Concept Forgetting in $\mathcal{ALCOI}$-Ontologies using an Ackermann Approach

Yizheng Zhao and Renate A. Schmidt

The University of Manchester, UK

Abstract. We present a method for forgetting concept symbols in ontologies specified in the description logic $\mathcal{ALCOI}$. The method is an adaptation and improvement of a second-order quantifier elimination method developed for modal logics and used for computing correspondence properties for modal axioms. It follows an approach exploiting a result of Ackermann adapted to description logics. Important features inherited from the modal approach are that the inference rules are guided by an ordering compatible with the elimination order of the concept symbols. This provides more control over the inference process and reduces non-determinism, and the size of the search space. The method is extended with a new case splitting inference rule, and several simplification rules. Compared to related forgetting and uniform interpolation methods for description logics, the method can handle inverse roles, nominals and ABoxes. Compared to the modal approach on which it is based, it is more efficient in time and has higher success rates. The method has been implemented in Java using the OWL API. Preliminary experimental results show that the order in which the concept symbols are eliminated significantly affects the success rate and efficiency.

1 Introduction

Ontology-based technologies provide novel ways of building knowledge processing systems and play an important role in many different areas, both in research projects but also in industry applications. Big ontologies contain however large numbers of symbols and knowledge modelled in them is rich and inevitably heterogeneous. Thus there are situations, where it is useful to be able to restrict the ontology to a subset of the signature and *forget* those symbols which do not belong to the subset, for example, when an ontology needs to be analysed by an ontology engineer to gain an understanding of the information represented in it. Another example is a scenario where ontologies are distributed at separate remote sites and information is exchanged via agents. Since the vocabularies known to the agents at the different sites will vary, communication between the agents needs to be limited to using the common language to avoid ambiguity and confusion caused by the inconsistency of the vocabularies being used. At this point, it would be beneficial if the signature symbols in one ontology that are not known to the other agents can be eliminated without losing information required for the communication. In other words, signature symbols belonging

to only one of the ontologies are forgotten, and communication is restricted to information expressed in the shared language of the agents' ontologies. Another use of forgetting is restricting the vocabulary of an ontology to more general concept symbols, and forgetting those that are more specific, to create a summary of the ontology [29]. Situations where ontologies are published, shared, or disseminated, but some sensitive parts described in terms of particular signature symbols needs to be kept confidential or unseen to the receiver, is a potential application of forgetting [3]. This is relevant for medical and military uses, and uses in industry to ensure proprietary information can be kept hidden.

The contribution of this paper is the presentation of a method for forgetting concept symbols in ontologies specified in the description logic $\mathcal{ALCOI}$. $\mathcal{ALCOI}$ extends the description logic $\mathcal{ALC}$ with nominals and inverse roles. Forgetting concept symbols for $\mathcal{ALCOI}$ is a topic where no method is available yet, but a number of related methods exist. Forgetting can be viewed as the problem dual to uniform interpolation. A lot of recent work has been focussed on uniform interpolation of mainly TBoxes represented in several description logics, ranging from ones with more limited expressivity, such as DL-Lite [31] and $\mathcal{EL}$ [20, 22] and $\mathcal{EL}$-extensions [11], to more expressive ones, such as $\mathcal{ALC}$ [21, 30, 19, 14, 13], $\mathcal{ALCH}$ [12], $\mathcal{SIF}$ [17] and $\mathcal{SHQ}$ [15].

Forgetting can however also be viewed as a second-order quantification problem, which is the view we take in this paper. In second-order quantifier elimination, the aim is to eliminate existentially quantified predicate symbols in order to translate second-order formulae into equivalent formulae in first-order logic [5, 6, 8, 4, 7, 26, 23, 24, 28]. In uniform interpolation the aim is to eliminate symbols too, though it is not required that the result is logically equivalent to the corresponding formula in second-order logic, only that all important consequences are preserved.

Our method is adapted from a method, called MSQEL, designed for modal logic to compute first-order frame correspondence properties for modal axioms and rules [26]. The adaptation exploits the close relationship between description logics and modal logics [25]. Our method contributes three novel aspects. It is the first method for forgetting concept symbols from ontologies specified in the description logic $\mathcal{ALCOI}$. It inherits from MSQEL the consideration of elimination orders, which has been shown to improve the success rate and make it succeed on a wider range for problems in the modal logic corresponding to $\mathcal{ALCOI}$ [26]. The success rate and its scope is further improved by the incorporation of a new case splitting rule and generalised simplification rules. Results of an empirical evaluation show improved success rates and performance for these techniques.

