Forgetting Concept and Role Symbols in $\mathcal{ALCH}$-Ontologies

Patrick Koopmann and Renate A. Schmidt

The University of Manchester, UK
{koopmanp, schmidt}@cs.man.ac.uk

Abstract. We develop a resolution-based method for forgetting concept and role symbols in $\mathcal{ALCH}$ ontologies, or for computing uniform interpolants in $\mathcal{ALCH}$. Uniform interpolants use only a restricted set of symbols, while preserving logical consequences of the original ontology involving these symbols. While recent work towards practical methods for uniform interpolation in expressive description logics limits attention to forgetting concept symbols, we believe most applications would benefit from the possibility to forget both role and concept symbols. We focus on the description logic $\mathcal{ALCH}$, which allows for the formalisation of role hierarchies. Our approach is based on a recently developed resolution-based calculus for forgetting concept symbols in $\mathcal{ALC}$ ontologies, which we extend by redundancy elimination techniques to make it practical for larger ontologies. Experiments on $\mathcal{ALCH}$ fragments of real life ontologies suggest that our method is applicable in a lot of real-life applications.

1 Introduction

Ontologies model a domain of interest using description logics by describing the vocabulary of this domain in terms of roles and concepts. Reflecting the different applications and contexts in which ontologies are used, ontologies are modelled using different description logics that vary in expressivity and complexities of common reasoning tasks. In the development of complex ontologies, it is often desirable to restrict the vocabulary of an ontology to a smaller set of symbols. Uniform interpolation, also known as forgetting, establishes this by constructing a new ontology that only uses a predefined set of symbols, such that all logical consequences of the original ontology using these symbols are preserved. Examples where this is useful are: (i) *Ontology Reuse.* When constructing larger ontologies, it can be useful to reuse parts from existing ontologies. Using uniform interpolation, one can restrict the vocabulary of the reused ontology to the symbols that are known and interesting for the new application. (ii) *Predicate Hiding.* When publishing or sharing an ontology, it is often desirable to hide confidential parts from the ontology, without affecting the intended meaning of the remaining vocabulary [5]. (iii) *Exhibiting Hidden Relations.* Relations between symbols are often stated indirectly in an ontology and only become visible through the use of reasoners. With increased complexity of the ontology, this makes it hard to get a deeper understanding of the ontology and to maintain ontology changes.

The uniform interpolant over a set of symbols makes the relations between these symbols explicit. (iv) *Logical difference.* In the development of evolving ontologies, it is important for ontology engineers to ensure that modifications do not interfere with the meaning of existing terms. This can be achieved by computing the uniform interpolants of two versions of an ontology over the common set of used symbols, or over a set of symbols under consideration, and checking whether the resulting ontologies are equivalent [12].

Uniform interpolation has been extensively investigated for simpler description logics such as $\mathcal{EL}$ or DL-Lite [8, 22, 15, 13]. Recently, practical algorithms for forgetting concept symbols in the more expressive description logic $\mathcal{ALC}$ have been developed [12, 11]. In this paper, we investigate forgetting of role symbols as well, and supplement earlier presented work with optimisation techniques to make it practical on larger ontologies. Since roles play a larger role in this context, we focus on the description logic $\mathcal{ALCH}$, which extends $\mathcal{ALC}$ with role hierarchies. It is known that already in the description logic $\mathcal{ALC}$ uniform interpolants cannot be finitely expressed in the language of the logic [14]. This also applies to $\mathcal{ALCH}$. For this reason our method computes uniform interpolants for the target language $\mathcal{ALCH}\mu$, which extends $\mathcal{ALCH}$ with fixpoint operators, thus enabling us to always compute finite representations. If fixpoints are used in the uniform interpolant, it is possible to represent it in $\mathcal{ALCH}$ by extending the signature of the interpolant.

Our work is based on a recently developed method for forgetting concept symbols in $\mathcal{ALC}$-ontologies [11]. The method is based on a resolution-based decision procedure for $\mathcal{ALCH}$. In order to analyse the practicality of our approach, we undertake an experimental evaluation on $\mathcal{ALCH}$-fragments of a set of real-life ontologies. The results suggest that uniform interpolation can be used for the presented applications in a lot of real-life situations.

Proofs of all theorems can be found in the accompanying technical report [9].

2 Preliminaries

Let N_c, N_r be two disjoint sets of *concept symbols* and *role symbols. Concepts* in $\mathcal{ALCH}$ are of the following form:

$$\bot \mid \top \mid A \mid \neg C \mid C \sqcup D \mid C \sqcap D \mid \exists r.C \mid \forall r.C,$$

where $A \in N_c$, $r \in N_r$ and C and D are arbitrary concepts. $\top$, $C \sqcap D$ and $\forall r.C$ are defined as abbreviations: $\top$ stands for $\neg\bot$, $C \sqcap D$ for $\neg(\neg C \sqcup \neg D)$ and $\forall r.C$ for $\neg\exists r.\neg C$.

A TBox is a set of *concept axioms* of the forms $C \sqsubseteq D$ (*concept inclusion*) and $C \equiv D$ (*concept equivalence*), where C and D are concepts. An RBox is a set of *role axioms* of the form $r \sqsubseteq s$ (*role inclusion*) and $r \equiv s$ (*role equivalence*), where r and s are role symbols. $C \equiv D$ is a short-hand for the two concept axioms $C \sqsubseteq D$ and $D \sqsubseteq C$, and $r \equiv s$ is a short-hand for the two role axioms $r \sqsubseteq s$ and $s \sqsubseteq r$. We assume an ontology consists of a TBox and an

RBox. Given an ontology $\mathcal{O}$, we define $\sqsubseteq_{\mathcal{O}}$ to be the reflexive transitive closure of the role inclusions in $\mathcal{O}$.

