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

Yizheng Zhao(✉) and Renate A. Schmidt

The University of Manchester, Manchester, UK
yizheng.zhao@student.manchester.ac.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. An important feature inherited from the modal approach is 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, resulting in a smaller 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 improves the success rates. The method has been implemented in Java using the OWL API. 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 large numbers of symbols and knowledge modelled in them is rich and inevitably heterogeneous. There are thus situations, where it is useful to be able to restrict the ontology to a subset of the signature and *forget* those symbols that 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. Other examples are scenarios where ontologies are located 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 mismatches between the vocabularies of the different agents. 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

M. Arenas et al. (Eds.): ISWC 2015, Part I, LNCS 9366, pp. 587–602, 2015.
DOI: 10.1007/978-3-319-25007-6_34

required for the communication. In other words, signature symbols belonging to only one of the ontologies are forgotten, and communication is confined 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 need to be kept confidential or unseen to the receiver, are some other potential applications of forgetting [4]. 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 that is 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 $\mathcal{ALC}$ [13,14,19,21,30], $\mathcal{ALCH}$ [12], $\mathcal{SIF}$ [17] and $\mathcal{SHQ}$ [15].

Forgetting can 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 [3,5–8,23,24,26,28]. 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 better success rates and performance for these techniques.

The rest of the paper is organised as follows. Section 2 defines basic notions of the problem of concept forgetting, including the syntax and semantics of $\mathcal{ALCOI}$-ontologies, the language that our proposed method is aimed for. A formal definition of concept forgetting for $\mathcal{ALCOI}$-ontologies follows in Section 3. Section 4 sketches the general method to forget selected concept symbols, and correctness and termination results are stated. The forgetting calculus is introduced in

Section 5, where all the inference rules and two important simplification rules are presented. Section 6 describes a heuristic method for calculating good forgetting orders of the concept symbols that need to be eliminated. Results of an empirical evaluation of the method are presented in Section 7. A brief chronological overview of the most related work on forgetting and second-order quantifier elimination is given in Section 8. We conclude in Section 9 with a summary of the work and an outline of directions of future work.

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

The basic syntactic elements in the language of $\mathcal{ALCOI}$ are the *atomic concepts*, *atomic roles*, and *nominals*. Together they form the *signature* of the language of $\mathcal{ALCOI}$. 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, $R^{--} := R$.

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 our considerations $\mathcal{ALCOI}$-ontologies are therefore 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 a 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:

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

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 notions. A *clause* is a disjunction of $\mathcal{ALCOI}$-concepts. Clauses in our calculus are interpreted as globally true, i.e., an interpretation $\mathcal{I}$ satisfies a clause C iff $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$.

By definition $\sim C = D$, if $C = \neg D$, else $\sim C = \neg D$. The $\sim$ operator helps avoid sequences of negations.

By R^{σ}, we denote the composition of a sequence of roles and by $R^{\sigma,-}$ we denote the composition of the sequence of inverses of the roles in R^{σ} with the order in the sequence reversed.

Let A be a concept symbol and let $\mathcal{I}$ and $\mathcal{I}'$ be interpretations. We say $\mathcal{I}$ and $\mathcal{I}'$ are *equivalent up to* A, or *A-equivalent*, if $\mathcal{I}$ and $\mathcal{I}'$ coincide but differ possibly in the 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 *equivalent up to* Σ, or *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 (i) $\mathrm{sig}(\mathcal{O}') \subseteq \mathrm{sig}(\mathcal{O}) \backslash \Sigma$ and (ii) 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 $\mathcal{O}$.*

Intuitively, the definition says that the forgetting result $\mathcal{O}'$ is equivalent to the given ontology up to the symbols in Σ, for which the truth assignments can be arbitrary. 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, in 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 $\mathcal{O}'$ 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

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 the different polarities of A to generate the ordering $\succ$.
3. Guided by $\succ$, apply the DSQEL to produce the ontology $\mathcal{O}'$ (with clauses interpreted in the obvious way as inclusions).
4. Apply simplification rules to $\mathcal{O}'$, if needed, and return the resultant ontology. If it contains only the symbols in $\text{sig}(\mathcal{O}')\backslash\Sigma$ the method was successful.

Fig. 1. The phases in the basic DSQEL routine

elimination techniques [8] 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.