2 Definition of $\mathcal{ALCOI}$ and Other Basic Definitions

Let N_C and N_R be the set of atomic concepts and the set of atomic roles, respectively, and let N_O be the set of nominals. $\mathcal{ALCOI}$-concepts have one of these forms:

$$a \mid \bot \mid \top \mid A \mid \neg C \mid C \sqcup D \mid C \sqcap D \mid \exists R.C \mid \exists R^-.C \mid \forall R.C \mid \forall R^-.C,$$

where $a \in N_O$, $A \in N_C$, $R \in N_R$, and C and D are arbitrary $\mathcal{ALCOI}$-concepts. R^- denotes the inverse of the role R. By definition, R^{--} is expressively the same as R.

An ontology usually consists of two parts, namely a TBox and an ABox. A TBox contains a set of axioms of the form $C \sqsubseteq D$ or $C \equiv D$, where C and D are concepts. A concept definition $C \equiv D$ can be expressed by two general inclusion axioms $C \sqsubseteq D$ and $D \sqsubseteq C$. In $\mathcal{ALCOI}$, ABox axioms can be expressed as inclusions in the TBox: a concept assertion $C(a)$ can be expressed as $a \sqsubseteq C$, and a role assertion $R(a, b)$ as $a \sqsubseteq \exists R.b$. In this paper, we treat the ABox axioms as inclusions, consequently in our considerations $\mathcal{ALCOI}$-ontologies are assumed to contain TBox axioms only.

We define an interpretation $\mathcal{I}$ for $\mathcal{ALCOI}$ over the signature (N_C, N_R, N_O) as the pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ is a non-empty set that represents the interpretation domain, and $\cdot^{\mathcal{I}}$ is the interpretation function that assigns to every nominal $a \in N_O$ a singleton set $a^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$; to every concept symbol $A \in N_C$ a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$; and to every role symbol $R \in N_R$ a subset $R^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. We specify the semantics of $\mathcal{ALCOI}$-concepts by extending the interpretation function to the following:

$$\bot^{\mathcal{I}} = \emptyset \qquad (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}} \qquad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$
$$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y.(x, y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$$
$$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y.(x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$$
$$(R^-)^{\mathcal{I}} = \{(y, x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (x, y) \in R^{\mathcal{I}}\}$$

The semantics of the TBox-axioms is defined as follows: an interpretation $\mathcal{I}$ *satisfies* $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and $\mathcal{I}$ *satisfies* $C \equiv D$ iff $C^{\mathcal{I}} \equiv D^{\mathcal{I}}$. If $\mathcal{O}$ is a set of TBox axioms, $\mathcal{I}$ is a *model* of $\mathcal{O}$ iff it satisfies every axiom in $\mathcal{O}$, denoted by $\mathcal{I} \models \mathcal{O}$.

In the rest of the paper, we also need the following definitions. A *clause* is a disjunction of $\mathcal{ALCOI}$-concepts. Let A be a concept symbol and let $\mathcal{I}$ and $\mathcal{I}'$ be interpretations. We say $\mathcal{I}$ and $\mathcal{I}'$ are *A-equivalent*, if $\mathcal{I}$ and $\mathcal{I}'$ coincide but differ possibly in the valuation assigned to A. This means their domains coincide, i.e., $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}'}$, and for each symbol s in the signature except for A, $s^{\mathcal{I}} = s^{\mathcal{I}'}$. More generally, suppose $\Sigma = \{A_1, \ldots, A_m\} \subseteq N_C$, $\mathcal{I}$ and $\mathcal{I}'$ are *Σ-equivalent*, if $\mathcal{I}$ and $\mathcal{I}'$ are the same but differ possibly in the valuations assigned to the concept symbols in Σ.

3 Forgetting as Second-Order Quantifier Elimination

We are interested in forgetting concept symbols in axioms of an ontology $\mathcal{O}$ of TBox axioms. Let $\mathrm{sig}(\mathcal{O})$ denote the signature of $\mathcal{O}$.

Definition 1. *Let $\mathcal{O}$ and $\mathcal{O}'$ be $\mathcal{ALCOI}$-ontologies and let $\Sigma = \{A_1, \ldots, A_m\}$ be a set of concept symbols. $\mathcal{O}'$ is the* result of forgetting the symbols in Σ from $\mathcal{O}$,

if $\mathrm{sig}(\mathcal{O}') \subseteq \mathrm{sig}(\mathcal{O}) \backslash \Sigma$ *and for any interpretation* $\mathcal{I}$*,*

$$\mathcal{I} \models \mathcal{O} \quad \textit{iff} \quad \mathcal{I}' \models \mathcal{O}' \quad \textit{for some interpretation } \mathcal{I}' \ \Sigma\textit{-equivalent to } \mathcal{I}.$$

