Saturation-Based Forgetting in the Description Logic $\mathcal{SIF}$

Patrick Koopmann, Renate A. Schmidt

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

Abstract. Forgetting, which has also been studied under the names uniform interpolation, projection and variable elimination, deals with the elimination of symbols from an input ontology in such a way that all entailments in the remaining signature are preserved. The computed result, the uniform interpolant, can be seen as a restricted view of the original ontology, projected to a specified subset of the signature. Forgetting has applications in ontology reuse, ontology analysis and information hiding, and has been studied for a variety of description logics in the last years. However, forgetting in description logics with functional role restrictions and inverse roles has been an open problem so far. In this paper, we study the problem of forgetting concept symbols in the description logic $\mathcal{SIF}$, an expressive description logic supporting transitive roles, inverse roles and functional role restrictions. Saturation-based reasoning has been proven to be a well-suited technique for computing uniform interpolants practically in recently introduced methods. Since existing methods are usually optimised towards a specific aim such as satisfiability checking or classification, they cannot always directly be used for forgetting. In this paper we present a new saturation technique that is complete for forgetting concept symbols, and show how it can be used for computing uniform interpolants.

1 Introduction

We present a method for forgetting concept symbols from $\mathcal{SIF}$-ontologies. Modern applications, especially in bio-informatics or medical domains, lead to the development of ontologies that cover a large vocabulary. With rising complexity, these ontologies become harder to maintain and modify. It can therefore be advantageous to have tool-support for reducing the vocabulary used in an ontology. Forgetting restricts the vocabulary in an ontology in such a way that all entailments over the restricted vocabulary are preserved. In contrast to modules, uniform interpolants are completely in the desired signature and may contain different axioms than the input ontology.

There are numerous applications of forgetting that make the problem worth studying, of which we give a few examples. **Ontology Summary.** Comprehending a complex ontology can be hindered by a too large vocabulary used in the ontology. If the central concepts and roles of the ontology are known, forgetting

all but the central concepts of an ontology can be used to compute a more focused high-level summary of the ontology. **Ontology Analysis.** With increasing complexity of the ontology, understanding the relations between concepts and roles involved becomes more difficult. By forgetting all but a few chosen symbols, one can obtain a direct representation of the interactions between them. **Logical Difference.** In order to maintain an evolving ontology, it is necessary to be able to exhibit changes between different ontology versions. However, a syntactical diff between text representations of the ontologies is rarely useful in practice. In contrast, the logical difference between two ontologies is a semantic notion, which is defined by the difference of logical entailments in the common signature of these ontologies, or in a specified signature [8,7]. [16] show that the logical difference can easily be computed using the uniform interpolants of the ontologies. **Information Hiding.** As pointed out in [4], ontology-based systems are increasingly used in a range of applications that deal with sensitive information. Forgetting can be used as a means to eliminate confidential information if an ontology is to be shared between users with differing privileges.

Methods for forgetting have been investigated for a range of description logics, including DL-Lite [29], $\mathcal{EL}$ [9,17,20], $\mathcal{ALC}$ [28,16,12,11,14,15], $\mathcal{ALCH}$ [10] and $\mathcal{SHQ}$ [13]. So far, forgetting for description logics with inverse roles and functional role restrictions was an open problem.

As with most expressive description logics, $\mathcal{SIF}$ does not have uniform interpolation. This means that the result of forgetting may not always be finitely representable in $\mathcal{SIF}$. Take as an example the ontology $\mathcal{O} = \{A \sqsubseteq \exists r.B, B \sqsubseteq \exists r.B\}$. If we only use the expressivity $\mathcal{SIF}$ provides, forgetting B from $\mathcal{O}$ would result in an infinite ontology of the form $\{A \sqsubseteq \exists r.\exists r.\exists r. \ldots\}$. A solution to this problem is to use fixpoint operators in the resulting ontology [19,12]. Fixpoints are not a common formalism for description logics, but can be simulated in classical description logics using additional concept symbols, and can serve as a basis for approximating of the result of forgetting [12].