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 DSQEL Forgetting Method

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. Once receiving the input ontology $\mathcal{O}$ 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 N 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 a *forgetting order* $\succ$ 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 forgetting order $\succ$. 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 continues. This means Phase 2 and Phase 3 will be executed alternately

and repeatedly with recomputed forgetting orders. If the largest current concept symbol 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 forgetting orders of concept symbols applied may lead to different but equivalent results. The intermediary results as well as the final result 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., the ontology returned is 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 with the target language being $\mathcal{ALCOI}$, this is to be expected. Concept forgetting is already not always computable for the description logic $\mathcal{EL}$ [10]. We also note that when concept symbols cannot be eliminated by our method this does not necessarily mean that 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 results for the MSQEL procedure proved in [26], since the calculus given in the next section is correct and terminates, and all adaptations of MSQEL to DSQEL preserve logical equivalence.

5 The DSQEL Forgetting Calculus

The order in which the non-base symbols are eliminated is determined by the forgetting order $\succ$ computed in Phase 2 of the DSQEL algorithm. (Formally, $\succ$ may be any irreflexive, transitive relation on the non-base symbols to be eliminated; no additional conditions need to be imposed.) 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*) wrt. a concept symbol A iff all occurrences of A in C are *positive* (*negative*). A set N of concepts 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

Ackermann:

$$\frac{N, C_1 \sqcup A, \ldots, C_n \sqcup A}{(N^A_{\sim C_1 \sqcup \ldots \sqcup \sim C_n})^{\neg\neg C_1, \ldots, \neg\neg C_n}_{C_1, \ldots, C_n}}$$

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 wrt. A.

Purify:

$$\frac{N}{(N^A_{\neg\top})^{\neg\neg\top}_{\top}}$$

provided:
(i) A is a non-base symbol in N, and
(ii) N is negative wrt. A.

Fig. 2. The forgetting rules

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. Like all other inference rules in the DSQEL calculus, the Ackermann rule is restricted by a set of side-conditions. 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.

The rules in Figures 3 and 4 are used to rewrite the clauses so they can be transformed into a form where either the Ackermann rule or the Purify rule is applicable. To apply the Ackermann or Purify rule, the clauses need to be in A-reduced form, where A is the largest non-base symbol. We say a clause is in *A-reduced form* if it is either negative in A or it has the form $A \sqcup C$, where C does not include any occurrences of A. A set of clauses is in *A-reduced form* if every clause is in A reduced form.

A set of clauses is transformed into A-reduced form, by repeatedly applying the Surfacing rule, the Skolemization rule, the Clausify rule and the Case Splitting rule to clauses containing positive occurrences of A that are not already in A-reduced form.

The Surfacing rule equivalently transforms a clause where the largest non-base symbol occurs positively below a universal restriction operator so that these occurrences pass up to levels closer to the top level of the clause. 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 $C \sqcup \neg(D_1 \sqcup \ldots \sqcup D_n)$ into a set of clauses. The Sign Switch-

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

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

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: $$\frac{N, C \sqcup \neg(D_1 \sqcup \ldots \sqcup D_n)}{N, C \sqcup \sim D_1, \ldots, C \sqcup \sim D_n}$$

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

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

provided: (i) N is closed wrt. 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

ing 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 wrt. this non-base symbol and the Sign Switching rule has not been performed for this non-base symbol before.

A novel aspect of the DSQEL calculus is the *Case Splitting* rule given in Figure 4. It splits a clause of the form $\neg a \sqcup C_1 \sqcup \ldots \sqcup C_n$ into smaller subclauses $\neg a \sqcup C_1, \ldots, \neg a \sqcup C_n$. A single clause $\neg a \sqcup C_i$, together with N, forms a *case*. The original clause means that a belongs to at least one of the disjuncts C_i ($1 \leq i \leq n$). The benefits of the Case Splitting rule are twofold. On the one hand, it 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 then able to handle. On the other hand, our tests show that it reduces the search space and increases the success rate, because the transformation to A-reduced form is easier in the cases, in which the clauses are smaller.

As the purpose of the rewriting rules is finding A-reduced forms and letting the two forgetting rules become applicable, it is not difficult to see that the rewriting rules, excluding the Sign Switching rule, should be performed before the forgetting rules. The Sign Switching rule is the exception because, as men-

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

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

Fig. 4. The Case Splitting rule

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

tioned earlier in this section, it is performed only when no other inference rules are applicable. Reruns of the rewriting rules, except for Sign Switching, are required since once a rule is applied, another rule that was previously unable to be applied may become applicable now. The rerun will continue until the clauses are not changed by any of the Clausify rule, the Surfacing rule, or the Skolemization rule. If either of the two forgetting rules becomes applicable, they are immediately applied.

