}}{\mathcal{O}(\mathcal{S}^-)^{\mathcal{S}}_{\alpha_1 \sqcup ... \sqcup \alpha_n} \quad \text{(conclusion)}} \qquad (1)$$

$$\frac{\mathcal{O}(\mathcal{S}^+), \mathcal{S} \sqsubseteq \alpha_1, ..., \mathcal{S} \sqsubseteq \alpha_n \quad \text{(premises)}}{\mathcal{O}(\mathcal{S}^+)^{\mathcal{S}}_{\alpha_1 \sqcap ... \sqcap \alpha_n} \quad \text{(conclusion)}} \qquad (2)$$

where $\mathcal{S} \in \mathsf{N_C}$ ($\mathcal{S} \in \mathsf{N_R}$) is the pivot, the α_i ($1 \leq i \leq n$) are concepts (roles) that do not contain $\mathcal{S}$, $\mathcal{O}(\mathcal{S}^+)$ and $\mathcal{O}(\mathcal{S}^-)$ denote respectively the ontology $\mathcal{O}$ being positive and negative w.r.t. $\mathcal{S}$, and $\mathcal{O}^{\mathcal{S}}_{\alpha}$ denotes the ontology obtained from $\mathcal{O}$ by substituting α for every occurrence of $\mathcal{S}$ in $\mathcal{O}$. $\mathcal{S}$ is thus eliminated from $\mathcal{O}$ (by substitution). In this case, α is referred to as the *definition* of $\mathcal{S}$ in $\mathcal{O}$.

Observe that the Rules (1) and (2) are applicable to an ontology $\mathcal{O}$ to eliminate a name $\mathcal{S}$, only if (i) every $\mathcal{S}^+$-axiom in $\mathcal{O}$ has the form $\alpha \sqsubseteq \mathcal{S}$, where $\mathcal{S} \notin \mathsf{sig}(\alpha)$, or (ii) every $\mathcal{S}^-$-axiom in $\mathcal{O}$ has the form $\mathcal{S} \sqsubseteq \alpha$, where $\mathcal{S} \notin \mathsf{sig}(\alpha)$. On the other hand, since concept names may occur under a role restriction or a sequence of role restrictions (in this paper, a number restriction or a sequence of number restrictions), e.g., $\alpha \sqsubseteq \forall r.\mathcal{S}$ and $\alpha \sqsubseteq \exists r.\exists s.\mathcal{S}$, there are cases where $\mathcal{O}$ is not in a form suitable for application of either of the two rules. In these cases, $\mathcal{O}$ needs to be reformulated so that those syntactically unsatisfactory $\mathcal{S}$-axioms in $\mathcal{O}$ are equivalently expressed in the form of $\alpha \sqsubseteq \mathcal{S}$ or $\mathcal{S} \sqsubseteq \alpha$.

In the approach of [Zhao and Schmidt, 2015] to semantic concept forgetting in the description logic $\mathcal{ALCOI}$, a concept name under a (or a sequence of) universal role restriction can be moved outwards using *Galois connections* between $\forall r$ and $\forall r^-$, e.g., $\alpha \sqsubseteq \forall r.\mathcal{S}$ is equivalently expressed as $\exists r^-.\alpha \sqsubseteq \mathcal{S}$, and in the case of concept assertions, *Skolemisation* allows a concept name under an (or a sequence of) existential role restriction to be moved outwards, e.g., $a \sqsubseteq \exists r.\mathcal{S}$ is replaced by $a \sqsubseteq \exists r.b$ and $b \sqsubseteq \mathcal{S}$, where a is a nominal and b is a fresh nominal. However, this approach has three drawbacks: (i) Galois connections require inverse roles in the language and Skolemisation adds new nominals in the forgetting solutions, (ii) Skolemisation can be used only in the cases where the axiom is a concept assertion, but not when the axiom is a concept inclusion, and most importantly, (iii) Galois connections and Skolemisation are not sufficient in axioms with number restrictions. This means that these two methods cannot be used for the logic considered in this paper.