The symbols in Σ *are the* symbols to be forgotten*. We refer to them as the* non-base symbols *and the symbols in* $\mathrm{sig}(\mathcal{O}) \backslash \Sigma$ *as the* base symbols*. The result of forgetting a concept symbol* A *from* $\mathcal{O}$ *is the result of forgetting* $\{A\}$ *from* $\mathcal{O}$*.*

The result of forgetting a symbol A from an ontology $\mathcal{O}$ can be represented as $\exists X\, \mathcal{O}^A_X$ in the extension of the language with existentially quantified concept variables. $\mathcal{O}^A_X$ is our notation for substituting every occurrence of A is $\mathcal{O}$ by X. In general, for the target language which extends the (source) language of the logic under consideration with existential quantification of predicate symbols, the result of forgetting always exists. The challenge of forgetting, as a computational problem, is to find an ontology $\mathcal{O}'$ (without any occurrences of X) in the source language (without second-order quantification) that is equivalent to $\exists X\, \mathcal{O}^A_X$, where $\mathcal{O}$ is expressed in the source language. Finding such an ontology $\mathcal{O}'$ that is equivalent to $\exists X\, \mathcal{O}^A_X$ is an instance of the *second-order quantifier elimination problem*. Forgetting a concept symbol A is thus the problem of eliminating the existential quantifier $\exists X$ from $\exists X\, \mathcal{O}^A_X$. In the following, we slightly informally say the aim is to eliminate the symbol A from $\mathcal{O}$. For this we apply second-order quantifier elimination techniques to the axioms of $\mathcal{O}$ in order to forget A (the non-base symbol). In particular, we are going to exploit an adaptation of a result of Ackermann [1], which is known as Ackermann's Lemma in the literature on second-order quantifier elimination [8].

Theorem 1 (Ackermann's Lemma for $\mathcal{ALCOI}$). *Let* $\mathcal{O}$ *be an* $\mathcal{ALCOI}$*-ontology, let* C *be a concept expression and suppose the concept symbol* A *does not occur in* C*. Let* $\mathcal{I}$ *be an arbitrary* $\mathcal{ALCOI}$*-interpretation. (i) If* A *occurs only positively in* $\mathcal{O}$*, then* $\mathcal{I} \models \mathcal{O}^A_C$ *iff for some interpretation* $\mathcal{I}'$ A*-equivalent to* $\mathcal{I}$*,* $\mathcal{I}' \models A \sqsubseteq C, \mathcal{O}$*. (ii) If* A *occurs only negatively in* $\mathcal{O}$ *then* $\mathcal{I} \models \mathcal{O}^A_C$ *iff for some interpretation* $\mathcal{I}'$ A*-equivalent to* $\mathcal{I}$*,* $\mathcal{I}' \models C \sqsubseteq A, \mathcal{O}$*.*

4 The Forgetting Method DSQEL

Our forgetting method is called DSQEL, which is short for Description logics Second-order Quantifier ELimination.

Figure 1 outlines the basic routine of the DSQEL method to forget concept symbols in $\mathcal{ALCOI}$-ontologies $\mathcal{O}$. Once receiving the input ontology and a set Σ of concept symbols to forget, the method proceeds as follows. In *Phase 1*, a preprocessing step is performed to transform the axioms into a set of clauses. This is done by replacing all inclusions $C \sqsubseteq D$ by $\neg C \sqcup D$, and all equivalences $C \equiv D$ by $\neg C \sqcup D$ and $\neg D \sqcup C$. Inexpensive equivalence-preserving syntactic simplification rules are also applied in this phase to simplify clauses. For example, $C \sqcup (C \sqcap D)$ is simplified to C. *Phase 2* counts the number of positive (and negative) occurrences of each concept symbol in Σ. Using these counts an

1. Transform ontology $\mathcal{O}$ to clausal representation $N := \text{clause}(\mathcal{O})$.
2. Process every concept symbol A in Σ and check the frequency of A in polarity terms to generate the ordering $\succ$.
3. Guided by $\succ$, apply the DSQEL calculus to each of the concept symbols in Σ to produce the ontology $\mathcal{O}'$ (with clauses interpreted in the obvious way as inclusions).
4. Apply the simplification rules provided to $\mathcal{O}'$ if needed and return the resultant ontology containing the symbols only in $\text{sig}(\mathcal{O}')\backslash\Sigma$.

Fig. 1. The phases in the basic DSQEL routine

ordering $\succ$ is defined on the symbols in Σ. This ordering determines the order in which the symbols in Σ are eliminated in the next phase. *Phase 3* applies the DSQEL calculus described in the next section to the non-base symbols in Σ one by one, starting with the symbol A largest in the ordering $\succ$. To forget A the inference rules of the DSQEL calculus are applied to the axioms containing A. Then the next largest non-base symbol is eliminated, and so on.