As is shown in [18,20], if finite and if fixpoints are not used in the result, already for the description logics $\mathcal{EL}$ and $\mathcal{ALC}$, the result of forgetting can be of size triple exponential in the size of the input. By using fixpoint operators, we obtain a double-exponential upper-bound, which also holds for the description logic $\mathcal{SIF}$. Since this is still of high complexity, a goal-oriented approach is required in order to be able to compute uniform interpolants practically. Practicality has been achieved in [10,13,14,16] by using a saturation-based approach, which eliminates concept symbols by resolving on these symbols. This approach is also followed in this paper. Saturation-based reasoning has recently received increased interest in the description logic community, due to the success of consequence-based reasoning methods for classification [6,22,24,25,26]. However, these methods are optimised for specific reasoning tasks, and can not directly be used for forgetting.

In this paper, we present a new sound and refutationally complete saturation-procedure for $\mathcal{SIF}$, and show that it can be used for forgetting concept symbols in a goal-oriented manner. The method is based on the methods presented

in [12,10], which we extend to incorporate transitive roles, inverse roles and functional role restrictions.

2 Definition of $\mathcal{SIF}\nu$ and Forgetting

We define the description logic $\mathcal{SIF}\nu$, which is $\mathcal{SIF}$ extended with fixpoint operators.

Let N_c, N_r, N_i and N_v be four pair-wise disjoint sets of respectively *concept symbols*, *role symbols*, *individuals* and *concept variables*. A *role* is either of the form r or r^-, where $r \in N_r$. We define the *inverse of a role* $\mathsf{Inv}(R)$ as $\mathsf{Inv}(r) = r^-$ and $\mathsf{Inv}(r^-) = r$. An *RBox* $\mathcal{R}$ is a set of *transitivity axioms* $\mathsf{trans}(R)$, where R is a role. A role R is *transitive* in $\mathcal{R}$ if $\mathsf{trans}(R) \in \mathcal{R}$ or $\mathsf{trans}(\mathsf{Inv}(R)) \in \mathcal{R}$.

$\mathcal{SIF}\nu$-concepts have the following form:

$$\bot \mid A \mid X \mid \neg C \mid C_1 \sqcup C_2 \mid \exists R.C \mid \leq 1R.\top \mid \nu X.C[X],$$

where $A \in N_c$, $X \in N_v$, C, C_1 and C_2 are arbitrary concepts and R is any role, and $C[X]$ is a concept expression in which X occurs under an even number of negations. We define further concept expressions as abbreviations: $\top = \neg\bot$, $C_1 \sqcap C_2 = \neg(\neg C_1 \sqcup \neg C_2)$, $\forall R.C = \neg(\exists R.\neg C)$, $\geq 2R.\top = \neg\leq 1R.\top$. Concepts of the form $\leq 1R.\top$ are called *functional role restrictions*. Concepts of the form $\nu X.C[X]$ are called *greatest fixpoint expressions*. $\nu X.C[X]$ denotes the *greatest fixpoint* of $C[X]$, and ν is the *greatest fixpoint operator*. A concept variable X is *bound* if it occurs in the scope $C[X]$ of a fixpoint expression $\nu X.C[X]$. Otherwise it is *free*. A concept is *closed* if it does not contain any free variables, otherwise it is *open*.

A *TBox* $\mathcal{T}$ 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 closed concepts. $C \equiv D$ is short-hand for the two concept axioms $C \sqsubseteq D$ and $D \sqsubseteq C$. We further require greatest fixpoint expressions to occur only *positively* in an axiom, that is, on the right-hand side of a concept inclusion and only under an even number of negations. An *ontology* $\mathcal{O} = \langle \mathcal{T}, \mathcal{R} \rangle$ consists of a TBox $\mathcal{T}$ and an RBox $\mathcal{R}$ with the additional restriction that roles that are transitive in $\mathcal{R}$ do not occur under functional role restrictions. This is a common restriction that has been used to guarantee decidability of common reasoning tasks for description logics with number restrictions [5] , and our forgetting method assumes that it is satisfied.