The semantics of $\mathcal{ALCH}$ is defined as follows. An *interpretation* is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where the *domain* $\Delta^{\mathcal{I}}$ is a nonempty set and the *interpretation function* $\cdot^{\mathcal{I}}$ assigns to each concept symbol $A \in N_c$ a subset of $\Delta^{\mathcal{I}}$ and to each role symbol $r \in N_r$ a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is extended to concepts as follows.

$$\bot^{\mathcal{I}} := \emptyset \qquad (\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \qquad (C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}$$
$$(\exists r.C)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid \exists y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$$

A concept inclusion $C \sqsubseteq D$ is *true* in an interpretation $\mathcal{I}$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. $\mathcal{I}$ is model of a TBox $\mathcal{T}$ if all concept inclusions in $\mathcal{T}$ are true in $\mathcal{I}$. A TBox $\mathcal{T}$ is *satisfiable* if there exists a model for $\mathcal{T}$, otherwise it is *unsatisfiable.* $\mathcal{T} \models C \sqsubseteq D$ holds iff in every model of $\mathcal{T}$ we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Two TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ are *equi-satisfiable* if every model of $\mathcal{T}_1$ can be extended to a model of $\mathcal{T}_2$, and vice versa. The definitions of truth, model, satisfiability and equi-satisfiability extend to roles, RBoxes and ontologies in a similar way. Observe that $\mathcal{O} \models r \sqsubseteq s$ iff $r \sqsubseteq_{\mathcal{O}} s$.

In order to define $\mathcal{ALCH}\mu$, we extend the language with a set N_v of *concept variables.* $\mathcal{ALCH}\mu$ extends $\mathcal{ALCH}$ with concepts of the form $\mu X.C$ and $\nu X.C$, where $X \in N_v$, and C is a concept in which X occurs as a concept symbol only positively (under an even number of negations). $\mu X.C$ denotes the *least fixpoint* of C on X and $\nu X.C$ the *greatest fixpoint.*

A concept variable X is *bound* if it occurs in the scope C of a fixpoint expression $\mu X.C$ or $\nu X.C$. Otherwise it is *free.* A concept is *closed* if it does not contain any free variables. Axioms in $\mathcal{ALCH}\mu$ are of the form $C \sqsubseteq D$ and $C \equiv D$, where C and D are closed concepts.

Following [2], we define the semantics of fixpoint expressions. Let $\mathcal{V}$ be an *assignment function* that maps concept variables to subsets of $\Delta^{\mathcal{I}}$. $\mathcal{V}[X \mapsto W]$ denotes $\mathcal{V}$ modified by setting $\mathcal{V}(X) = W$. $C^{\mathcal{I},\mathcal{V}}$ is the interpretation of C taking into account this assignment, and when $\mathcal{V}$ is defined for all variables in C, $C^{\mathcal{I},\mathcal{V}} = C^{\mathcal{I}}$. The semantics of fixpoint concepts is defined as follows:

$$(\mu X.C)^{\mathcal{I},\mathcal{V}} := \bigcap \{W \subseteq \Delta^{\mathcal{I}} \mid C^{\mathcal{I},\mathcal{V}[X \mapsto W]} \subseteq W\}$$
$$(\nu X.C)^{\mathcal{I},\mathcal{V}} := \bigcup \{W \subseteq \Delta^{\mathcal{I}} \mid W \subseteq C^{\mathcal{I},\mathcal{V}[X \mapsto W]}\}.$$

The *size* of an ($\mathcal{ALCH}$- or $\mathcal{ALCH}\mu$-)axiom is defined recursively as follows: *size*$(A) = 1$, where A is a concept symbol, *size*$(\neg C) =$ *size*$(C) + 1$, *size*$(\exists r.C) =$ *size*$(\forall r.C) =$ *size*$(C) + 2$, *size*$(C \sqcup D) =$ *size*$(C \sqcap D) =$ *size*$(C) +$ *size*$(D) + 1$, *size*$(\mu X.C) =$ *size*$(\nu X.C) =$ *size*$(C) + 2$ and *size*$(C \sqsubseteq D) =$ *size*$(C \equiv D) =$ *size*$(C) +$ *size*$(D) + 1$.

A *signature* Σ is a subset of $N_c \cup N_r$. *sig*(E) denotes the concept and role symbols occurring in E, where E ranges over concept descriptions, axioms, TBoxes, RBoxes and ontologies. Given two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ and a signature Σ, we say $\mathcal{O}_1$ and $\mathcal{O}_2$ are *Σ-inseparable*, in symbols $\mathcal{O}_1 \equiv_{\Sigma} \mathcal{O}_2$, iff for every concept or role inclusion α with $sig(\alpha) \subseteq \Sigma$, $\mathcal{O}_1 \models \alpha$ implies $\mathcal{O}_2 \models \alpha$, and vice versa. Given an ontology $\mathcal{O}$ and a signature Σ, $\mathcal{O}'$ is a *uniform interpolant of* $\mathcal{O}$ if $sig(\mathcal{O}') \subseteq \Sigma$ and $\mathcal{O} \equiv_\Sigma \mathcal{O}'$. From this definition, it follows that uniform interpolants for a given ontology and signature are unique modulo logical equivalence. For a given ontology $\mathcal{O}$ and signature Σ, we will therefore speak of *the* uniform interpolant and denote it by $\mathcal{O}^\Sigma$. Given an ontology $\mathcal{O}$ and a concept or role symbol σ, the result of *forgetting σ in $\mathcal{O}$*, denoted by $\mathcal{O}^{-\sigma}$, is the uniform interpolant $\mathcal{O}^\Sigma$, where $\Sigma = sig(\mathcal{O}) \setminus \{\sigma\}$.

3 Overview of the Method

We reduce the problem of computing uniform interpolants to the problem of forgetting single symbols. In order to compute the uniform interpolant for any signature Σ, we forget each symbol in $sig(\mathcal{O}) \setminus \Sigma$ one by one. The method for computing $\mathcal{O}^{-\sigma}$, where σ is either a role or a concept symbol, consists of three phases:

Phase 1: Eliminate the symbol using a resolution-based calculus, obtaining $\mathcal{O}' = \mathcal{F}^\sigma_{\mathcal{ALCH}}(\mathcal{O})$.
Phase 2: Eliminate the newly introduced symbols, obtaining $\mathcal{O}^{-\sigma} = \mathcal{F}_D(\mathcal{O}')$.
Phase 3: Apply simplifications and represent clauses as proper concept inclusions.