Forgetting a concept symbol may lead to a change of the polarities of the occurrences of the remaining Σ-symbols, and a new elimination order may have to be computed based on the refreshed polarity counts, before the forgetting method to continues. This means Phase 2 and Phase 3 will be alternately and repeatedly executed with recomputed elimination orders. If the largest current concept symbols to be eliminated could not be completely eliminated by DSQEL, then a different ordering not attempted before will be used. In the case that all possible orderings have been tried and every attempt to eliminate all non-base symbols using DSQEL is not successful, the method returns failure, because it was unable to solve the problem. On the other hand, when after a call of DSQEL the set returned does not contain any non-base symbols, then this is the result of forgetting Σ from $\mathcal{O}$.

Phase 4 subsequently applies further simplification rules and transforms the resulting axioms to simpler representations.

Different elimination orders of concept symbols applied may lead to different but equivalent results. The results can be viewed (when the remaining non-base symbols are existentially quantified) as equivalent representations of $\exists\Sigma\,\mathcal{O}$.

What is returned by the algorithm, if it terminates successfully, is a (possibly empty) ontology with all occurrences of the non-base symbols eliminated. I.e., what is returned is an ontology specified in terms of only the symbols in $\text{sig}(\mathcal{O})\backslash\Sigma$.

There are situations where our method does not succeed, for instance, when no forgetting result finitely expressible in $\mathcal{ALCOI}$ exists. This means the method is not complete, but since no complete method can exist for forgetting, as considered in this paper, this is to be expected. Concept forgetting is already not always computable for the description logic $\mathcal{EL}$ [10]. We also note that concept symbols cannot be eliminated by our method does not necessarily mean that

Ackermann:	$\dfrac{N, C_1 \sqcup A, \ldots, C_n \sqcup A}{(N^A_{\sim C_1 \sqcup \ldots \sqcup \sim C_n})^{\neg\neg C_1, \ldots, \neg\neg C_n}_{C_1, \ldots, C_n}}$
provided:	(i) A is a non-base symbol, (ii) A does not occur in any of the C_i, (iii) A is strictly maximal wrt. each C_i, and (iv) N is negative with respect to A.
Purify:	$\dfrac{N}{(N^A_{\neg\top})^{\neg\neg\top}_{\top}}$
provided:	(i) A is a non-base symbol in N, and (ii) N is negative with respect to A.

Fig. 2. The elimination rules

they are ineliminable. It might be the case that they are eliminable, but simply our method is unable to find a solution.

We can show DSQEL algorithm is correct and is guaranteed to terminate. This follows as an adaptation of the correctness and termination of the MSQEL procedure proved in [26], since all adaptations of MSQEL to DSQEL preserve logical equivalence and the calculus given in the next section is correct and terminates.

5 The DSQEL Forgetting Calculus

The order in which the non-base symbols are eliminated is determined by the ordering $\succ$ computed in Phase 2 of the DSQEL algorithm. We say a concept symbol A is *strictly maximal* with respect to a concept C if for any concept symbol B ($\neq A$) in C, $A \succ B$.

A concept C is *positive* (*negative*) with respect to a concept symbol A iff all occurrences of A in C are *positive* (*negative*). A set of concepts N is *positive* (*negative*) with respect to a concept symbol A iff all occurrences of A in N are *positive* (*negative*).

The Ackermann rule and the Purify rule, given in Figure 2, are the forgetting rules in the DSQEL calculus, which will lead to the elimination of a non-base concept symbol. Both of them have to meet particular requirements on the form of the concepts to which they apply. N is a set of $\mathcal{ALCOI}$-clauses, and by N^D_C, we mean the set obtained from N by substituting the expression C for all occurrences of D in N, where C and D are both $\mathcal{ALCOI}$-concepts. The inference rules in the DSQEL calculus are restricted by a set of side-conditions; for example, the side-conditions of the Ackermann rule require that A must be a non-base symbol and does not occur in $C_1, \ldots, C_n$, no non-base symbol occurring in C_i ($1 \leq i \leq n$) is larger than A under the ordering $\succ$, and every occurrence

of A in N must be negative. The Purify rule can be seen as a special case of the Ackermann rule, since it eliminates the non-base symbols that occur only negatively, that is, when there are no positive occurrences of A.

Surfacing: $\dfrac{N, C \sqcup \forall R^{\sigma}.D}{N, \forall R^{\sigma,-}.C \sqcup D}$

provided: (i) A is the largest non-base symbol in $C \sqcup \forall R^{\sigma}.D$,
(ii) A does not occur in C, and
(iii) A occurs positively in $\forall R^{\sigma}.D$.