In the definition of the semantics of $\mathcal{SIF}\nu$, an *interpretation* $\mathcal{I}$ is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ of the *domain* $\Delta^{\mathcal{I}}$, a nonempty set, and the *interpretation function* $\cdot^{\mathcal{I}}$, which 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 $\mathcal{SIF}\nu$-concepts as follows.

$$\bot^{\mathcal{I}} = \emptyset \qquad (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \qquad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$
$$(\geq nr.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \#\{(x,y) \in r^{\mathcal{I}} \mid y \in C^{\mathcal{I}}\} \geq n\}$$

The semantics of fixpoint expressions is defined following [2]. Whereas concept symbols are assigned fixed subsets of the domain, concept variables range over

arbitrary subsets, which is why only closed concepts have a fixed interpretation. Open concepts are interpreted using *valuations* ρ that map concept variables to subsets of $\Delta^{\mathcal{I}}$. Given a valuation ρ and a set $W \subseteq \Delta^{\mathcal{I}}$, $\rho[X \mapsto W]$ denotes a valuation identical to ρ except that $\rho[X \mapsto W](X) = W$. Given an interpretation $\mathcal{I}$ and a valuation ρ, the function $\cdot^{\mathcal{I}}_{\rho}$ is $\cdot^{\mathcal{I}}$ extended with the cases $X^{\mathcal{I}}_{\rho} = \rho(X)$ and

$$(\nu X.C)^{\mathcal{I}}_{\rho} = \bigcup\{W \subseteq \Delta^{\mathcal{I}} \mid W \subseteq C^{\mathcal{I}}_{\rho[X \mapsto W]}\}.$$

If C is closed, we define $C^{\mathcal{I}} = C^{\mathcal{I}}_{\rho}$ for any valuation ρ. Since C does not contain any free variables in this case, this defines $C^{\mathcal{I}}$ uniquely.

A concept inclusion $C \sqsubseteq D$ is *true* in an interpretation $\mathcal{I}$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and a transitivity axiom $\mathsf{trans}(R)$ is true in $\mathcal{I}$ if for any domain elements $x, y, z \in \Delta^{\mathcal{I}}$ we have $(x, z) \in R^{\mathcal{I}}$ if $(x, y) \in R^{\mathcal{I}}$ and $(y, z) \in R^{\mathcal{I}}$. $\mathcal{I}$ is a *model* of an ontology $\mathcal{O}$ if all axioms in $\mathcal{O}$ are true in $\mathcal{I}$. An ontology $\mathcal{O}$ is *satisfiable* if a model exists for $\mathcal{O}$, otherwise it is *unsatisfiable*. 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. $\mathcal{T} \models C \sqsubseteq D$ holds iff in every model $\mathcal{I}$ of $\mathcal{T}$ we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. If an axiom α is true in all models of $\mathcal{O}$, we write $\mathcal{O} \models \alpha$.

Let $sig(E)$ denote the concept and role symbols occurring in E, where E can denote a concept, an axiom, a TBox, an RBox or an ontology.

Definition 1 (Forgetting). *Let A be a concept symbol and $\mathcal{O}$ and $\mathcal{O}^{-A}$ be ontologies. An ontology $\mathcal{O}^{-A}$ is a* result of forgetting *A from $\mathcal{O}$ if the following conditions hold:*

1. *$A \notin sig(\mathcal{O}^{-A})$, and*
2. *for all concept inclusions α with $A \notin sig(\alpha)$: $\mathcal{O}^{-A} \models \alpha$ iff $\mathcal{O} \models \alpha$.*

Given an ontology $\mathcal{O}$ and a set of concept symbols $\mathcal{S}$, a result of forgetting $\mathcal{S}$ from $\mathcal{O}$ is a result of forgetting each symbol in $\mathcal{S}$ one by one. An ontology $\mathcal{O}^{\mathcal{S}}$ is a uniform interpolant of $\mathcal{O}$ for $\mathcal{S}$ *iff $\mathcal{O}$ is a result of forgetting every symbol from $\mathcal{O}$ that is not in $\mathcal{S}$.*

3 Normalisation

The saturation method works on ontologies of a specific normal form, which is defined as follows.