Central to the method is a new resolution-based calculus which works on a structural transformation based normal form. The calculus is described in Section 4. Depending on whether the symbol to be forgotten is a role or a concept symbol, in Phase 1 a different method based on this resolution calculus is used to derive consequences on the selected symbol. This is described in Section 5. The result is a finitely bounded set N of axioms such that $\sigma \notin sig(N)$ and $N \equiv_\Sigma \mathcal{O}$ for $\Sigma = sig(\mathcal{O}) \setminus \{\sigma\}$, but N may use new symbols due to structural transformation. These symbols, called definers, all occur in a form that allows for elimination in a simple and uniform way, following a known principle first presented in [16]. This is performed in Phase 2 and described in Section 6. Depending on whether the aim is to compute a representation in $\mathcal{ALCH}\mu$ or in $\mathcal{ALCH}$, the result may involve fixpoint operators or extend the signature of the original ontology.

After Phase 2, the uniform interpolant is already computed, but we add a third phase that makes the resulting ontology more accessible by applying several equivalence-preserving transformations. The following main theorem of this paper states the correctness of the method.

Theorem 1. *For any $\mathcal{ALCH}$-ontology $\mathcal{O}$ and any symbol σ, our method terminates and returns the uniform interpolant of $\mathcal{O}$ over $sig(\mathcal{O}) \setminus \{\sigma\}$ in $\mathcal{ALCH}\mu$. If the result does not make use of a fixpoint operator, it is the uniform interpolant of $\mathcal{O}$ over $sig(\mathcal{O}) \setminus \{\sigma\}$ in $\mathcal{ALCH}$.*

4 The Underlying Calculus

Our method for forgetting concept and role symbols is based on a resolution calculus $\mathcal{R}_{\mathcal{ALCH}}$ which provides a decision procedure for $\mathcal{ALCH}$-ontology satisfiability. The calculus extends a calculus introduced in [11] by incorporating the role hierarchy. In order to make our method practical for larger ontologies, we extend $\mathcal{R}_{\mathcal{ALCH}}$ with redundancy elimination techniques, resulting in the calculus $\mathcal{R}^s_{\mathcal{ALCH}}$.

Both calculi operate on sets of clauses, which are defined as follows. Let $N_D \subseteq N_c$ be a set of *definer symbols* or *definers*, which do not occur in any input ontology.

Definition 1. *An* $\mathcal{ALCH}$-literal *is a concept description of the form A, $\neg A$, $\forall r.D$ or $\exists r.D$, where A is a concept symbol, r a role symbol and D is a definer.*

A TBox is in $\mathcal{ALCH}$-conjunctive normal form *if every axiom is of the form $\top \sqsubseteq L_1 \sqcup ... \sqcup L_n$, where each L_i is an $\mathcal{ALCH}$-literal. The right part of such a concept inclusion is called* $\mathcal{ALCH}$-clause. *In the following we assume $\mathcal{ALCH}$-clauses are represented as sets of literals (this means no clause contains the same literal more than once and the order of the literals does not matter). The empty clause is denoted by $\bot$ and represents a contradiction.*

For our method it is crucial that any $\mathcal{ALCH}$-TBox is transformed into an equi-satisfiable TBox in $\mathcal{ALCH}$-conjunctive normal form using structural transformation as follows. First the input TBox is transformed into negation normal form. Then every concept C that occurs immediately below a role restriction is replaced by a definer D, and we add the axiom $D \sqsubseteq C$ for each such subconcept. The resulting TBox does not contain any nested role restrictions and can be brought into $\mathcal{ALCH}$-conjunctive normal form by applying standard CNF-transformation techniques. For an ontology $\mathcal{O}$, let *clauses*$(\mathcal{O})$ refer to the set of clauses generated in this way from the TBox of $\mathcal{O}$.

The calculus $\mathcal{R}_{\mathcal{ALCH}}$ uses the rules shown in Figure 1. Since the normal form has to be preserved, the role propagation rule may require the introduction of a new definer symbol D_3 representing the conjunction of the definers D_1 and D_2 occurring in the premises. This is done by adding new clauses $\neg D_3 \sqcup D_1$ and $\neg D_3 \sqcup D_2$ to the clause set. Observe that the resolution rule also applies to definer literals. This way for each pair of clauses $\neg D_1 \sqcup C_1$ and $\neg D_2 \sqcup C_2$ we derive the clauses $\neg D_3 \sqcup C_1$ and $\neg D_3 \sqcup C_2$, for which the side conditions of the rules are satisfied.

In order to ensure termination, it is necessary to reuse definers whenever possible. For this we define an identification function for introduced definers, that identifies definers with the context from which they have been created, and whose range is finitely bounded. The function $id(D)$ is defined as follows. (i) If D is introduced by the initial normal form transformation, then $id(D) = \{D\}$. (ii) If D is required by the role propagation rule and the respective role restrictions are $\forall s.D_1$ and $\mathsf{Q}r.D_2$, then $id(D) = id(D_1) \cup id(D_2)$.

If the role propagation rule requires a new definer D, we first check whether a definer D' with $id(D) = id(D')$ is already present, and reuse it in this case.

Resolution:

$$\frac{C_1 \sqcup A \qquad C_2 \sqcup \neg A}{C_1 \sqcup C_2}$$

provided $C_1 \sqcup C_2$ does not contain more than one negative definer literal.

Role Propagation:

$$\frac{C_1 \sqcup \forall s.D_1 \qquad C_2 \sqcup \mathsf{Q}r.D_2}{C_1 \sqcup C_2 \sqcup \mathsf{Q}r.D_3} \quad r \sqsubseteq_{\mathcal{O}} s$$

where $\mathsf{Q} \in \{\exists, \forall\}$ and D_3 is a (possibly new) definer representing $D_1 \sqcap D_2$, provided $C_1 \sqcup C_2$ does not contain more than one negative definer literal.

Existential Role Restriction Elimination:

$$\frac{C \sqcup \exists r.D \qquad \neg D}{C}$$

Fig. 1. Rules of the calculus $\mathcal{R}_{\mathcal{ALCH}}$.