[Zhao and Schmidt, 2017] have developed an Ackermann-based approach to semantic role forgetting in description logics with number restrictions (but have left the problem of concept forgetting open). For concept forgetting in these logics, a seemingly feasible solution is the reduction of concept forgetting to role forgetting. The idea is to first replace in the given ontology every occurrence of the concept name one wants to forget by the concept $\geq 1r.\top$, where r is a fresh role name, and then forget $\{r\}$ from the present ontology using the approach of [Zhao and Schmidt, 2017]. However, this approach has proved to be unfeasible in practice, because the solution of role forgetting (i.e., an ontology that does not contain r) computed by the method include newly introduced definer names,[1] which are supposed not to be present in the forgetting solutions (they are supposed to be eliminated from the intermediate solutions as regular concept names). Moreover, it can be observed that the axioms in resulting ontology often have the same syntactic patterns as those given in the original concept forgetting. This brings the problem back to the original concept forgetting. For example, forgetting the concept name $\{B\}$ from the ontology $\{A_1 \sqsubseteq \geq 2s.B, A_2 \sqsubseteq \leq 1s.B\}$ can be reduced to the problem of forgetting the role name $\{r\}$ from $\{A_1 \sqsubseteq \geq 2s.\geq 1r.\top, A_2 \sqsubseteq \leq 1s.\geq 1r.\top\}$, where r is a fresh role name. The solution of the role forgetting computed by the method is $\{A_1 \sqsubseteq \geq 2s.\mathsf{D}, A_2 \sqsubseteq \leq 1s.\mathsf{D}\}$, where $\mathsf{D} \in \mathsf{N_D}$ is a fresh concept definer name. Observe that the axioms in the resulting set and those in the given one have the same syntactic patterns, and the problem then becomes forgetting the definer $\{\mathsf{D}\}$ from $\{A_1 \sqsubseteq \geq 2s.\mathsf{D}, A_2 \sqsubseteq \leq 1s.\mathsf{D}\}$, which is basically the same problem of forgetting $\{B\}$ from $\{A_1 \sqsubseteq \geq 2s.B, A_2 \sqsubseteq \leq 1s.B\}$.

Inspired by the same work we consider solving the problem by working on translations of ontologies in first-order logic, where transforming a formula into a specific form seems to be much easier (because in first-order logic concept and role names are syntactically more 'easy-going'; concept names are not explicitly preceded by role restrictions). Then concept names can be eliminated using methods for semantic forgetting in first-order logic such as the SCAN algorithm (based on resolution) and the DLS algorithm (based on Ackermann's Lemma) [Gabbay and Ohlbach, 1992; Doherty *et al.*, 1997; Gabbay *et al.*, 2008]. The forgetting solution, which is a set of first-order formulas, is then translated into equivalent expressions in a description logic. However, this is unfeasible as well. The reasons can be illustrated with two examples.

Example 1. First, we consider the case of forgetting the role name $\{r\}$ from the ontology $\{A_1 \sqsubseteq \geq 2r.B, A_2 \sqsubseteq \leq 1r.B\}$. The translation of this ontology in first-order logic is:

$$\begin{aligned}\{&1.\ \forall x(A_1(x) \to r(x, f_1(x))),\ 2.\ \forall x(A_1(x) \to B(f_1(x))),\\ &3.\ \forall x(A_1(x) \to r(x, f_2(x))),\ 4.\ \forall x(A_1(x) \to B(f_2(x))),\\ &5.\ \forall x(A_1(x) \to f_1(x) \not\approx f_2(x)),\ 6.\ \forall x, y, z(A_2(x) \to \neg r(x, y)\\ &\quad \vee \neg B(y) \vee \neg r(x, z) \vee \neg B(z) \vee y = z)\},\end{aligned}$$

[1]*Definers names* (or *definers*) are auxiliary concept names that do not occur in the present ontology [Koopmann and Schmidt, 2013b]; they are used to facilitate the normalisation of a given ontology.

where $f_1(x)$ and $f_2(x)$ are Skolem terms. The forgetting solution computed by both the SCAN or DLS method in existential quantifier-free first-order logic is:

$$\begin{aligned}&\{1.\ \forall x(A_1(x) \to B(f_1(x))),\ 2.\ \forall x(A_1(x) \to B(f_2(x))),\\ &\ 3.\ \forall x(A_1(x) \to f_1(x) \not\approx f_2(x)),\\ &\ 4.\ \forall x(A_1(x) \land A_2(x) \to \neg B(f_1(x)) \lor \neg B(f_2(x)))\}.\end{aligned}$$

This can be expressed equivalently in a description logic as follows (if the target logic includes the universal role):

$$\{A_1 \sqsubseteq {\geq}2\triangledown.B, A_1 \sqcap A_2 \sqsubseteq \bot\}.$$

Example 2. Next, we consider the case of forgetting the concept name $\{B\}$ from the same ontology as given in Example 1. The solution computed by SCAN or DLS in existential quantifier-free first-order logic is:

$$\begin{aligned}&\{1.\ \forall x(A_1(x) \to r(x, f_1(x))),\ 2.\ \forall x(A_1(x) \to r(x, f_2(x))),\\ &\ 3.\ \forall x(A_1(x) \to f_1(x) \not\approx f_2(x)),\\ &\ 4.\ \forall x, \forall x'(A_1(x) \land A_2(x') \to \neg r(x', f_1(x)) \lor \neg r(x', f_2(x)))\},\end{aligned}$$

Because of the two variables x and x' not being unified in the last formula of the forgetting solution, it is not clear if this solution can be expressed equivalently in a description logic. It has been found that solutions of concept forgetting usually contain such ununified variables, which makes the translation to a description logic seem impossible, but this does not happen when forgetting role names. An important reason is that role names are binary relations and are guards, and resolving on these unifies variables in the premises. In contrast, concept names are unary relations, and resolving on these produces a resolvent with more than two variables. These are the cases in our approach (described in the next section) where the preservation of equivalence is open.