Skolemization: $\dfrac{N, \neg a \sqcup \neg\forall R^{\sigma}.C}{N, \neg a \sqcup \neg\forall R^{\sigma}.\neg b, \neg b \sqcup \sim C}$

provided: (i) A is the largest non-base symbol in $\neg a \sqcup \neg\forall R^{\sigma}.C$,
(ii) A occurs positively in $\neg\forall R^{\sigma}.C$, and
(iii) b is a new nominal.

Clausify: $\dfrac{N, C \sqcup \neg(D_1 \sqcup ... \sqcup D_n)}{N, C \sqcup \sim D_1, ..., C \sqcup \sim D_n}$

provided: (i) A is the largest non-base symbol in $C \sqcup \neg(D_1 \sqcup ... \sqcup D_n)$,
(ii) A occurs positively in $D_1 \sqcup ... \sqcup D_n$.

Sign Switching: $\dfrac{N}{(N^{A}_{\neg A})^{\neg\neg A}_{A}}$

provided: (i) N is closed with respect to the other rules,
(ii) A is the largest non-base symbol in N, and
(iii) Sign switching wrt. A has not been performed before.

Fig. 3. The rewriting rules

The rules in Figure 3, are used to rewrite the formulae so they can be transformed into the form where either the Ackermann rule or the Purify rule is applicable. To apply these rules, the positive occurrences of the non-base symbol first have to be made 'individually isolated'. In addition, every positive occurrence of the non-base symbol must occur at the top level as a literal in a clause [26], that is, they must not occur under a quantifier restriction operator or any other logical operator except for disjunction. This is done by repeatedly using the surfacing rule. By R^{σ}, we mean the composition of a sequence of roles and by $R^{\sigma,-}$ we mean the composition of the sequence of inverses of the roles in R^{σ} with the order in the sequence reversed.

The Skolemization rule rewrites the existential expression in a clause of the form $\neg a \sqcup \neg \forall R^{\sigma}.C$, where a is a nominal. The implicit existential quantifier in $\neg \forall R^{\sigma}.C$ is Skolemized by introducing a new Skolem constant (nominal) b.

The clausify rule transforms a concept of the form $C \sqcup \neg(D_1 \sqcup ... \sqcup D_n)$ into a set of clauses.

The Sign switching rule is used to switch the polarity of a non-base symbol. It is applicable only when no other rules in the calculus are applicable with respect to this non-base symbol and the Sign switching rule has not been performed on this non-base symbol before.

Case Splitting: $$\frac{N, \neg a \sqcup C_1 \sqcup ... \sqcup C_n}{N, \neg a \sqcup C_1 \mid ... \mid N, \neg a \sqcup C_n}$$

provided: (i) A is the largest non-base symbol in $\neg a \sqcup C_1 \sqcup ... \sqcup C_n$,
(ii) A occurs positively in $C_1 \sqcup ... \sqcup C_n$.

Fig. 4. The case splitting rule

A novelty in the DSQEL calculus is the *case splitting* inference rule given in Figure 4. It splits a clause of the form $\neg a \sqcup C_1 \sqcup ... \sqcup C_n$ into smaller subclauses $\neg a \sqcup C_1$, $\neg a \sqcup C_2$, ..., $\neg a \sqcup C_n$. A single clause $\neg a \sqcup C$, together with N, forms a *case*. The original formula means that at least one of the disjuncts C_i ($1 \leq i \leq n$) is true for a. The benefits of the case splitting rule are twofold. On the one hand, the case splitting rule makes up for a limitation of the Skolemization rule, because it splits a disjunction with more than two disjuncts into several smaller cases, which the Skolemization rule is able to handle. On the other hand, our tests show it reduces the search space and increases the success rate.

Condensing I: $$\frac{N\big[C \sqcup \forall R^{\sigma_1}.\forall R^{\sigma_1,-}. \ldots . \forall R^{\sigma_n}.\forall R^{\sigma_n,-}.(C \sqcup D)\big]}{N\big[C \sqcup D \sqcup \forall R^{\sigma_1} \bot\big]}$$

provided: (i) C and D are arbitrary concepts, and
(ii) $\sigma^i \leq \sigma^1$ for $1 \leq i \leq n$

$$\frac{N\big[C \sqcup \forall R^{\sigma_1}.\forall R^{\sigma_1,-}. \ldots . \forall R^{\sigma_n}.\forall R^{\sigma_n,-}.(C \sqcup D)\big]}{N\big[C \sqcup D \sqcup \forall R^{\sigma_n} \bot\big]}$$

provided: (i) C and D are arbitrary concepts, and
(ii) $\sigma^i \leq \sigma^n$ for $1 \leq i \leq n$

Fig. 5. Sample simplification rule

We also introduced several simplification rules to transform more expressions so that inference rules become applicable, they keep expressions in simpler forms for efficiency, and most importantly, they lead to success of forgetting in more cases. Figure 5 displays two cases of the simplification rules, called Condensing I, with which one can handle clauses of a particular pattern where other existing methods fail.