We also introduced several simplification rules to transform more expressions so that inference rules become applicable, and in order to keep expressions in simpler forms for efficiency. 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 clauses of a particular pattern can be simplified, which other forgetting and second-order quantifier elimination methods cannot handle. The rules have the form $N[C]/N[D]$ and have the effect of replacing an occurrence of a subexpression C in some clause in N by the expression D.

6 Calculating the Forgetting Order

In a forgetting problem, the forgetting order is the order in which the non-base symbols are forgotten. Given n non-base symbols, in the worst case there are $n!$ possible orderings for the forgetting procedure to follow. Selecting a good

forgetting order is important for the efficiency of the forgetting method and the success rate (when a timeout is used). This is best illustrated with an example.

We first show that the forgetting order matters. Consider the following ontology in clause form and suppose the forgetting order is $A \succ B$.

$$\begin{aligned} \star\, 1.&\ (\forall R^-.\neg a) \sqcup (\neg\forall R.A) \sqcup B \\ 2.&\ (\forall R^-.\forall R^-.\neg a) \sqcup (\forall R.A) \sqcup \neg B \\ 3.&\ (\forall R^-.\neg a) \sqcup \neg\forall R.\forall R.\neg A \end{aligned}$$

Since A is the largest non-base symbol the initial aim is to bring the clauses into A-reduced form and then eliminate A with one of the forgetting rules. The starred clause (Clause 1) is negative wrt. the current non-base symbol A; all others contain positive occurrences of A. At this point, no rules in the DSQEL calculus (excluding the Sign Switching rule) can be applied to transform Clause 3 to reduced form wrt. A. The Sign Switching rule is unable to change the situation either in this case. However, changing the forgetting order to $B \succ A$ opens a survival window for the problem.

Assume now the forgetting order is $B \succ A$,

$$\begin{aligned} 1.&\ (\forall R^-.\neg a) \sqcup (\neg\forall R.A) \sqcup B \\ \star\, 2.&\ (\forall R^-.\forall R^-.\neg a) \sqcup (\forall R.A) \sqcup \neg B \\ 3.&\ (\forall R^-.\neg a) \sqcup \neg\forall R.\forall R.\neg A \end{aligned}$$

The aim is to eliminate B first. Clause 2 is negative wrt. B and Clause 1 has a positive occurrence of B. Applying the Ackermann rule to Clauses 1 and 2 leads to Clause 4, which proves to be a tautology and thus can be deleted.

$$\begin{aligned} 4.&\ (\forall R^-.\forall R^-.\neg a) \sqcup \underline{(\forall R.A)} \sqcup (\forall R^-.\neg a) \sqcup \underline{(\neg\forall R.A)} && \text{1 into 2, Acker.} \\ 5.&\ (\forall R^-.\neg a) \sqcup \neg\forall R.\forall R.\bot && \text{3, Sign Sw. \& Purify} \end{aligned}$$

What remains is Clause 3, from which A can be forgotten by applying the Sign Switching and Purify rules, which produces Clause 5. The method terminates successfully, returning $\neg a \sqcup \forall R.\neg\forall R.\forall R.\bot$ after simplifying Clause 5.

A good forgetting order allows non-base symbols to be forgotten as quickly as possible, however it does not generally guarantee success of the procedure. In this case, another ordering will be used, and the success of forgetting will be pursued until all possible orderings have been attempted.

Our implementation of the DSQEL calculus involves a heuristic method to calculate forgetting orders with increased chances of quick and successful elimination of the non-base symbols. The method exploits polarity counts of the non-base symbols. Given n non-base symbols, we first count the number of positive and negative occurrences of each symbol and represent the results as pairs. We then choose the smaller value of each pair as their *actual counts*. These actual counts are sorted into ascending order, which is then taken as the forgetting order to be used. If the actual counts of some of the non-base symbols

Table 1. Results of polarity counting for each non-base symbol

Non-Base Symbol	A_1	A_2	A_3	A_4	A_5	A_6
No. of Positive Occurrences	6	3	**2**	**0**	**2**	**1**
No. of Negative Occurrences	**1**	**2**	3	8	4	1

are identical, we compare the counts for their opposite polarity. If these are the same, the positive counts have higher priority than the negative ones. Generally, symbols with lower counts are selected to be forgotten before symbols with higher counts. Note that if we fail to forget a symbol in an ordering, we simply go to the next symbol. Once a symbol has been forgotten, the forgetting order will be recomputed since the forgetting of a non-base symbol might affect the occurrences of the remaining non-base symbols. The reason why counting is not conducted wrt. a particular polarity (either positive or negative) of the non-base symbols, is the Sign Switching rule, with which we can change the polarity of the occurrences of a particular non-base symbol.