4 Our Ackermann-Based Approach to Eliminating One Concept Name

In this section, we present our approach to eliminating a concept name from a set of TBox clauses in $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$. It is a direct approach based on a non-trivial generalisation of Ackermann's Lemma. The approach has two ingredients: (i) transformation of the given clause set into reduced form, and (ii) an Ackermann rule. The *reduced form* is a specialised normal form that is suitable for application of the Ackermann rule. The *Ackermann rule* reflects the generalisation of Ackermann's Lemma and allows a concept name to be eliminated from a set of clauses in reduced form.

4.1 Transformation into Reduced Form

Definition 3 (Reduced Form). *Suppose $A \in \mathsf{N_C}$ is the pivot. A clause is in* reduced form *if it has the form $C \sqcup A$, $C \sqcup \neg A$, $C \sqcup {\geq}xR.A$, $C \sqcup {\geq}xR.\neg A$, $C \sqcup {\leq}yR.A$ or $C \sqcup {\leq}yR.\neg A$, where C is a clause that does not contain A, R is a role, and $x \geq 1$ and $y \geq 0$ are natural numbers. A set $\mathcal{N}$ of clauses is in* reduced form *if every A-clause in $\mathcal{N}$ is in reduced form.*

The reduced forms include all elementary forms of clauses in which a concept name could occur (i.e., a concept name could occur at the top level, or under a $\geq$-restriction or $\leq$-restriction of a clause). Clauses not in reduced form have the form $C \sqcup {\geq}xR.D$ or $C \sqcup {\leq}yR.D$, where (i) C is a clause that contains A, or (ii) $D \neq (\neg)A$ is a concept that contains A.

Given a set $\mathcal{N}$ of TBox clauses (not in reduced form) with $A \in \mathsf{sig_C}(\mathcal{N})$ being the pivot, the first step in our approach is to transform $\mathcal{N}$ into reduced form so that the Ackermann rule can be applied to $\mathcal{N}$ to eliminate A. This involves the introduction of *definers*. Let $\mathsf{N_D} \subset \mathsf{N_C}$ be a set of definers disjoint from $\mathsf{sig_C}(\mathcal{N})$. Definers are introduced, incrementally replacing 'C' and 'D' in every A-clause not in reduced form until (i) and (ii) above do not hold. A new clause $\neg\mathsf{D}_1 \sqcup C$ is added to $\mathcal{N}$ for each replaced subconcept C, a new clause $\neg\mathsf{D}_2 \sqcup D$ is added to $\mathcal{N}$ for each replaced subconcept D immediately under a $\geq$-restriction, and a new clause $\mathsf{D}_3 \sqcup D$ is added to $\mathcal{N}$ for each replaced subconcept D immediately under a $\leq$-restriction, where $\mathsf{D}_1, \mathsf{D}_2, \mathsf{D}_3 \in \mathsf{N_D}$ are fresh definers. In this way, $\mathcal{N}$ is transformed into reduced form.

Theorem 2. *Using the definer introduction, any set of clauses in $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$ can be transformed into reduced form. The transformation is polynomial and preserves equivalence up to the interpretations of the introduced definers.*

Proof (sketch). The definer introduction is basically the standard structural transformation (polynomial). It uses Ackermann's Lemma in the reverse direction to introduce definers.

4.2 Ackermann Rule

Let $\mathcal{N}$ be a set of TBox clauses in reduced form for the pivot $A \in \mathrm{sig}_\mathsf{C}(\mathcal{N})$. By $\mathcal{P}^+_\bigstar(A)$ and $\mathcal{P}^-_\bigstar(A)$ we denote the sets of the clauses of the form $C \sqcup A$ and $C \sqcup \neg A$, respectively. By $\mathcal{P}^+_\geq(A)$ and $\mathcal{P}^-_\geq(A)$ we denote the sets of the clauses of the form $C \sqcup {\geq}xR.A$ and $C \sqcup {\geq}xR.\neg A$, respectively. By $\mathcal{P}^+_\leq(A)$ and $\mathcal{P}^-_\leq(A)$ we denote the sets of the clauses of the form $C \sqcup {\leq}yR.\neg A$ and $C \sqcup {\leq}yR.A$, respectively. We use $\mathcal{P}^+(A)$ to denote the union of the sets $\mathcal{P}^+_\bigstar(A)$, $\mathcal{P}^+_\geq(A)$ and $\mathcal{P}^+_\leq(A)$ (the *positive premises*). We use $\mathcal{P}^-(A)$ to denote the union of the sets $\mathcal{P}^-_\bigstar(A)$, $\mathcal{P}^-_\geq(A)$ and $\mathcal{P}^-_\leq(A)$ (the *negative premises*). We use $\mathcal{N}^{-A}$ to denote the set of the non-A-clauses in $\mathcal{N}$.