6 Empirical Results

In order to evaluate how the DSQEL method behaves on real-life ontologies, we implemented it in Java using the OWL API and applied the implementation to a set of ontologies from the NCBO BioPortal[1], a large repository of biomedical ontologies. The experiments were run on a machine 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.

Since DSQEL handles expressivity as far as $\mathcal{ALCOI}$, the ontologies for our evaluation were restricted to their $\mathcal{ALCOI}$-fragments, and axioms outside of the scope of $\mathcal{ALCOI}$ were dropped from the ontologies. Consequently, we used 292 ontologies from the repository for our evaluation. We ran the experiments on each ontology 100 times and averaged the results to explore how forgetting was influenced by the number of the concept symbols in an ontology. A timeout of 1000 seconds was used.

To fit with possible needs of applications, we conducted experiments where we forget 10%, 30%, and 50% of the concept symbols in the ontologies. The DSQEL algorithm processes each non-base symbol and counts the number of their positive (and negative) occurrences. Based on these counts, an elimination order is generated by a heuristic algorithm. In order to see how the elimination order affects the performance of the calculus, we ran two sets of experiments, where we omitted the analysis of the elimination order for one set, and applied the analysis to the other set. The evaluation results with respect to forgetting 10%, 30%, 50% of the concept symbols in the ontologies, without and with the analysis of the elimination order, are shown in Table 1.

It can be seen that, the analysis of the elimination order leads to a decrease in the average duration of the runs of every experiment, which means that it takes shorter time to complete the same task than when analysis of the elimination order is not performed. It is evident that the analysis for the elimination order has brought a positive effect on the overall success rate (increase by 8.1%) and the timeout rate (decrease by 5.9%).

Evaluations of more aspects are being conducted at the moment. These evaluations are focussed on, for example, how the case splitting rule makes a difference to the behaviour of the DSQEL calculus, and how our method compares to the related methods of SCAN [7], DLS [5], DLS* [6], SQEMA [4], MSQEL [26], and LETHE [16] for computing uniform interpolants, in terms of the success rate and efficiency (duration and timeouts).

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

Input		Experiment				Output
Axioms Avg.	Symbols Avg.	Percentage	Order	Timeouts	Duration Avg.	Success Rate
1407	876	10%	✗	3.8%	4.509 sec.	90.1%
			✓	1.7%	2.404 sec.	97.6%
		30%	✗	7.5%	8.562 sec.	88.4%
			✓	2.2%	2.753 sec.	95.5%
		50%	✗	13.4%	15.068 sec.	85.3%
			✓	3.1%	3.004 sec.	94.9%
1407	876	Average	✗	8.2%	9.380 sec.	87.9%
			✓	2.3%	2.720 sec.	96.0%

Table 1. Forgetting 10%, 30%, and 50% of concept symbols in ontologies

7 Related Work

Probably the most important early work on the elimination of second-order quantifiers is that of Ackermann [1] in 1935. Only in 1992, Gabbay and Ohlbach [7] developed the first practical algorithm, called SCAN. SCAN is a resolution-based second-order quantifier elimination algorithm and can be used to forget predicate symbols from first-order logic formulae [24].

It has been shown that the SCAN algorithm is complete and terminates for modal axioms belonging to the famous Sahlqvist class [9]. In 1994, the hierarchical theorem proving method was developed by Bachmair et al. [2] and it has been shown that it can be used to solve second-order quantification problems. Around the same time, in 1995, Szałas [27] described a different algorithm for the second-order quantifier elimination problem, which exploits Ackermann's Lemma. The method was further extended to the DLS algorithm by Doherty et al. [5]. DLS uses a generalised version of Ackermann's Lemma and allows the elimination of existential second-order quantifiers from second-order formulae, and obtaining corresponding first-order equivalents. Nonnengart and Szałas [23] generalised the main result underlying the DLS algorithm to include fixpoints. Based on this work, Doherty et al. [6] proposed the DLS* algorithm, which attempts the derivation of either an equivalent first-order formula or a fixpoint formula from the original formula. DLS and DLS* are Ackermann-based second-order quantifier elimination methods. Ackermann-based second-order quantifier elimination was first applied to description logics in [28] by Szałas, where description logics were extended by a form of second-order quantification over concepts. More recently, Conradie et al. [4] introduced the SQEMA algorithm, which is also an Ackermann-based method but for modal logic formulae. It is specialised to find correspondences between modal formulae and hybrid modal logic formulae (and first-order formulae). Schmidt [26] has extended SQEMA and developed MSQEL as a refinement. A key novelty is the use of elimination orders, and the presentation of second-order quantifier elimination as an abstract calculus.