Definition 2. *Let $N_d \subseteq N_c$ be a set of designated concept symbols called* definer symbols *or, simply,* definers*. A $\mathcal{SIF}$* literal *is a concept of one of the following forms:*

$$A \mid \neg A \mid \exists R.D \mid \forall R.D \mid \geq 2R.\top \mid \leq 1R.\top,$$

where $A \in N_c$, $D \in N_d$, and R is of the form r or r^-, with $r \in N_r$. A $\mathcal{SIF}$ clause *is a transitivity axiom or an axiom of the form $\top \sqsubseteq L_1 \sqcup \ldots \sqcup L_n$, where $L_1, \ldots, L_n$ are $\mathcal{SIF}$ literals. We may omit the leading "$\top \sqsubseteq$" in clauses, and*

Non-Cyclic Definer Elimination:

$$\frac{\mathcal{O} \cup \{D \sqsubseteq C\}}{\mathcal{O}^{[D/C]}} \quad \text{provided } D \notin \mathit{sig}(C)$$

Definer Purification:

$$\frac{\mathcal{O}}{\mathcal{O}^{[D/\top]}} \quad \text{provided } D \text{ occurs only positively in } \mathcal{O}$$

Cyclic Definer Elimination:

$$\frac{\mathcal{O} \cup \{D \sqsubseteq C[D]\}}{\mathcal{O}^{[D/\nu X.C[X]]}} \quad \text{provided } D \in \mathit{sig}(C[D])$$

Fig. 1. Rules for eliminating definer symbols

assume clauses are represented as sets, that is, they do not contain duplicate elements and the order is not important.

An ontology $\mathcal{N}$ is in $\mathcal{SIF}$ normal form *if every axiom in $\mathcal{N}$ is a $\mathcal{SIF}$ clause, and if for every clause $\mathsf{trans}(R) \in \mathcal{N}$, there is also a clause $\mathsf{trans}(\mathsf{Inv}(R)) \in \mathcal{N}$.*

Note that the description logic $\mathcal{SIF}$ allows for number restrictions of the form $\geq 2R.\top$, since $\mathcal{SIF}$ concepts are closed under negation, and $\geq 2R.\top \equiv \neg(\leq 1R.\top)$.

Any $\mathcal{SIF}\nu$ ontology can be transformed into an ontology in $\mathcal{SIF}$ normal form using standard structural transformation and CNF transformation techniques.

Example 1. Consider the following ontology $\mathcal{O}_1$:

$$A_1 \sqcap B_1 \sqsubseteq \bot \qquad A_1 \sqsubseteq \exists r^-.B_2 \qquad B_2 \sqsubseteq \leq 1r.\top \qquad B_2 \sqsubseteq \exists r.(B_1 \sqcup A_2)$$

The $\mathcal{SIF}$ normal form of $\mathcal{O}_1$ is the following set of clauses:

1. $\neg A_1 \sqcup \neg B_1$
2. $\neg A_1 \sqcup \exists r^-.D_1$
3. $\neg D_1 \sqcup B_2$
4. $\neg B_2 \sqcup \leq 1r.\top$
5. $\neg B_2 \sqcup \exists r.D_2$
6. $\neg D_2 \sqcup B_1 \sqcup A_2$

Given a set $\mathcal{N}$ of $\mathcal{SIF}$ clauses such that every clause contains at most one literal of the form $\neg D$, where $D \in N_d$, it is possible to eliminate all definer literals in $\mathcal{N}$ using the rewrite rules shown in Figure 1. This is crucial for our method for forgetting concept symbols, since the method described in the next section operates on sets of $\mathcal{SIF}$ clauses. For the rules in Figure 1, it is assumed that for every definer D occurring negatively in $\mathcal{N}$, the set of clauses of the form $\neg D \sqcup C_1$, $\ldots$, $\neg D \sqcup C_n$ is replaced by a single axiom $D \sqsubseteq C_1 \sqcap \ldots \sqcap C_n$. Note that the last rule in Figure 1 introduces fixpoint operators to the ontology. The rules in Figure 1 are applications of Ackermann's Lemma [1] and its generalisation [21], which have been used in the context of second-order quantifier elimination to eliminate predicate symbols [3]. The result of applying these rules does not contain any definers, but preserves all entailments of input that do not involve definer symbols, which is a consequence of these lemmata.

Transitivity Rule:

$$\frac{C \sqcup \forall R.D \qquad \mathsf{trans}(R)}{C \sqcup \forall R.D_{\mathsf{trans}} \quad \neg D' \sqcup D \quad \neg D' \sqcup \forall R.D_{\mathsf{trans}}}$$

Universalisation Rule:

$$\frac{C_1 \sqcup \exists R.D \qquad C_2 \sqcup \leq 1R.\top}{C_1 \sqcup C_2 \sqcup \forall R.D}$$

Fig. 2. Rules used in the first stage of reasoning.