Otherwise we introduce a new definer in the way described above. Observe that the domain of id is bounded by 2^n, where n is the number of definers introduced by the initial normal form transformation. Therefore the number of clauses that can possibly be derived is limited by a double-exponential bound. We can prove:

Theorem 2. *$\mathcal{R}_{\mathcal{ALCH}}$ is sound and refutationally complete, and provides a decision procedure for $\mathcal{ALCH}$-ontology satisfiability.*

As in traditional resolution-based decision procedures, it is possible to extend the method with redundancy elimination and further simplification techniques. For this purpose, it is possible to exploit the structure imposed by the introduced definers. Note that new definers are introduced by adding clauses of the form $\neg D_1 \sqcup D_2$. $\neg D_1 \sqcup D_2$ is equivalent to the concept inclusion $D_1 \sqsubseteq D_2$. This concept inclusion can be transferred to subsumption between existential and universal role restrictions, and to subsumption between clauses.

Definition 2. *A literal l_1 is* subsumed *by a literal l_2 ($l_1 \sqsubseteq_l l_2$) if either $l_1 = l_2$ or if $l_1 = \mathsf{Q}r.D_1$ and $l_2 = \mathsf{Q}r.D_2$ for $\mathsf{Q} \in \{\exists, \forall\}$ and there is a clause $\neg D_1 \sqcup D_2$ in the current clause set. A clause C_1 is* subsumed *by a clause C_2 ($C_1 \sqsubseteq_C C_2$) if every literal $l_1 \in C_1$ is subsumed by a literal $l_2 \in C_2$. A clause C is* redundant *with respect to a clause set N, if N contains a clause C' with $C' \sqsubseteq_C C$. The* reduction *of a clause C, $red(C)$, is obtained from C by removing every literal that is subsumed by another literal in C.*

Example 1 (Subsumption and reduction). Assume D_3 represents $D_1 \sqcap D_2$, which means we have the clauses $\neg D_3 \sqcup D_1$ and $\neg D_3 \sqcup D_2$. Then $\neg A \sqcup B$ is subsumed by

Tautology deletion: $$\frac{N \cup \{C \sqcup A \sqcup \neg A\}}{N}$$

Subsumption deletion: $$\frac{N \cup \{C,\ D\}}{N \cup \{C\}}$$ provided $C \sqsubseteq_C D$

Reduction: $$\frac{N \cup \{C\}}{N \cup \{\mathit{red}(C)\}}$$

Fig. 2. Simplification rules.

$\neg A \sqcup B \sqcup C$, $\exists r.D_3$ is subsumed by $\exists r.D_1$, $\forall r.D_3 \sqcup B$ is subsumed by $\forall r.D_1 \sqcup A \sqcup B$ and $\mathit{red}(A \sqcup \exists r.D_3 \sqcup \exists r.D_2) = A \sqcup \exists r.D_2$.

In addition to subsumption and reduction, we also detect tautological clauses which contain pairs of contradictory literals. This leads to a set of simplification rules shown in Figure 2. We denote the calculus $\mathcal{R}_{\mathcal{ALCH}}$ extended with these rules by $\mathcal{R}^s_{\mathcal{ALCH}}$. It can be shown that these rules preserve soundness and refutational completeness, as stated by the following theorem.

Theorem 3. *$\mathcal{R}^s_{\mathcal{ALCH}}$ is sound and refutationally complete and provides a decision procedure for $\mathcal{ALCH}$-ontology satisfiability.*

5 Forgetting Concept and Role Symbols

In this section, we describe the methods $\mathcal{F}^A_{\mathcal{ALCH}}$ and $\mathcal{F}^r_{\mathcal{ALCH}}$ for forgetting respectively concept symbols and role symbols. Both methods are based on $\mathcal{R}^s_{\mathcal{ALCH}}$.

For any definer D, we say D is *connected to* A, if D either co-occurs with A in a clause or if D co-occurs in a clause with another definer D' that is connected to A. If the aim is to forget a concept symbol, we restrict the rules of $\mathcal{R}^s_{\mathcal{ALCH}}$ by adding the following conditions:

Resolution: A is the symbol we want to forget or a definer.
Role Propagation: D_1 and D_2 are connected to the symbol we want to forget.

For a concept symbol A, $\mathcal{F}^A_{\mathcal{ALCH}}$ denotes the calculus $\mathcal{R}^s_{\mathcal{ALCH}}$ with these modifications for A. For any ontology $\mathcal{O}$, $\mathcal{F}^A_{\mathcal{ALCH}}(\mathcal{O})$ denotes the ontology consisting of the RBox of $\mathcal{O}$ and the TBox represented by $\mathit{clauses}(\mathcal{O})$ saturated using the rules of $\mathcal{F}^A_{\mathcal{ALCH}}$, after removing all clauses containing A or positive definer literals that are not role restrictions.

Theorem 4. *Given an ontology $\mathcal{O}$, $\mathcal{F}^A_{\mathcal{ALCH}}(\mathcal{O})$ is a clausal representation of $\mathcal{O}^{-A}$, that is, $\mathcal{F}^A_{\mathcal{ALCH}}(\mathcal{O}) \equiv_\Sigma \mathcal{O}$, where $\Sigma = \mathit{sig}(\mathcal{T}) \setminus \{A\}$, and every symbol in $\mathcal{F}^A_{\mathcal{ALCH}}(\mathcal{O})$ is either a definer or in Σ.*

Role hierarchy:

$$\frac{s \sqsubseteq r \qquad r \sqsubseteq t}{s \sqsubseteq t}$$

Universal role restriction monotonicity:

$$\frac{C \sqcup \forall r.D}{C \sqcup \forall s.D} \qquad s \sqsubseteq r \in \mathcal{O}$$

Existential role restriction monotonicity:

$$\frac{C \sqcup \exists r.D}{C \sqcup \exists s.D} \qquad r \sqsubseteq s \in \mathcal{O}$$

Role restriction resolution:

$$\frac{C_0 \sqcup \forall r.D_0 \quad ... \quad C_n \sqcup \forall r.D_n \qquad C \sqcup \exists r.D}{C_0 \sqcup ... \sqcup C_n \sqcup C} \qquad \mathcal{O} \models D_0 \sqcap ... \sqcap D_n \sqcap D \sqsubseteq \bot$$

provided (i) there is no role s with $r \sqsubseteq s \in \mathcal{O}$ and (ii) $C_0 \sqcup ... \sqcup C_n \sqcup C$ does not contain more than one negative definer literal.

Fig. 3. Rules for forgetting role symbol r.