Investigation of forgetting as uniform interpolation in more expressive description logics was started in [29] and [21]. The first approach to compute uni-

form interpolations for $\mathcal{ALC}$-TBoxes was presented in [29]. It is a tableau-based approach, where a disjunctive normal form is required for the representation of the TBox-axioms and the uniform interpolants are incrementally approximated. It was shown in [21] that deciding the existence of uniform interpolants that can be finitely represented in $\mathcal{ALC}$ without fixpoints is 2-ExpTime-complete and in the worst case, the size of uniform interpolants is triple exponential with respect to the size of the original TBox. The first goal-oriented method based on clausal resolution was presented in [19] for computing uniform interpolants of $\mathcal{ALC}$-TBoxes, where experimental results show the practicality for real-life ontologies. Koopmann and Schmidt presented another resolution-based method exploiting structural transformation to compute uniform interpolants of $\mathcal{ALC}$-ontologies, which uses fixpoint operators to make uniform interpolants finitely representable [14]. The method has been further extended to handle $\mathcal{ALCH}$ [12], $\mathcal{SIF}$ [17], $\mathcal{SHQ}$ [15], and $\mathcal{ALC}$ with ABoxes [18].

8 Conclusion and Future Work

We have presented a second-order quantifier elimination method, called Dsqel, for forgetting concept symbols in ontologies specified in the description logic $\mathcal{ALCOI}$.

It is adapted from Msqel, an Ackermann-based second-order quantifier elimination method for a multi-modal tense logic with second-order quantification. The method is enhanced with new inference and simplification rules. The adaptation was motivated for the purpose of applying the second-order quantifier elimination techniques to the area of knowledge representation, where description logics provide important logical formalisms.

We have had a prototype implementation of our forgetting method, fully realising the Dsqel method. It is known that the success of a forgetting problem is highly dependent on, apart from the calculus itself, the non-base symbols Σ to be forgotten, and the elimination order which the method follows. The evaluation results of first experiments reported in this paper show promising and very good success rates for concept symbol forgetting.

Optimisation to both the calculus and the implementation is underway. One optimisation being investigated is the incorporation of more simplification rules in order to increase the efficiency and success rate further. We are also currently working on finding a heuristic algorithm using a dynamic way of ordering the non-base symbols, because the elimination of a particular concept symbol may affect the optimality of the elimination order, and thus computing a new order may be beneficial. Research shows that the order in which the inference rules are applied is significant because, as we observed, it is a main factor affecting the efficiency of the method.

Extending the method to handle ontologies going expressively further than $\mathcal{ALCOI}$ is a direction of the ongoing research. To explore how the forgetting of role symbols can be incorporated into our method is also of interest.