4 The Calculus

In order to forget a concept symbol A, we infer all inferences on that symbol using a saturation-based approach, and then eliminate occurrences of that symbol from the resulting clause set. Using the transformation rules in Figure 1, we can then eliminate all definer symbols introduced by the normalisation or throughout the reasoning process.

Due to possible interactions between the rules of our calculus, we process the clause set in several stages, where only certain rules are allowed in each stage. This is a major difference between this method and the methods for $\mathcal{ALC}$ and $\mathcal{ALCH}$ presented in [12,10], and is necessary to guarantee termination of the method. In the first stage, we only apply rules that infer clauses with universal restrictions. The function of this stage is to infer all clauses that can serve as premise in the second stage of the calculus. In the second stage, we handle inverse roles in universal restrictions using the role inversion rule. In the last stage, we apply all remaining inferences of the calculus with the aim of computing inferences on A. We describe the stages one by one, and illustrate them on the example introduced in the last section.

Stage 1. The rules for the first stage are shown in Figure 2. The transitivity rule is directly taken from [13], which presents a similar calculus for the description logic $\mathcal{SHQ}$. The rule works in a similar fashion as common rewriting rules to reduce reasoning with transitivity roles to reasoning without (see for example [27,23]).

The universalisation rule infers from a functional role restriction and an existential role restriction a universal restriction. The soundness of this rule can be explained as follows. If a domain element x in a model $\mathcal{I}$ has an R-successor that satisfies D ($x \in (\exists R.D)^{\mathcal{I}}$), and if x has maximally one R-successor ($x \in (\leq 1R.\top)^{\mathcal{I}}$), then every R-successor of x satisfies D, since there is only one such successor ($x \in (\forall R.D)^{\mathcal{I}}$). This means $(\exists R.D \sqcap \leq 1R.\top) \models \forall R.D$, and implies that the universalisation rule is sound.

Role Inversion Rule:

$$\frac{C \sqcup \forall R.D}{D \sqcup \forall \mathsf{Inv}(R).D_{\mathsf{Inv}} \qquad \neg D_{\mathsf{Inv}} \sqcup C}$$

Fig. 3. Role inversion rule used in the second stage of reasoning.

Resolution Rule:

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

$\forall$-Rule:

$$\frac{C_1 \sqcup \forall R.D_1 \qquad C_2 \sqcup \mathsf{Q}R.D_2}{C_1 \sqcup C_2 \sqcup \mathsf{Q}R.D_{12}}$$

where $\mathsf{Q} \in \{\forall, \exists\}$ and D_{12} is a possibly new definer representing $D_1 \sqcap D_2$.

$\exists$-Rule:

$$\frac{C_1 \sqcup \exists R.D_1 \qquad C_2 \sqcup \exists R.D_2}{C_1 \sqcup C_2 \sqcup \exists R.D_{12} \sqcup \geq 2R.\top}$$

where D_{12} is a possibly new definer representing $D_1 \sqcap D_2$.

Fig. 4. Rules used in the third stage of reasoning.

Example 2. Following the clause set constructed in the last example, in Stage 1, we apply the universalisation rule on Clause 4 and 5 to infer the following clause.

$$7.\ \neg B_2 \sqcup \forall r.D_2 \qquad \textit{(Universalisation 4, 5)}$$

Stage 2. In this stage, the inversion rule shown in Figure 3 is applied.

Example 3. Continuing the previous example, there is only one clause in the current clause set that has a universal role restriction, which is Clause 7 that was derived in Stage 1. By applying the role inversion rule, the following new clauses are derived:

$$8.\ D_2 \sqcup \forall r^{-}.D_{\mathsf{Inv}} \qquad \textit{(Role Inversion 7)}$$
$$9.\ \neg D_{\mathsf{Inv}} \sqcup \neg B_2 \qquad \textit{(Role Inversion 7)}$$

Stage 3. This is the main reasoning stage, where we compute all inferences on the symbols we want to forget. The rules for this stage are shown in Figure 4. The rules of this stage are an extension of the calculus used for forgetting presented in [12,10]. The $\exists$-rule is influenced by the calculus for reasoning in $\mathcal{SHQ}$ presented in [13].