The method $\mathcal{F}^{A}_{\mathcal{ALCH}}$ provides a focused way to forget the concept symbol A. In order to forget role symbols, a few modifications have to be made. Since role symbols also occur in the RBox of an ontology, the RBox has to be processed as well. Additionally, we need rules that compute all derivations on a selected role symbol in a focused way.

The rules in Figure 3, together with the rules of $\mathcal{R}_{\mathcal{ALCH}}$, where the resolution rule is restricted to only resolve on definer literals, constitute the method $\mathcal{F}^{r}_{\mathcal{ALCH}}$, where r is the role symbol to be forgotten. The role hierarchy rule is the only rule applied on the RBox of the input ontology, and makes implicit role inclusions around the role symbol to be forgotten explicit. The universal and existential role monotonicity rules compute inferences on the basis of clauses and RBox axioms. If there is no role inclusion $s \sqsubseteq r$, the universal role monotonicity rule cannot be applied and we have to apply role propagation on that role exhaustively in order to preserve all consequences when forgetting r.

If there is no role inclusion $r \sqsubseteq s$, we can neither apply the existential role restriction monotonicity rule nor role propagation. Instead we use the role restriction resolution rule in this case, which is similarly motivated as the resolution rule, but works on larger sets of clauses. This rule is formulated to allow the use of an external reasoner to check satisfiability of concepts (even though in theory $\mathcal{R}_{\mathcal{ALCH}}$ can be used for this as well).

Non-cyclic definer elimination:

$$\frac{\mathcal{T} \cup \{D \sqsubseteq C\}}{\mathcal{T}^{[D \mapsto C]}} \qquad \text{provided } D \notin \mathit{sig}(C)$$

Definer purification:

$$\frac{\mathcal{T}}{\mathcal{T}^{[D \mapsto \top]}} \qquad \text{provided } D \text{ occurs only positively in } \mathcal{T}$$

Cyclic definer elimination:

$$\frac{\mathcal{T} \cup \{D \sqsubseteq C[D]\}}{\mathcal{T}^{[D \mapsto \nu X.C[X]]}} \qquad \text{provided } D \in \mathit{sig}(C[D])$$

Fig. 4. Rules for eliminating definer concept symbols

For any ontology $\mathcal{O}$, we define $\mathcal{F}^r_{\mathcal{ALCH}}(\mathcal{O})$ as the ontology consisting of the RBox of $\mathcal{O}$ and the TBox represented by *clauses*($\mathcal{O}$) saturated using the rules of $\mathcal{F}^r_{\mathcal{ALCH}}$, after removing all the axioms and clauses that use the symbol r or contain a positive definer literal that is not a role restriction.

Theorem 5. *For any ontology $\mathcal{O}$, $\mathcal{F}^r_{\mathcal{ALCH}}(\mathcal{O})$ is a clausal representation of $\mathcal{O}^{-r}$.*

6 Definer Elimination

In Phase 2, the symbols introduced by the normal form transformation or the role propagation rule are eliminated. Note that we only derive clauses that contain at most one negative definer literal in Phase 1. This means we can for each definer D group the clauses of the form $\neg D \sqcup C_i$, $0 \leq i \leq n$, into a single axiom of the form $D \sqsubseteq \bigsqcap_{0 \leq i \leq n} C_i$ that can be seen as a *definition* of the definer. This definition can be used to undo the structural transformation and eliminate the remaining definers. If a definition is cyclic, we use a fixpoint operator in the result. Figure 4 shows the rules for definer elimination. The rules are justified by Ackermann's Lemma and its generalisation to the fixpoint case [1, 16].

If the output of the algorithm contains fixpoints, we can represent it in $\mathcal{ALCH}$ by extending the desired signature Σ by the cyclic definers. This is done by omitting the cyclic definer elimination rule.

7 Examples

To illustrate the presented method this section includes two examples of respectively forgetting concept and role symbols.

Example 2 (Forgetting Concept Symbols). Let $\mathcal{O}_1$ be the following ontology.

$$A \sqsubseteq B \sqcup C \qquad B \sqsubseteq \exists r.B \qquad C \sqsubseteq \forall s.\neg B \qquad r \sqsubseteq s$$

We want to compute $\mathcal{O}_1^{-B}$. We obtain the following clause set *clauses*$(\mathcal{O}_1)$.

1. $\neg A \sqcup B \sqcup C$ 2. $\neg B \sqcup \exists r.D_1$ 3. $\neg D_1 \sqcup B$
4. $\neg C \sqcup \forall s.D_2$ 5. $\neg D_2 \sqcup \neg B$

We first apply the resolution rule.

6. $\neg A \sqcup C \sqcup \exists r.D_1$ (*resolution on 2 and 1*)
7. $\neg D_1 \sqcup \exists r.D_1$ (*resolution on 2 and 3*)
8. $\neg D_2 \sqcup \neg A \sqcup C$ (*resolution on 5 and 1*)

We cannot resolve on clauses 3 and 5, since the conclusion would contain more than one negative definer literal. We can however apply role propagation on clauses 2 and 4, which makes further applications of the resolution rule possible.

$\cancel{9.}$ $\neg B \sqcup \neg C \sqcup \exists r.D_3$ (*role propagation on 2 and 4, id*$(D_3) = \{D_1, D_2\}$)
$\cancel{10.}$ $\neg D_3 \sqcup D_1$
$\cancel{11.}$ $\neg D_3 \sqcup D_2$
$\cancel{12.}$ $\neg D_3 \sqcup B$ (*resolution on 10 and 3*)
$\cancel{13.}$ $\neg D_3 \sqcup \neg B$ (*resolution on 11 and 5*)
14. $\neg D_3$ (*resolution on 12 and 13*)

Clause 14 makes clauses 10–13 become redundant, and existential role restriction elimination on Clause 9 possible.

15. $\neg B \sqcup \neg C$ (*exist. role restr. elimination on 9 and 14*)

Clause 15 makes Clause 9 become redundant. We saturate the remaining clauses.