The second step in our approach is to apply the Ackermann rule, shown in Figure 1, to $\mathcal{N}$ to eliminate A. The Ackermann rule is a replacement rule which replaces the clauses above the line (the *premises*) by those under the line (the *conclusion*). The idea is to combine the set of the positive premises $\mathcal{P}^+(A)$ with every negative premise $\alpha \in \mathcal{P}^-(A)$ (or to combine the set of the negative premises $\mathcal{P}^-(A)$ with every positive premise $\alpha \in \mathcal{P}^+(A)$). The result of each combination is a finite set of clauses that does not contain A, denoted by $\textbf{BLOCK}(\mathcal{P}^+(A), \alpha)$ (or $\textbf{BLOCK}(\mathcal{P}^-(A), \alpha)$). The conclusion of the rule, which is the solution of forgetting A from $\mathcal{N}$, is the union of the blocks obtained from these combinations. It is found that in some combination cases the result obtained from combining $\mathcal{P}^{+(-)}(A)$ with α is identical to the union of the results obtained from combining respectively $\mathcal{P}^{+(-)}_\bigstar(A)$, $\mathcal{P}^{+(-)}_\geq(A)$ and $\mathcal{P}^{+(-)}_\leq(A)$ with α. Thus, for clarity's sake, we split such combinations into separate cases in the presentation of the Ackermann rule. For different premises, the combination is performed as nine different cases (see Fig. 1). Soundness of the rule follows from the following two lemmas.

$$\begin{array}{c}
\mathcal{N}^{-\mathsf{A}}, \overbrace{C_1 \sqcup \mathsf{A}, \ldots, C_l \sqcup \mathsf{A}}^{\mathcal{P}^+_{\bigstar}(\mathsf{A})}, \overbrace{D_1 \sqcup {\geq} w_1 Q_1.\mathsf{A}, \ldots, D_m \sqcup {\geq} w_m Q_m.\mathsf{A}}^{\mathcal{P}^+_{\geq}(\mathsf{A})}, \overbrace{E_1 \sqcup {\leq} x_1 R_1.\neg\mathsf{A}, \ldots, E_n \sqcup {\leq} x_n R_n.\neg\mathsf{A}}^{\mathcal{P}^+_{\leq}(\mathsf{A})} \\
\overbrace{F_1 \sqcup \neg\mathsf{A}, \ldots, F_{l'} \sqcup \neg\mathsf{A}}^{\mathcal{P}^-_{\bigstar}(\mathsf{A})}, \overbrace{G_1 \sqcup {\geq} y_1 S_1.\neg\mathsf{A}, \ldots, G_{m'} \sqcup {\geq} y_{m'} S_{m'}.\neg\mathsf{A}}^{\mathcal{P}^-_{\geq}(\mathsf{A})}, \overbrace{H_1 \sqcup {\leq} z_1 T_1.\mathsf{A}, \ldots, H_{n'} \sqcup {\leq} z_{n'} T_{n'}.\mathsf{A}}^{\mathcal{P}^-_{\leq}(\mathsf{A})} \\
\hline
\mathcal{N}^{-\mathsf{A}}, \textbf{Block}(\mathcal{P}^+_{\bigstar}(\mathsf{A}), \mathcal{P}^-_{\bigstar}(\mathsf{A})), \textbf{Block}(\mathcal{P}^+_{\bigstar}(\mathsf{A}), G_1 \sqcup {\geq} y_1 S_1.\neg\mathsf{A}), ..., \textbf{Block}(\mathcal{P}^+_{\bigstar}(\mathsf{A}), G_{m'} \sqcup {\geq} y_{m'} S_{m'}.\neg\mathsf{A}), \\
\textbf{Block}(\mathcal{P}^+_{\bigstar}(\mathsf{A}), H_1 \sqcup {\leq} z_1 T_1.\mathsf{A}), ..., \textbf{Block}(\mathcal{P}^+_{\bigstar}(\mathsf{A}), H_{n'} \sqcup {\leq} z_{n'} T_{n'}.\mathsf{A}), \textbf{Block}(\mathcal{P}^-(\mathsf{A}), D_1 \sqcup {\geq} w_1 Q_1.\mathsf{A}), \ldots, \\
\textbf{Block}(\mathcal{P}^-(\mathsf{A}), D_m \sqcup {\geq} w_m Q_m.\mathsf{A}), \textbf{Block}(\mathcal{P}^-(\mathsf{A}), E_1 \sqcup {\leq} x_1 R_1.\neg\mathsf{A}), ..., \textbf{Block}(\mathcal{P}^-(\mathsf{A}), E_n \sqcup {\leq} x_n R_n.\neg\mathsf{A})
\end{array}$$

Notation in the Ackermann rule ($1 \leq j \leq m, 1 \leq k \leq n, 1 \leq j' \leq m', 1 \leq k' \leq n'$):