A key for ensuring termination is the dynamic introduction of new symbols through the rules of the calculus. This is necessary to preserve the normal form

and still be able to infer all clauses that are required for the forgetting result. There are two rules in the first stage that introduce new definers: the transitivity rule and the role inversion rule. In the third stage, the $\forall$-rule and the $\exists$-rule may introduce new definers that represent conjunctions of existing definers. More specifically, given two definers D_1 and D_2, we may introduce a new definer D_{12} representing $D_1 \sqcap D_2$ by adding the clauses $\neg D_{12} \sqcup D_1$ and $\neg D_{12} \sqcup D_2$. By doing this in an optimised way, the number of definer symbols introduced can be restricted to by 2^n, where n is the number of definer symbols present in the input clause set [12].

The resolution rule is standard in resolution-based reasoning. The $\forall$-rule are directly taken from [12]. The $\forall$-rule propagates concepts below a universal role restrictions into other role restrictions. If $\forall R.D_1$ is satisfied by some element x of the domain, we know that all R-successors of x have to satisfy D_1. This implies that, if x furthermore satisfies $\mathsf{Q}R.D_2$ for some $\mathsf{Q} \in \{\exists, \forall\}$, then it also satisfies $\mathsf{Q}R.(D_1 \sqcap D_2)$.

The $\exists$-rule is new and necessary in order to preserve entailments that use a $\geq$2-restriction. Assume we have a model with a domain element x that has at least one R-successor x_1 satisfying D_1 and at least one R-successor x_2 that satisfies D_2. If $x_1 = x_2$, then x satisfies $\exists R.(D_1 \sqcap D_2)$. If $x_1 \neq x_2$, then x satisfies $\geq 2R.\top$. Hence, we have $(C_1 \sqcup \exists R.D_1) \sqcap (C_2 \sqcup \exists R.D_2) \models (C_1 \sqcup C_2 \sqcup \exists R.(D_1 \sqcap D_2) \sqcup \geq 2R.\top)$, and the $\exists$-rule is sound.

In order to forget a concept symbol A, we compute all resolvents on A and on definer literals that lead to clauses with maximally one negative definer literal. Clauses with multiple negative definer literals are not needed for the computation, which can be argued in similar ways as in [12].

Note that even though only the resolution rule directly infers clauses on a concept symbol, the $\forall$- and the $\exists$-rule may introduce new definers, which subsequently makes new inferences on the symbol to be forgotten possible. For the forgetting result, only inferences with maximally one negative definer literal are required. The role inversion rule may derive clauses with more than one negative definer literal, but these inferences are only required if they allow for further inferences of clauses with maximally one negative definer.

In the resulting clause set, we omit all clauses containing A or more than one negative definer literal. Then, we eliminate all definers using the technique described in the last section. The ontology obtained is the result of forgetting A from the original ontology.

Example 4. Assume we want to forget B_1 and B_2. We begin by computing inferences on B_1, for which only one inference step is needed.

$$10.\ \neg D_2 \sqcup \neg A_1 \sqcup A_2 \qquad \textit{(Resolution 1, 6)}$$

We continue by computing inferences on B_2. This time, in order to make all inferences possible, we have to use the $\forall$-rule first.

11. $\neg D_1 \sqcup \leq 1 r.\top$	*(Resolution 3, 4)*
12. $\neg D_1 \sqcup \exists r.D_2$	*(Resolution 3, 5)*
13. $\neg D_1 \sqcup \forall r.D_2$	*(Resolution 3, 7)*
14. $\neg A_1 \sqcup D_2 \sqcup \exists r^-.D_{1\mathsf{Inv}}$	*($\forall$-rule 2, 8)*
15. $\neg D_{1\mathsf{Inv}} \sqcup D_1$	*($D_{1\mathsf{Inv}} \sqsubseteq D_1 \sqcap D_{\mathsf{Inv}}$)*
16. $\neg D_{1\mathsf{Inv}} \sqcup D_{\mathsf{Inv}}$	*($D_{1\mathsf{Inv}} \sqsubseteq D_1 \sqcap D_{\mathsf{Inv}}$)*
17. $\neg A_1 \sqcup A_2 \sqcup \exists r^-.D_{1\mathsf{Inv}}$	*(Resolution 10, 14)*
18. $\neg D_{1\mathsf{Inv}} \sqcup B_2$	*(Resolution 3, 15)*
19. $\neg D_{1\mathsf{Inv}} \sqcup \neg B_2$	*(Resolution 9, 16)*
20. $\neg D_{1\mathsf{Inv}}$	*(Resolution 18, 19)*