We use an example to illustrate the operation of this heuristic method. Suppose the frequency analysis of the polarity counting reveals the pairs as listed in Table 1. The numbers in bold indicate the smaller values of each pair, i.e., the actual counts. Thus the forgetting order $\succ$ calculated is $A_4 \succ A_6 \succ A_1 \succ A_3 \succ A_2 \succ A_5$.

In the previous example the frequency analysis computes the order $B \succ A$, which immediately leads to success, without requiring another round.

7 Empirical Results

We implemented our forgetting method in Java using the OWL API, fully realising every aspect of the inference rules and the simplification rules in DSQEL. An important part of the implementation is the calculation of the forgetting order of the non-base symbols based on a frequency analysis as described in the previous section. In order to evaluate how the DSQEL method behaves on real-life ontologies, we tested the system on 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 in applications, we conducted experiments where 10%, 30%, and 50% of the concept symbols in the ontologies were forgotten.

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

Table 2. Forgetting 10%, 30%, and 50% of the concept symbols in ontologies

Input		Experiment		Results		
Axioms Avg.	Symbols Avg.	%	Analysis	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%

The DSQEL algorithm processed each non-base symbol and counted the number of their positive and negative occurrences. Based on these counts, a forgetting order was generated by the heuristic algorithm. In order to see how the forgetting order affected the performance of the method, we ran two sets of experiments, where we omitted the frequency analysis for determining the forgetting order for one set, and applied the analysis to the other set. Without the frequency analysis, the symbols were forgotten in the order as returned by an OWL API function that gets all concept symbols in the ontology.

The evaluation results obtained from 10%, 30%, 50% of the concept symbols in the ontologies, without and with the frequency analysis for determining the forgetting order, are shown in Table 2. It can be seen that, the frequency analysis led to a decrease in the average duration of the runs of every experiment, which means that it took less time to complete the same task than when the frequency analysis was not performed. It is evident from the last two rows in the table that basing the forgetting order to the frequency analysis has brought a positive effect on the overall success rate (increase by 8.1%) and the number of timeouts (decrease by 5.9%).

To show the difficulty of the forgetting problem, and how well our method behaves, we considered the extreme scenario of forgetting all concept symbols from each ontology. In this case, the selected ontologies (which were the same as used in the previous experiment) were divided first into three groups and each of them contained the ontologies with the numbers of concept symbols ranging from 1 to 1000, from 1001 to 4000, and more than 4000, respectively, in order to explore how forgetting was influenced by the number of the concept symbols that the ontologies contained. The other specifications remained the same, unless otherwise stated.

The results of the evaluation with and without the frequency analysis are shown in Tables 3 and 4, respectively. As with the results of the previous evaluation, the analysis of the forgetting order made a significant difference to the overall success rate (increase by 13.7%) and the number of timeouts (decrease by 14%). What can also be observed is that for smaller ontologies with fewer concept symbols, there were fewer timeouts and the success rate of the method was higher.

Table 3. Forgetting all concept symbols with frequency analysis

Input			Results		
Corpora	Axioms Avg.	Concept Symbols	Timeouts	Duration Avg.	Success Rate
1 – 1000	652	258	1.3%	0.869 sec.	96.4%
1001 – 4000	3091	2021	3.8%	9.148 sec.	92.5%
$\geq$ 4001	6506	6048	13.3%	29.898 sec.	86.7%
Total	1407	876	2.4%	4.352 sec.	95.2%

Table 4. Forgetting all concept symbols without frequency analysis

Input			Results		
Corpora	Axioms Avg.	Concept Symbols	Timeouts	Duration Avg.	Success Rate
1 – 1000	652	258	9.8%	4.589 sec.	87.9%
1001 – 4000	3091	2021	28.3%	51.400 sec.	67.9%
$\geq$ 4001	6506	6048	60.0%	224.133 sec.	40.0%
Total	1407	876	16.4%	24.363 sec.	81.5%

Evaluations of more aspects are being conducted at the moment. These evaluations are focussed on measuring the difference that case splitting and simplification make to the behaviour of the DSQEL calculus, and how our method compares to the related methods of SCAN [7], DLS [5], DLS* [6], SQEMA [3], MSQEL [26], and LETHE [16] in terms of success rate and efficiency (duration and number of timeouts).