Block$(\mathcal{P}^-(\mathsf{A}), \alpha) =$ **Block**$(\mathcal{P}^-_{\bigstar}(\mathsf{A}), \alpha) \cup$ **Block**$(\mathcal{P}^-_{\geq}(\mathsf{A}), \alpha) \cup$ **Block**$(\mathcal{P}^-_{\leq}(\mathsf{A}), \alpha)$, where $\alpha \in \mathcal{P}^+_{\geq}(\mathsf{A}) \cup \mathcal{P}^+_{\leq}(\mathsf{A})$

Case 1: Block$(\mathcal{P}^+_{\bigstar}(\mathsf{A}), \mathcal{P}^-_{\bigstar}(\mathsf{A}))$ denotes the set $\{C_1 \sqcup (F_1 \sqcap \ldots \sqcap F_{l'}), \ldots, C_l \sqcup (F_1 \sqcap \ldots \sqcap F_{l'})\}$.

Case 2: Block$(\mathcal{P}^+_{\bigstar}(\mathsf{A}), G_{j'} \sqcup {\geq} y_{j'} S_{j'}.\neg\mathsf{A})$ denotes the set $\{G_{j'} \sqcup {\geq} y_{j'} S_{j'}.(C_1 \sqcap \ldots \sqcap C_l)\}$.

Case 3: Block$(\mathcal{P}^+_{\bigstar}(\mathsf{A}), H_{k'} \sqcup {\leq} z_{k'} T_{k'}.\mathsf{A}\}$ denotes the set $\{H_{k'} \sqcup {\leq} z_{k'} T_{k'}.\neg(C_1 \sqcap \ldots \sqcap C_l)\}$.

Case 4: Block$(\mathcal{P}^-_{\bigstar}(\mathsf{A}), D_j \sqcup {\geq} w_j Q_j.\mathsf{A})$ denotes the set $\{D_j \sqcup {\geq} w_j Q_j.(F_1 \sqcap \ldots \sqcap F_{l'})\}$.

Case 5: Block$(\mathcal{P}^-_{\bigstar}(\mathsf{A}), E_k \sqcup {\leq} x_k R_k.\neg\mathsf{A})$ denotes the set $\{E_k \sqcup {\leq} x_k R_k.\neg(F_1 \sqcap \ldots \sqcap F_{l'})\}$.

Case 6: Block$(\mathcal{P}^-_{\geq}(\mathsf{A}), D_j \sqcup {\geq} w_j Q_j.\mathsf{A})$ denotes the union of the following sets:

$$\{D_j \sqcup {\geq} w_j Q_j.\top\} \text{ and } \bigcup_{1 \leq j' \leq m'} \{G_{j'} \sqcup {\geq} y_{j'} S_{j'}.\top\} \text{ and } \bigcup_{1 \leq j' \leq m'} \{D_j \sqcup G_{j'} \sqcup (w_j + y_{j'})(Q_j \sqcup S_{j'}).\top\}$$

Case 7: Block$(\mathcal{P}^-_{\geq}(\mathsf{A}), E_k \sqcup {\leq} x_k R_k.\neg\mathsf{A})$ denotes the union of the following sets:

$$\bigcup_{1 \leq j' \leq m'} \{G_{j'} \sqcup {\geq} y_{j'} S_{j'}.\top\} \text{ and } \bigcup_{1 \leq j' \leq m'} \{G_{j'} \sqcup E_k \sqcup {\geq}(y_{j'} - x_k)(S_{j'} \sqcap \neg R_k).\top\} \text{ for any } y_{j'} > x_k$$

Case 8: Block$(\mathcal{P}^-_{\leq}(\mathsf{A}), D_j \sqcup {\geq} w_j Q_j.\mathsf{A})$ denotes the union of the following sets:

$$\{D_j \sqcup {\geq} w_j Q_j.\top\} \text{ and } \bigcup_{1 \leq k' \leq n'} \{D_j \sqcup H_{k'} \sqcup {\geq}(w_j - z_{k'})(Q_j \sqcap \neg T_{k'}).\top\} \text{ for any } w_j > z_{k'}$$

Case 9: Block$(\mathcal{P}^-_{\leq}(\mathsf{A}), E_k \sqcup {\leq} x_k R_k.\neg\mathsf{A})$ denotes the set $\displaystyle\bigcup_{1 \leq k' \leq n'} \{E_k \sqcup H_{k'} \sqcup {\leq}(x_k + z_{k'})(R_k \sqcap T_{k'}).\top\}$

Figure 1: The Ackermann rule for eliminating $\mathsf{A} \in \mathrm{sig}_{\mathsf{C}}(\mathcal{N})$ from a set $\mathcal{N}$ of TBox clauses in reduced form

Lemma 1. *Cases 1, 2, 3, 4 and 5 in Figure 1 preserve equivalence up to the interpretation of the pivot.*

Proof. For Cases 1, 2, 3, 4 and 5, the idea of the combination is analogous to that of Ackermann's Lemma, where the pivot is eliminated by computing the definition of the pivot from the negative (positive) premises and substituting this definition for the pivot in every positive (negative) premise. □

The preservation of equivalence in Cases 6, 7, 8 and 9 is open (because of the cases discussed in Section 3). We show that the results in these cases are at least uniform interpolants.