Even though we did not discuss redundancy elimination techniques here, one can easily see that further inferences of clauses of the form $\neg D_{1\mathsf{Inv}} \sqcup C$ are not needed, since we already inferred the unary clause $\neg D_{1\mathsf{Inv}}$. For the same reason, we do not have to include any other clause containing $\neg D_{1\mathsf{Inv}}$ in the result. In fact, no further inferences are necessary for the forgetting result. If we ignore all clauses that do contain the symbols B_1 and B_2 we are forgetting, we are left with Clauses 2, 7, 10–13, 17 and 20. Eliminating the definers in these clauses results in the following ontology:

$$A_1 \sqsubseteq \exists r^-.(\leq 1 r.\top \sqcap \exists r.(\neg A_1 \sqcup A_2) \sqcap \forall r.(\neg A_1 \sqcup A_2))$$
$$A_1 \sqsubseteq A_2 \sqcup \exists r^-.\bot$$

We can simplify the second axiom to $A_1 \sqsubseteq A_2$, which allows us to further simplify the first axiom, and obtain as result of forgetting B_1 and B_2 the following:

$$A_1 \sqsubseteq \exists r^-.(\leq 1 r.\top \sqcap \exists r.\top) \qquad\qquad A_1 \sqsubseteq A_2$$

5 Correctness of the Method

In order to prove that the method is correct, we show that the resulting ontology preserves all entailments of axioms that do not use the symbols to be forgotten. More formally, if $\mathcal{O}^{-A}$ denotes the output of our method for an ontology $\mathcal{O}$ and a concept symbol A, we show that $\mathcal{O}^{-A} \models \alpha$ iff $\mathcal{O} \models \alpha$ for all axioms α with $A \notin \mathit{sig}(\alpha)$. If $\alpha = C_1 \sqsubseteq C_2$, we can prove $\mathcal{O} \models \alpha$ by showing that $\mathcal{O} \cup \{\exists r^*.(C_1 \sqcap \neg C_2)\}$ is unsatisfiable, where r^* is a role not occurring in $\mathcal{O}$.

In order to show that our forgetting method works correctly, we extend our reasoning method to a refutational complete calculus, that, in order to prove the satisfiability of a clause set, first infers inferences on a designated concept symbol A using only the rules of the forgetting method. If this calculus is refutational complete, we have that a contradiction can be derived from $\mathcal{O} \cup \{\exists r^*.(C_1 \sqcap \neg C_2)\}$

$\exists$-Elimination Rule:

$$\frac{C_1 \sqcup \exists R.D \qquad \neg D}{C_1}$$

$\geq$2-Elimination Rule I:

$$\frac{C_1 \sqcup \geq 2R.\top \qquad C_2 \sqcup \leq 1R.\top}{C_1 \sqcup C_2}$$

$\geq$2-Elimination Rule II:

$$\frac{C_1 \sqcup \geq 2R.\top \qquad C_2 \sqcup \forall R.D \qquad \neg D}{C_1 \sqcup C_2}$$

Fig. 5. Additional rules needed for refutational completeness.

exactly in the same cases as it can from $\mathcal{O}^{-A} \cup \{\exists r^*.(C_1 \sqcap \neg C_2)\}$. This implies $\mathcal{O}^{-A} \models C_1 \sqsubseteq C_2$ iff $\mathcal{O} \models C_1 \sqsubseteq C_2$ for all concept inclusions $C_1 \sqsubseteq C_2$ with $A \notin sig(C_1 \sqsubseteq C_2)$, as required.

In order to obtain a refutationally complete calculus, we additionally need the rules shown in Figure 5. Whereas the rules for the forgetting procedure are aimed at making inferences on concept symbols possible, they are not sufficient for detecting whether a set of clauses is unsatisfiable. The rules in Figure 5 are aimed at detecting inconsistencies between clauses and eliminating unsatisfiable literals, and transform the set of rules into a decision procedure for $\mathcal{SIF}$-ontology satisfiability.