$\cancel{16.}$ $\neg A \sqcup C \sqcup \neg C$ (*resolution on 15 and 1, tautology*)
17. $\neg D_1 \sqcup \neg C$ (*resolution on 15 and 3*)

Only clauses that do not contain B or a positive definer are included in $\mathcal{F}^B_{\mathcal{ALCH}}(\mathcal{O}_1)$. These are the clauses 4, 6, 7, 8, 14 and 17. Eliminating the definers and expressing clauses as concept inclusions (Phases 2 and 3) results in the following ontology $\mathcal{O}_1^{-B}$:

$$A \sqsubseteq C \sqcup \exists r.\nu X.(\neg C \sqcap \exists r.X) \qquad C \sqsubseteq \forall s.(\neg A \sqcup C)$$

Example 3 (Forgetting Role Symbols). Let $\mathcal{O}_2$ contain the following axioms. We want to compute $\mathcal{O}_2^{-r}$.

$$A \sqsubseteq \exists r.(A \sqcup B) \qquad B \sqsubseteq \forall r.\neg A$$
$$C \sqsubseteq \forall r.\neg B \qquad s \sqsubseteq r$$

We obtain the following clausal representation *clauses*$(\mathcal{O}_2)$:

1. $\neg A \sqcup \exists r.D_1$ 2. $\neg D_1 \sqcup A \sqcup B$
3. $\neg B \sqcup \forall r.D_2$ 4. $\neg D_2 \sqcup \neg A$
5. $\neg C \sqcup \forall r.D_3$ 6. $\neg D_3 \sqcup \neg B$

We observe that there is no role r' with $r \sqsubseteq r'$ and that $D_1 \sqcap D_2 \sqcap D_3$ is unsatisfiable, which means we can apply role restriction resolution on 1, 3 and 5:

7. $\neg A \sqcup \neg B \sqcup \neg C$ (*role restriction resolution on 1, 3 and 5*)

Furthermore, we do have a role r' with $r' \sqsubseteq r$, namely s which means we can apply universal role restriction monotonicity:

8. $\neg B \sqcup \forall s.D_2$ (*universal role restriction monotonicity on 3*)
9. $\neg C \sqcup \forall s.D_3$ (*universal role restriction monotonicity on 5*)

Omitting all clauses containing r and applying Phases 2 and 3 leads to the uniform interpolant $\mathcal{O}_2^{-r}$ consisting of the following axioms:

$$A \sqcap B \sqcap C \sqsubseteq \bot \qquad B \sqsubseteq \forall s.\neg A \qquad C \sqsubseteq \forall s.\neg B$$

8 Experimental Evaluation

In order to investigate the practicality of our approach, we implemented our method in Scala[1] using the OWL-API[2] and evaluated it on $\mathcal{ALCH}$-fragments of ontologies from the NCBO Bioportal ontology repository.[3] The ontologies of this corpus are known to have diverse complexity, size and structure [7]. For the role restriction resolution rule, we made use of the HermiT reasoner Version 1.3.6 [18] for checking satisfiability of conjunctions of definer concepts.

It turns out that several additional optimisations are necessary to make the method perform well on larger ontologies. Especially the role propagation rule creates a lot of unnecessary derivations when applied in its unrestricted form. This can be reduced by analysing the structure of the clause set before applying the rule to see in which cases it actually leads to new derivations on the symbol we want to forget. We further used module extraction in order to reduce the size of the input ontologies. Given an ontology $\mathcal{O}$, the $\top\bot*$-module of $\mathcal{O}$ over Σ contains a subset of the axioms of $\mathcal{O}$ that preserves all consequences of $\mathcal{O}$ in Σ, given $\mathcal{O}$ is consistent [17]. In order to compute $\mathcal{O}^{\Sigma}$, it is therefore sufficient to apply our method on the $\top\bot*$-module of $\mathcal{O}$ over Σ. In order to keep the clauses small, we further apply structural transformation to replace every subconcept C in the TBox that does not contain the symbol we want to forget by a new symbol X, which reduces the number of clauses a lot [12]. These symbols are replaced by the original subconcepts in the final result. For a complete overview of optimisations used we refer to the paper [10] on practical aspects of computing uniform interpolants in $\mathcal{ALC}$.

The corpus for our experiments was created as follows. From the NCBO Bioportal repository, we selected those ontologies that contain role hierarchies, and for which parsing and module extraction using the OWL-API was possible. We then restricted the selected ontologies to $\mathcal{ALCH}$ by removing all axioms that

[1] `http://www.scala-lang.org`
[2] `http://owlapi.sourceforge.net`
[3] `http://bioportal.bioontology.org`

are not expressible in $\mathcal{ALCH}$ using simple reformulations. This led to a corpus of 115 ontologies, on which we ran our experiments.

The experiments were conducted on an Intel Core i5-2400 CPU with four cores running at 3.10 GHz and 8 GB of RAM. Since our implementation does not make use of multi-threading (except for computations of the HermiT reasoner), we ran several experiments in parallel, taking care that experiments do not affect each other due to use of resources.

We started with a series of experiments to evaluate the perfomance of forgetting small sets of symbols, which may for example be interesting for predicate hiding or for computing logical differences between ontology versions, as mentioned in the Introduction. First, we evaluated the performance of concept forgetting. For this, we randomly selected samples of 5, 50, 100 and 150 concept symbols for each ontology and computed the result of forgetting these, with a timeout set to 100 seconds. In 4.5% of the cases, our implementation was not able to compute the uniform interpolant in the given time limit, and in 16.7% of the remaining cases, fixpoints where used in the result. Even though it known that uniform interpolants can be of size triple exponential of the size of the input ontology [14], in our experiments uniform interpolants were much smaller. In fact, in 62.8% of the cases where a uniform interpolant could be computed, the uniform interpolant was smaller than the input ontology. In the worst case however, the uniform interpolant was 104 times bigger than the input ontology. The difference also becomes more apparent when looking at the axiom size.

On average, the average axiom size of the uniform interpolant was 1.8 times bigger than in the input ontology, and the largest axiom size 10.3 times bigger. This effect was to be expected since more information about the role structure of the ontology and indirect concept relations has to be presented in the definitions of fewer concepts. Considering that in the input ontologies the average axiom size was only 3.48, and the average maximal axiom size was 15.21, this still means most axioms were not overly complex. However, in the worst case, the computed uniform interpolant contained an axiom that was 1,406 times bigger than the largest one in the input.