Lemma 2. *Cases 6, 7, 8 and 9 in Figure 1 preserve all logical consequences up to the names in* $\mathsf{sig}_{\mathsf{C}}(\mathcal{N}) \backslash \mathcal{F}$.

Proof (sketch). For space reasons, we only prove Case 8. The other cases can be proved similarly. $\{D_j \sqcup {\geq} w_j Q_j.\top\}$ follows directly from the premises. Assume a domain element has minimally x R-successors satisfying A and maximally y S-successors satisfying A, then, if $x > y$, it has minimally $x - y$ R- and $\neg S$-successors satisfying A or $\neg A$ ($\top$). Thus, $\{D_j \sqcup H_{k'} \sqcup {\geq}(w_j - z_{k'})(Q_j \sqcap \neg T_{k'}).\top\}$ holds.

Two special cases are that: A occurs (i) only positively, or (ii) only negatively, in $\mathcal{N}$. Then A is said to be *pure* in $\mathcal{N}$ (*impure*, otherwise). In Case (i), the solution of forgetting A from $\mathcal{N}$ is computed by substituting $\top$ for every occurrence of A in $\mathcal{N}$, and in Case (ii), the solution is computed by substituting $\bot$ for every occurrence of A in $\mathcal{N}$. This is referred to as the *purification*, which preserves equivalence up to the interpretations of the pivot [Zhao and Schmidt, 2016].

5 The Forgetting Method

The input to the method are a set $\mathsf{cls\_set}$ of TBox clauses and a set $\mathsf{f\_sig}$ of concept names to be forgotten. Our method performs purification first (on the entire clause set), because purification often results in numerous redundancies, tautologies and contradictions inside the clauses, which are immediately simplified or eliminated thereby leading to a much reduced clause set. This makes subsequent forgetting less difficult. A notable feature of the method is that impure names are eliminated in a *local* manner, that is, a concept name $A \in \mathsf{f\_sig}$ is eliminated from only the A-clauses, rather than the entire clause set. This significantly reduces the search space. In this way, the method eliminates the concept names in $\mathsf{f\_sig}$ one by one. Once all these concept names have been eliminated, the method attempts to eliminate the introduced definers to compute the forgetting solution. Yet the definer elimination is not always successful; it may fail when the original concept forgetting involves cyclic dependencies. For example, forgetting the concept name $\{A\}$ from $\{\neg A \sqcup \exists r.A\}$ yields $\{\neg \mathsf{D} \sqcup \exists r.\mathsf{D}\}$, where $\mathsf{D} \in \mathsf{N_D}$ is a fresh definer. Observe that the clause in the resulting set has the same pattern with the one in the given set. Thus, eliminating the definer $\{\mathsf{D}\}$ from $\{\neg \mathsf{D} \sqcup \exists r.\mathsf{D}\}$ would, as one can expect, yield another 'same pattern' clause. This problem can be solved using fixpoints [Koopmann and Schmidt, 2013b], with which in the target language we may even make our method complete. However, since at present mainstream tools do not support fixpoints, we do not include them for practicality of the method. What the method returns as output are a finite set $\mathsf{cls\_set}$ of TBox clauses. If $\mathsf{cls\_set}$ does not contain any names in $\mathsf{f\_sig}$ or any introduced definers, then the method was *successful*. The following theorem states termination and soundness of the method.

Theorem 3. *For any $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$-ontology $\mathcal{O}$ and any forgetting signature $\mathcal{F} \subseteq \mathrm{sig}_\mathsf{C}(\mathcal{O})$, our method always terminates and returns a finite set $\mathcal{O}'$ of TBox clauses. If $\mathcal{O}'$ does not contain any names in $\mathcal{F}$ or any introduced definers, then $\mathcal{O}'$ is either a* semantic solution *of forgetting $\mathcal{F}$ from $\mathcal{O}$, or a* uniform interpolant *of $\mathcal{O}$ for the signature $\mathsf{sig_C}(\mathcal{N}) \backslash \mathcal{F}$.*

Theorem 4. *Given an $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$-ontology $\mathcal{O}$ and a forgetting signature $\mathcal{F} \subseteq \mathrm{sig}_\mathsf{C}(\mathcal{O})$, our method is guaranteed to compute a semantic solution of forgetting $\mathcal{F}$ from $\mathcal{O}$ if the following conditions hold for each $A \in \mathcal{F}$: (i) $\mathcal{O}$ is an* acyclic *TBox w.r.t. the names in $\mathcal{F}$ (including internalised ABox axioms), and (ii) A does not occur positively (negatively) under a $\geq$-restriction and a $\leq$-restriction.*

Proof (sketch). Condition (i) ensures all definers can be eliminated from $\mathcal{O}$. Condition (ii) means in the reduced form of $\mathcal{O}$ for any $A \in \mathcal{F}$, $\mathcal{P}^+_{\geq}(A) = \emptyset$ and $\mathcal{P}^+_{\leq}(A) = \emptyset$, or $\mathcal{P}^-_{\geq}(A) = \emptyset$ and $\mathcal{P}^-_{\leq}(A) = \emptyset$. This avoids Cases 6, 7, 8 and 9.