8 Related Work

Probably the most important early work on the elimination of second-order quantifiers is that of Ackermann [1] in the nineteen-thirties and forties. Only in 1992, the first practical algorithm, called SCAN, was developed by Gabbay and Ohlbach [7]. 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, for 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. [3] 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, with the use of elimination orders, and the presentation of second-order quantifier elimination as an abstract calculus, as key novelties.

Investigation of forgetting as uniform interpolation in more expressive description logics was started in [29] and [21]. The first approach to compute uniform 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 2EXPTIME-complete and in the worst case, the size of uniform interpolants is triple exponential wrt. the size of the original TBox. The first goal-oriented method based on 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 $\mathcal{ALC}$-TBoxes, 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].

9 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 second-order quantifier elimination techniques to the area of knowledge representation, where description logics provide important logical formalisms.

We have implemented a prototype system of our forgetting method, fully realising the DSQEL calculus. The evaluation results have confirmed 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 forgetting order which the method follows. Overall, the results showed promising and very good success rates for concept symbol forgetting for our method.

Optimisations to both the calculus and the implementation are 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 better heuristics for computing better forgetting orders of the non-base symbols.

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

References

1. Ackermann, W.: Untersuchungen *ü*ber das Eliminationsproblem der mathematischen Logik. Mathematische Annalen **110**(1), 390–413 (1935)
2. Bachmair, L., Ganzinger, H., Waldmann, U.: Refutational theorem proving for hierarchic first-order theories. Applicable Algebra in Engineering, Communication and Computing **5**(3–4), 193–212 (1994)
3. Conradie, W., Goranko, V., Vakarelov, D.: Algorithmic correspondence and completeness in modal logic. I. The core algorithm SQEMA. Logical Methods in Computer Science **2**(1) (2006)
4. Grau, B.C., Motik, B.: Reasoning over ontologies with hidden content: The importby-query approach. Journal of Artificial Intelligence Research **45**, 197–255 (2012)
5. Doherty, P., Łukaszewicz, W., Szałas, A.: Computing circumscription revisited: A reduction algorithm. Journal of Automated Reasoning **18**(3), 297–336 (1997)
6. Doherty, P., Łukaszewicz, W., Szałas, A.: General domain circumscription and its effective reductions. Fundamenta Informaticae **36**(1), 23–55 (1998)
7. Gabbay, D.M., Ohlbach, H.J.: Quantifier elimination in second-order predicate logic. In: Principles of Knowledge Representation and Reasoning (KR92), pp. 425–435. Morgan Kaufmann (1992)
8. Gabbay, D.M., Schmidt, R.A., Szałas, A.: Second Order Quantifier Elimination: Foundations, Computational Aspects and Applications. College Publications (2008)
9. Goranko, V., Hustadt, U., Schmidt, R.A., Vakarelov, D.: SCAN is complete for all Sahlqvist formulae. In: Berghammer, R., Möller, B., Struth, G. (eds.) RelMiCS 2003. LNCS, vol. 3051, pp. 149–162. Springer, Heidelberg (2004)
10. Konev, B., Lutz, C., Walther, D., Wolter, F.: Model-theoretic inseparability and modularity of description logic ontologies. Artificial Intelligence **203**, 66–103 (2013)
11. Konev, B., Walther, D., Wolter, F.: Forgetting and uniform interpolation in extensions of the description logic $\mathcal{EL}$. In: Proceedings of the 22nd International Workshop on Description Logics (DL 2009). CEUR Workshop Proceedings, vol. 477. CEUR-WS.org (2009)
12. Koopmann, P., Schmidt, R.A.: Forgetting concept and role symbols in $\mathcal{ALCH}$-ontologies. In: McMillan, K., Middeldorp, A., Voronkov, A. (eds.) LPAR 2013. LNCS, vol. 8312, pp. 552–567. Springer, Heidelberg (2013)
13. Koopmann, P., Schmidt, R.A.: Implementation and evaluation of forgetting in ALC-ontologies. In: Proceedings of the 7th International Workshop on Modular Ontologies (WoMo 2013). CEUR Workshop Proceedings, vol. 1081, pp. 1–12. CEUR-WS.org (2013)
14. Koopmann, P., Schmidt, R.A.: Uniform interpolation of $\mathcal{ALC}$-ontologies using fixpoints. In: Fontaine, P., Ringeissen, C., Schmidt, R.A. (eds.) FroCoS 2013. LNCS, vol. 8152, pp. 87–102. Springer, Heidelberg (2013)