Next we evaluated forgetting of role symbols. Since the role restriction resolution rule makes use of an external reasoner, and can have more than two clauses as premises, one could expect that forgetting role symbols is much more expensive than forgetting concept symbols. On the other hand, since most ontologies have much fewer role symbols than concept symbols, it seems reasonable to conduct the experiments with smaller sets of symbols to be forgotten. We therefore compared how forgetting 5 role symbols performed in comparison with forgetting 5 concept symbols, again with a timeout of 100 seconds. Forgetting role symbols could be performed in 86.6% of the cases in the given time frame, whereas forgetting concept symbols succeeded in 99.8% of the cases. The impact on the ontology size was on the other hand less apparent. In only 3.8% of the cases the uniform interpolant was actually bigger than the input ontology (10.5% for concept symbols), and on average the interpolant was 93% of the size of the input ontology (97% for concept symbols). The largest axiom per ontology was on

$\|\Sigma\|$	Timeouts	Fixpoints	Interpolant Size	Axiom Size	Max. Axiom Size	Average Duration
50	15.12%	6.99%	22.50%	799.37%	1,053.68%	24.2 sec.
100	18.38%	11.57%	45.21%	646.32%	847.36%	21.0 sec.
150	22.25%	13.58%	76.55%	837.66%	5,657.87%	23.7 sec.
All	18.38%	10.44%	45.74%	757.69%	2,309.08%	23.0 sec.

Table 1. Results for computing uniform interpolants.

average 1.58 times larger than in the input ontology (1.18 for concept symbols), and in the worst case 51.1 times larger (360.3 for concept symbols). One might suspect that this result is partly due to the exploitation of role hierarchies using the role restriction monotonicity rules. But it turned out that when ignoring the RBox, the results were nearly unchanged, and even slightly better.

To evaluate our complete method, we computed uniform interpolants for small signatures of size 50, 100 and 150. This corresponds to the applications exhibiting hidden relations and ontology reuse mentioned in the Introduction, as well as predicate hiding, if only a small part of the ontology is to be published.

For these uniform interpolants, usually a large number of symbols, including both role and concept symbols, had to be forgotten from the input ontology, even though module extraction already performs part of the job. For this reason we set a higher timeout of 1,000 seconds. The results are summarised in Table 1. It shows the percentage of experimental runs where a timeout occured, the percentage in the remaining set where fixpoints were used in the result, the ontology size, average axiom size and maximal axiom size of each uniform interpolant compared to the respective values of the input ontologies, and the average duration. In 18.38% of the cases the uniform interpolant could not be computed in 1,000 seconds, and in only 10.44% of the remaining cases, it made use of fixpoint operators. Despite the relatively high number of timeouts, the average duration was only 23 seconds, and the cumulative distribution of durations shows (Figure 5), that around 1,600 out of 2,911 runs (more than half of them) could be performed in less than a second. This suggests that computing uniform interpolants is in most cases a cheap operation.

It is known that uniform interpolants of $\mathcal{ALC}$-ontologies can be in the worst case be triple exponential in the size of the input ontology [14]. When fixpoints are used, the worst case complexity is better, but still double exponential. This bound was not at all reflected in the empirical results, where the average interpolant is less than half the size of the input ontology. In only 6.05% of the cases the uniform interpolant was bigger (see also Figure 5). The axioms in the uniform interpolant were usually around 8–10 times larger than in the input ontology, which is still a reasonable size for ontology analysis considering that in the input ontologies the average axiom size was less than 4.

It should be noted that randomly drawn samples of signatures not neccessarily reflect realistic use cases. One might assume that it is most often desirable

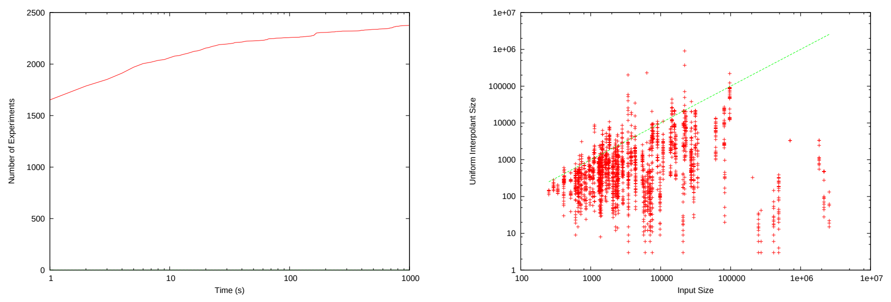

Fig. 5. Cumulative distribution of durations of experimental runs and sizes of the computed uniform interpolants.

to forget or preserve symbols that are closer related to each other, whereas randomly selected symbols are more likely to be randomly distributed along the whole ontology, which can contain thousands of symbols. We therefore believe that our method would perform even better in realistic use cases.

9 Related Work

Most previous work has focused on uniform interpolation in simpler description logics like $\mathcal{EL}$ and DL-Lite (see for example [8, 22, 15, 13]). In [21, 20], one of the first approaches for a more expressive description logic, namely $\mathcal{ALC}$, is presented. Their method uses a tableaux-reasoner to add inferences from the input ontology in an incremental way. Regular checking for TBox-equivalence is used to decide whether the uniform interpolant is computed and the process can stop. By using tableaux-reasoning as a basis, the authors hope to make their method easily extendable with known techniques from existing tableau-reasoners. Its less focused way of deriving inferences make it however unfeasible for large ontologies. In [14], it was discovered that the method is incomplete. The solution offered can be seen as an extension of the original method, even though tableau-reasoning is not stated explicitly. The resulting method can be used to compute all uniform interpolants that can be finitely represented in $\mathcal{ALC}$, but offers no solutions for ontologies where the interpolant cannot be represented without fixpoints.

A more practical approach for forgetting concept symbols in $\mathcal{ALC}$ is presented in [12]. A resolution-based method influenced by [6] is used to derive consequences on the selected concept symbol in a focused way. Experiments on modified ontologies from the NCBO Bioportal ontology show the practicality of this approach under certain restrictions. Since their approach does not use structural transformation, a calculus based on meta-rules is used to make resolutions on nested concepts expressions possible. A disadvantage is that the method does not terminate if infinite chains of nested role-restrictions are derivable. The solution offered is to approximate interpolants by a given (lower) bound instead.