6 Evaluation and Empirical Results

In order to gain insight into the practicality of our forgetting method, we implemented a prototype in Java using the OWL API[2] and evaluated it on a corpus of biomedical ontologies. The corpus was based on a snapshot of the BioPortal repository taken in March 2017 [Matentzoglu and Parsia, 2017], containing 396 OWL API compatible ontologies.

Because the ontologies in the snapshot can be as expressive as $\mathcal{SHOIN}(\mathcal{D})$ and $\mathcal{SROIQ}(\mathcal{D})$, we adjusted the selected ontologies to the language of $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$ using simple simulations and replacements, e.g., an exact number restriction $=1r.C$ was simulated by$\geq 1r.C \sqcap \leq 1r.C$ and concepts not expressible in $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$ were replaced by $\top$. Table 1 shows statistical information about the adjusted ontologies, where $\#(\mathcal{O})$ denotes the numbers of axioms in the ontologies, and $\#\mathsf{sig_C}(\mathcal{O})$, $\#\mathsf{sig_R}(\mathcal{O})$ and $\#\mathsf{sig_I}(\mathcal{O})$ denote respectively the numbers of the concept names, role names and individual names in the ontologies. It was found that 38.9% of the test ontologies included ABoxes (154 out of 396).

	Max.	**Min.**	**Mean**	**Mdn.**	**90th Ptl.**
$\#(\mathcal{O})$	1.8M	100	4.6K	1.1K	12.6K
$\#\mathsf{sig_C}(\mathcal{O})$	847.8K	36	2.1K	502	5.6K
$\#\mathsf{sig_R}(\mathcal{O})$	1.4K	0	54	12	144
$\#\mathsf{sig_I}(\mathcal{O})$	87.9K	0	216	0	206

Mdn.: Median, 90th Pct.: 90th Percentile

Table 1: Statistics of adjusted ontologies used for our evaluation

Settings	Results (unit of time: second)				
Forget %	**Time**	**T.O.**	**S. Rate**	$\mathsf{N_D}$	**S. Sol.**
10%	3.131	1.3%	96.2%	2.5%	25.0%
30%	8.995	4.0%	89.7%	6.3%	8.8%
50%	14.213	7.5%	85.2%	8.8%	5.8%
Avg.	8.780	4.3%	90.4%	5.9%	13.2%

T.O.: Timeout, S. Rate: Success Rate, S. Sol.: Semantic Solution

Table 2: Results of forgetting 10%, 30% and 50% of concept names

To fit in with different real-world application scenarios, we evaluated the performance of the prototype for forgetting different numbers of concept names from each test ontology. In particular, we considered respectively the cases of forgetting 10%, 30% and 50% of concept names in the signature of each ontology. The names to be forgotten were randomly selected. The experiments were conducted on a desktop computer with an Intel® Core™ i7-4790 processor, four cores running at up to 3.60 GHz, and 8 GB of DDR3-1600 MHz RAM. The experiments were run 100 times on each ontology and we averaged the results in order to verify the accuracy of our findings. A timeout of 1000 seconds was imposed on each run.

The results are shown in Table 2. The most encouraging results are that on average: (i) the prototype was successful (i.e., eliminated all specified concept names and the introduced definers) in more than 90% of the test cases, and (ii) in most of these cases forgetting solutions were computed within 10 seconds. The column headed T.O. shows the percentages of the test cases where the solutions could not be computed within the timeout. The column headed $\mathsf{N_D}$ shows the percentages of the test cases where the definers could not be all eliminated from the results. The column headed S. Sol. shows the percentages of the test cases where the solution was a semantic solution; in 13.2% of the successful cases, the solution was a semantic solution and it was a uniform interpolant in the other cases. An important finding, which is not shown in the table, is that role constructs ($\neg$, $\sqcap$ and $\sqcup$) occurred in the forgetting solutions in 32% of the test cases.

7 Conclusions

In this paper, we developed an Ackermann-based method for computing solutions of concept forgetting in the description logic $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$. When it succeeds, the method outputs either a semantic solution of concept forgetting or a uniform interpolant in $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$. An evaluation with a prototype has shown that the method is practical and semantic solutions of concept forgetting are rare for $\mathcal{ALCOQ}(\neg, \sqcap, \sqcup)$.

[2]https://github.com/owlcs/owlapi

References

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

[Bicarregui *et al.*, 2001] J. Bicarregui, T. Dimitrakos, D. M. Gabbay, and T. S. E. Maibaum. Interpolation in practical formal development. *Logic Journal of the IGPL*, 9(2):231–244, 2001.