References

1. W. Ackermann. Untersuchungen über das Eliminationsproblem der mathematischen Logik. *Mathematische Annalen*, 110(1):390–413, 1935.
2. L. Bachmair, H. Ganzinger, and U. Waldmann. Refutational theorem proving for hierarchic first-order theories. *Applicable Algebra in Engineering, Communication and Computing*, 5(3-4):193–212, 1994.
3. C. Bernardo and B. Motik. B.: Reasoning over ontologies with hidden content: The importby-query approach. *J. Artificial Intelligence Research*, 45:197–255, 2012.
4. 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.
5. P. Doherty, W. Łukaszewicz, and A. Szałas. Computing circumscription revisited: A reduction algorithm. *Journal of Automated Reasoning*, 18(3):297–336, 1997.
6. P. Doherty, W. Łukaszewicz, and A. Szałas. General domain circumscription and its effective reductions. *Fundam. Inf.*, 36(1):23–55, 1998.
7. D. M. Gabbay and H. J. Ohlbach. Quantifier elimination in second–order predicate logic. In *Principles of Knowledge Representation and Reasoning (KR92)*, pages 425–435. Morgan Kaufmann, 1992.
8. D. M. Gabbay, R. A. Schmidt, and A. Szałas. *Second Order Quantifier Elimination: Foundations, Computational Aspects and Applications*. College Publications, 2008.
9. V. Goranko, U. Hustadt, R. A. Schmidt, and D. Vakarelov. SCAN is complete for all sahlqvist formulae. In *Relational and Kleene-Algebraic Methods in Computer Science*, volume 3051 of *Lecture Notes in Computer Science*, pages 149–162, 2004.
10. B. Konev, C. Lutz, D. Walther, and F. Wolter. Model-theoretic inseparability and modularity of description logic ontologies. *Artificial Intelligence*, 203(0):66–103, 2013.
11. B. Konev, D. Walther, and F. Wolter. Forgetting and uniform interpolation in extensions of the description logic $\mathcal{EL}$. In *Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford, UK*, 2009.
12. P. Koopmann and R. A. Schmidt. Forgetting concept and role symbols in $\mathcal{ALCH}$-ontologies. In *Logic for Programming, Artificial Intelligence, and Reasoning*, volume 8312 of *Lecture Notes in Computer Science*, pages 552–567. Springer, 2013.
13. P. Koopmann and R. A. Schmidt. Implementation and evaluation of forgetting in ALC-ontologies. In *Proceedings of the 7th International Workshop on Modular Ontologies (WoMo 2013)*, volume CEUR-WS/Vol-1081 of *CEUR Workshop Proceedings*, pages 1–12. CEUR-WS.org, 2013.
14. P. Koopmann and R. A. Schmidt. Uniform interpolation of $\mathcal{ALC}$-ontologies using fixpoints. In *Frontiers of Combining Systems*, volume 8152 of *Lecture Notes in Computer Science*, pages 87–102. Springer, 2013.
15. P. Koopmann and R. A. Schmidt. Count and forget: Uniform interpolation of $\mathcal{SHQ}$-Ontologies. In *Automated Reasoning - IJCAR 2014. Proceedings*, pages 434–448. Springer, 2014.
16. P. Koopmann and R. A. Schmidt. LETHE: A saturation-based tool for non-classical reasoning, 2015. Manuscript, submitted.
17. P. Koopmann and R. A. Schmidt. Saturation-based forgetting in the description logic $\mathcal{SIF}$, 2015. This volume.
18. P. Koopmann and R. A. Schmidt. Uniform interpolation and forgetting $\mathcal{ALC}$-ontologies with aboxes. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 175–181, 2015.

19. M. Ludwig and B. Konev. Towards practical uniform interpolation and forgetting for $\mathcal{ALC}$ tboxes. In *Proceedings of the 26th International Workshop on Description Logics (DL-2013)*, volume 1014 of *CEUR-WS*, pages 377–389. CEUR-WS.org, 2013.
20. C. Lutz, I. Seylan, and F. Wolter. An automata-theoretic approach to uniform interpolation and approximation in the description logic $\mathcal{EL}$. In *Principles of Knowledge Representation and Reasoning: KR 2012*. AAAI Press, 2012.
21. C. Lutz and F. Wolter. Foundations for uniform interpolation and forgetting in expressive description logics. In *Proceedings of IJCAI 2011*, pages 989–995. IJCAI/AAAI, 2011.
22. N. Nikitina. Forgetting in general $\mathcal{EL}$ terminologies. In *Description Logics*, volume 745 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011.
23. A. Nonnengart and A. Szałas. A fixpoint approach to second-order quantifier elimination with applications to correspondence theory. 24:307–328, 1999.
24. H. J. Ohlbach. SCAN—elimination of predicate quantifiers. In M. A. McRobbie and J. K. Slaney, editors, *Automated Deduction: In proceedings of CADE-13*, volume 1104 of *Lecture Notes in Artificial Intelligence*, pages 161–165. Springer, 1996.
25. K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of IJCAI'91*, pages 466–471. Morgan Kaufmann, 1991.
26. R. A. Schmidt. The Ackermann approach for modal logic, correspondence theory and second-order reduction. *Journal of Applied Logic*, 10(1):52–74, 2012.
27. A. Szałas. On the correspondence between modal and classical logic: an automated approach. *Journal of Logic and Computation*, 3:605–620, 1993.
28. A. Szałas. Second-order reasoning in description logics. *Journal of Applied Non-Classical Logics*, 16(3-4):517–530, 2006.
29. K. Wang, Z. Wang, R. Topor, J. Pan, and G. Antoniou. Concept and role forgetting in $\mathcal{ALC}$ ontologies. In *The Semantic Web - ISWC 2009*, volume 5823 of *Lecture Notes in Computer Science*, pages 666–681. Springer, 2009.
30. 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.
31. Z. Wang, K. Wang, R. Topor, and J. Pan. Forgetting for knowledge bases in DL-lite. In *The Semantic Web: Research and Applications*, volume 5021 of *Lecture Notes in Computer Science*, pages 245–257. Springer, 2008.