15. Koopmann, P., Schmidt, R.A.: Count and forget: uniform interpolation of $\mathcal{SHQ}$-ontologies. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) IJCAR 2014. LNCS, vol. 8562, pp. 434–448. Springer, Heidelberg (2014)
16. Koopmann, P., Schmidt, R.A.: LETHE: A saturation-based tool for non-classical reasoning (2015). Manuscript, submitted
17. Koopmann, P., Schmidt, R.A.: Saturated-based forgetting in the description logic SIF. In: Proceedings of the 28th International Workshop on Description Logics (DL 2015). CEUR Workshop Proceedings, vol. 1350. CEUR-WS.org (2015)
18. Koopmann, P., Schmidt, R.A.: Uniform interpolation and forgetting for $\mathcal{ALC}$-ontologies with ABoxes. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, pp. 175–181. AAAI Press (2015)
19. Ludwig, M., Konev, B.: Towards practical uniform interpolation and forgetting for $\mathcal{ALC}$ TBoxes. In: Proceedings of the 26th International Workshop on Description Logics (DL 2013). CEUR Workshop Proceedings, vol. 1014, pp. 377–389. CEUR-WS.org (2013)
20. Lutz, C., Seylan, I., Wolter, F.: An automata-theoretic approach to uniform interpolation and approximation in the description logic $\mathcal{EL}$. In: Principles of Knowledge Representation and Reasoning: KR 2012, pp. 286–297. AAAI Press (2012)
21. Lutz, C., Wolter, F.: Foundations for uniform interpolation and forgetting in expressive description logics. In: Proceedings of IJCAI 2011, pp. 989–995. IJCAI/AAAI (2011)
22. Nikitina, N.: Forgetting in general $\mathcal{EL}$ terminologies. In: Proceedings of the 24th International Workshop on Description Logics (DL 2011). CEUR Workshop Proceedings, vol. 745. CEUR-WS.org (2011)
23. Nonnengart, A., Szałas, A.: A fixpoint approach to second-order quantifier elimination with applications to correspondence theory. In: Orlowska, E. (ed.) Logic at Work: Essays Dedicated to the Memory of Helena Rasiowa, pp. 307–328. Springer (1999)
24. Ohlbach, H.J.: SCAN–elimination of predicate quantifiers. In: McRobbie, M.A., Slaney, J.K. (eds.) CADE 1996. LNCS, vol. 1104, pp. 161–165. Springer, Heidelberg (1996)
25. Schild, K.: A correspondence theory for terminological logics: preliminary report. In: Proceedings of IJCAI 1991, pp. 466–471. Morgan Kaufmann (1991)
26. Schmidt, R.A.: The Ackermann approach for modal logic, correspondence theory and second-order reduction. Journal of Applied Logic **10**(1), 52–74 (2012)
27. Szałas, A.: On the correspondence between modal and classical logic: An automated approach. Journal of Logic and Computation **3**, 605–620 (1993)
28. Szałas, A.: Second-order reasoning in description logics. Journal of Applied Non-Classical Logics **16**(3–4), 517–530 (2006)
29. Wang, K., Wang, Z., Topor, R., Pan, J.Z., Antoniou, G.: Concept and role forgetting in $\mathcal{ALC}$ ontologies. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 666–681. Springer, Heidelberg (2009)
30. Wang, K., Wang, Z., Topor, R., Pan, J.Z., Antoniou, G.: Eliminating concepts and roles from ontologies in expressive description logics. Computational Intelligence **30**(2), 205–232 (2014)
31. Wang, Z., Wang, K., Topor, R., Pan, J.Z.: Forgetting concepts in DL-Lite. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 245–257. Springer, Heidelberg (2008)