[Botoeva *et al.*, 2016] E. Botoeva, B. Konev, C. Lutz, V. Ryzhikov, F. Wolter, and M. Zakharyaschev. Inseparability and conservative extensions of description logic ontologies: A survey. In *Proc. RW'14*, volume 9885 of *LNCS*, pages 27–89. Springer, 2016.

[Doherty *et al.*, 1997] P. Doherty, W. Lukaszewicz, and A. Szalas. Computing circumscription revisited: A reduction algorithm. *J. Autom. 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*, pages 425–435. Morgan Kaufmann, 1992.

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

[Ghilardi *et al.*, 2006] S. Ghilardi, C. Lutz, and F. Wolter. Did i damage my ontology? A case for conservative extensions in description logics. In *Proc. KR'06*, pages 187–197. AAAI Press, 2006.

[Grädel *et al.*, 1997] E. Grädel, M. Otto, and E. Rosen. Two-variable logic with counting is decidable. In *Proc. LICS'97*, pages 306–317. IEEE Computer Society, 1997.

[Grau and Motik, 2012] B. C. Grau and B. Motik. Reasoning over ontologies with hidden content: The import-by-query approach. *J. Artif. Intell. Res.*, 45:197–255, 2012.

[Konev *et al.*, 2008] B. Konev, D. Walther, and F. Wolter. The logical difference problem for description logic terminologies. In *Proc. IJCAR'08*, volume 5195 of *LNCS*, pages 259–274. Springer, 2008.

[Konev *et al.*, 2009] B. Konev, D. Walther, and F. Wolter. Forgetting and uniform interpolation in large-scale description logic terminologies. In *Proc. IJCAI'09*, pages 830–835. IJCAI/AAAI Press, 2009.

[Konev *et al.*, 2013] B. Konev, C. Lutz, D. Walther, and F. Wolter. Model-theoretic inseparability and modularity of description logic ontologies. *Artif. Intell.*, 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*, volume 8312 of *LNCS*, pages 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*, volume 8152 of *LNCS*, pages 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*, volume 8562 of *LNCS*, pages 434–448. Springer, 2014.

[Koopmann and Schmidt, 2015a] 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*, 2015.

[Koopmann and Schmidt, 2015b] 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.

[Koopmann, 2015] P. Koopmann. *Practical Uniform Interpolation for expressive Description Logics*. PhD thesis, The University of Manchester, UK, 2015.

[Lang *et al.*, 2003] J. Lang, P. Liberatore, and P. Marquis. Propositional independence: Formula-variable independence and forgetting. *J. Artif. Intell. Res.*, 18:391–443, 2003.

[Lin and Reiter, 1994] F. Lin and R. Reiter. Forget It! In *Proc. AAAI Fall Symposium on Relevance*, pages 154–159. AAAI Press, 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*, pages 318–327. AAAI Press, 2014.

[Lutz and Wolter, 2010] C. Lutz and F. Wolter. Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. *J. Symb. Comput.*, 45(2):194–228, 2010.

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

[Lutz *et al.*, 2007] C. Lutz, D. Walther, and F. Wolter. Conservative extensions in expressive description logics. In *Proc. IJCAI'07*, pages 453–458, 2007.

[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*. AAAI Press, 2012.

[Matentzoglu and Parsia, 2017] N. Matentzoglu and B. Parsia. BioPortal Snapshot 30.03.2017, March 2017.

[Nikitina and Rudolph, 2014] N. Nikitina and S. Rudolph. (Non-)Succinctness of uniform interpolants of general terminologies in the description logic $\mathcal{EL}$. *Artif. Intell.*, 215:120–140, 2014.

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

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

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

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

[Wernhard, 2013] C. Wernhard. Abduction in logic programming as second-order quantifier elimination. In *Proc. FroCoS'13*, volume 8152 of *LNCS*, pages 103–119. Springer, 2013.

[Zhang and Zhou, 2010] Y. Zhang and Y. Zhou. Forgetting Revisited. In *Proc. KR'10*, pages 602–604. AAAI Press, 2010.

[Zhao and Schmidt, 2015] Y. Zhao and R. A. Schmidt. Concept forgetting in $\mathcal{ALCOI}$-ontologies using an Ackermann approach. In *Proc. ISWC'15*, volume 9366 of *LNCS*, pages 587–602. Springer, 2015.

[Zhao and Schmidt, 2016] Y. Zhao and R. A. Schmidt. Forgetting concept and role symbols in $\mathcal{ALCOIH}\mu^+(\nabla,\sqcap)$ ontologies. In *Proc. IJCAI'16*, pages 1345–1352. IJCAI/AAAI Press, 2016.

[Zhao and Schmidt, 2017] Y. Zhao and R. A. Schmidt. Role forgetting for $\mathcal{ALCOQH}(\nabla)$-ontologies using an Ackermann-based approach. In *Proc. IJCAI'17*, pages 1354–1361. IJCAI/AAAI Press, 2017.