A method using fixpoints for the description logic $\mathcal{EL}$ was presented in [15]. This method aims at forgetting concept symbols, and computes derivation graphs for least common subsumers and most general subsumees of the concept to be eliminated. This graph is analysed to decide whether fixpoint operators are necessary in the result or not. In [13], an automata based representation is used to make finite representations of uniform interpolants possible. The computed automata can be used to decide whether a finite representation in pure $\mathcal{EL}$ is possible and can be translated into corresponding $\mathcal{EL}$-TBoxes in this case.

The method presented in this paper is an extension of a recently introduced method for forgetting concept symbols in $\mathcal{ALC}$-ontologies [11], which is evaluated in [10]. Both methods take ideas from second-order quantifier elimination techniques presented in [4], especially from the resolution-based method SCAN [3] and a method based on a generalised version of Ackermann's Lemma [16]. The latter technique has first been applied for description logics in [19]. Like the methods presented in [12] and [15], the method presented in [11] focuses on forgetting concept symbols. Our current method adds redundancy elimination techniques and is the first practical algorithm for forgetting role symbols from ontologies in expressive description logics.

10 Conclusion and Future Work

We presented a method for forgetting concept and role symbols from $\mathcal{ALCH}$-ontologies, or for computing uniform interpolants of $\mathcal{ALCH}$-ontologies. Since uniform interpolants cannot always be represented in a finite way, the resulting ontology may use fixpoint operators, which can be simulated in $\mathcal{ALCH}$ by extending the signature of the interpolant. Our experimental results suggest that the method is already applicable in a lot of real life situations.

An open point regards the use of fixpoints. One can construct easy examples where our method computes an interpolant with fixpoints, even though the uniform interpolant can be represented in $\mathcal{ALCH}$. Reasons for this are interactions between different fixpoint expressions in the ontology and indirect knowledge encoded in the remaining part of the ontology. For example, it is possible that the cyclic relation expressed by a fixpoint expression is already covered by a set of axioms that was not touched by the method, or that the fixpoint can be represented in a finite way due to entailments from the remaining ontology. Of course this leaves also the question on whether optimal use of fixpoint is actually practical on large ontologies, since an approach focused solely on the symbols we want to forget would not be sufficient here.

References

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

2. Calvanese, D., Giacomo, G.D., Lenzerini, M.: Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In: Proc. IJCAI '99. pp. 84–89. Morgan Kaufmann (1999)
3. Gabbay, D., Ohlbach, H.J.: Quantifier elimination in second-order predicate logic. In: Proc. KR '92. pp. 425–435. Morgan Kaufmann (1992)
4. Gabbay, D.M., Schmidt, R.A., Szalas, A.: Second Order Quantifier Elimination: Foundations, Computational Aspects and Applications. College Publ. (2008)
5. Grau, B.C., Motik, B.: Reasoning over ontologies with hidden content: The import-by-query approach. J. Artificial Intelligence Research 45, 197–255 (2012)
6. Herzig, A., Mengin, J.: Uniform interpolation by resolution in modal logic. In: JELIA '08. LNAI, vol. 5293, pp. 219–231. Springer (2008)
7. Horridge, M., Parsia, B., Sattler, U.: The state of bio-medical ontologies. Bio-Ontologies 2011 (2011)
8. Konev, B., Walther, D., Wolter, F.: Forgetting and uniform interpolation in large-scale description logic terminologies. In: Proc. IJCAI '09. pp. 830–835 (2009)
9. Koopmann, P., Schmidt, R.A.: Forgetting concept and role symbols in $\mathcal{ALCH}$-ontologies. Technical Report, available from `http://www.cs.man.ac.uk/~koopmanp` (2013)
10. Koopmann, P., Schmidt, R.A.: Implementation and evaluation of forgetting in $\mathcal{ALC}$-ontologies. In: Proc. WoMO'13. CEUR-WS.org (2013)
11. Koopmann, P., Schmidt, R.A.: Uniform interpolation of $\mathcal{ALC}$-ontologies using fixpoints. In: Proc. FroCoS'13. LNAI, vol. 8152, pp. 87–102. Springer (2013)
12. Ludwig, M., Konev, B.: Towards practical uniform interpolation and forgetting for $\mathcal{ALC}$ TBoxes. In: Proc. DL'13. pp. 377–389. CEUR-WS.org (2013)
13. Lutz, C., Seylan, I., Wolter, F.: An automata-theoretic approach to uniform interpolation and approximation in the description logic $\mathcal{EL}$. In: Proc. KR'12. AAAI Press (2012)
14. Lutz, C., Wolter, F.: Foundations for uniform interpolation and forgetting in expressive description logics. In: Proc. IJCAI '11. pp. 989–995. AAAI Press (2011)
15. Nikitina, N.: Forgetting in general $\mathcal{EL}$ terminologies. In: Proc. DL'11. CEUR-WS.org (2011)
16. Nonnengart, A., Szałas, A.: A fixpoint approach to second-order quantifier elimination with applications to correspondence theory. In: Logic at Work, pp. 307–328. Springer (1999)
17. Sattler, U., Schneider, T., Zakharyaschev, M.: Which kind of module should I extract? In: Proc. DL'09. CEUR-WS.org (2009)
18. Shearer, R., Motik, B., Horrocks, I.: HermiT: A highly-efficient OWL reasoner. In: Proc. OWLED'08. pp. 26–27. CEUR-WS.org (2008)
19. Szałas, A.: Second-order reasoning in description logics. J. Appl. Non-Classical Logics 16(3-4), 517–530 (2006)
20. Wang, K., Wang, Z., Topor, R., Pan, J.Z., Antoniou, G.: Eliminating concepts and roles from ontologies in expressive descriptive logics. Computational Intelligence (2012)
21. Wang, Z., Wang, K., Topor, R., Zhang, X.: Tableau-based forgetting in $\mathcal{ALC}$ ontologies. In: Proc. ECAI '10. pp. 47–52. IOS Press (2010)
22. Wang, Z., Wang, K., Topor, R.W., Pan, J.Z.: Forgetting for knowledge bases in DL-Lite. Ann. Math. Artif. Intell. 58(1-2), 117–151 (2010)