The $\exists$-elimination rule eliminates unsatisfiable literals of the form $\exists R.D$. The $\geq$2-elimination rule I resolves on literals of the form $\geq 2R.\top$ and $\leq 1R.\top$. The rule is sound since an individual can only satisfy $\geq 2R.\top$ or $\leq 1R.\top$ at the same time. The $\geq$2-elimination rule II eliminates unsatisfiable literals, similarly to the $\exists$-elimination rule. If the definer D is unsatisfiable, any instance of $\forall R.D$ cannot have R-successors. Therefore, we can resolve between $\forall R.D$ and $\geq 2R.\top$.

We obtain the reasoning procedure $Ref_{\mathcal{SIF}}$ by extending the forgetting procedure presented in the last section by the rules in Figure 5, which are applied in the last reasoning stage. We further refine the calculus with an ordering. The reasoning procedure $Ref^A_{\mathcal{SIF}}$ uses the same rules as $Ref_{\mathcal{SIF}}$, but for clauses containing the concept symbol A, it only performs inferences on literals of the form A or $\neg A$.

We have the following theorem.

Theorem 1. *Let A be any concept symbol, $Ref_{\mathcal{SIF}}$ and $Ref^A_{\mathcal{SIF}}$ are sound and refutationally complete, that is, for any set $\mathcal{N}$ of clauses, one can infer the empty clause iff $\mathcal{N}$ is inconsistent.*

Proof (Sketch). In order to prove refutational completeness, we have to show that we can build a model based on any saturated set $\mathcal{N}^*$ of clauses such that $\bot \notin \mathcal{N}^*$.

Such a model can be obtained by adapting the model construction presented in [10]. This construction first constructs a model fragment for each definer, and then connects these elements to a complete model, where each definer is represented by a designated domain element. Different to $\mathcal{ALCH}$, $\mathcal{SIF}$ does not have the finite model property. By unravelling the possibly cyclic models created in [10] to a possibly infinite tree, it is however possible to construct a model for $\mathcal{N}^*$, which can be verified by a careful analysis of the cases used in the proof in [10]. □

This theorem allows us to establish that the forgetting procedure described in the last section is correct.

Theorem 2. *For any $\mathcal{SIF}$-ontology $\mathcal{O}$ and any concept symbol A, the described method always terminates and computes the result of forgetting A from $\mathcal{O}$. A uniform interpolant for any ontology $\mathcal{O}$ and signature $\mathcal{S}$ with $N_r \subseteq \mathcal{S}$ can be computed by step-wise forgetting every symbol in $sig(\mathcal{O}) \setminus \mathcal{S}$.*

6 Conclusion and Future Work

We described a method for forgetting concept symbols from ontologies formulated in the description logic $\mathcal{SIF}$, where results are represented in $\mathcal{SIF}\nu$. Forgetting eliminates a specified set of symbols from an ontology in such a way, that all entailments that do not use these symbols are preserved. The method uses a resolution-based saturation procedure to compute inferences on the symbols to be eliminated. By extending this saturation procedure to a refutationally complete reasoning method, which performs inferences on the symbols to be forgotten first, we could prove that our method indeed preserves all entailments in the desired signature.

In order to properly handle the interactions between functional role restrictions and inverse roles without losing termination, the method works in three stages. In the first stage all clauses with a universal restriction are computed, on which in the second stage inferences based on inverse roles are performed. An interesting question is whether this approach can also be used in connection with role hierarchies or with cardinality restrictions. A method for forgetting in the description logic $\mathcal{SHQ}$ is presented in [13]. A simple combination of the rules presented in this paper and presented in [13] is not sufficient to obtain a refutational complete method already for $\mathcal{SHIF}$. An open question is whether this also affects the appropriateness of such a calculus for forgetting concept symbols, and whether a sufficient calculus can be developed by adding additional rules and possibly further extending the underlying description logic, for example with role conjunctions.

We are working on a prototypical implementation of the method, which we aim to integrate into our forgetting tool LETHE.[1] Experiments on similar methods for $\mathcal{ALC}$, $\mathcal{ALCH}$ and $\mathcal{SHQ}$ had promising results [11,10,13,15], and we are confident that similar results can be obtained for $\mathcal{SIF}$.

[1] `www.cs.man.ac.uk/~koopmanp/lethe`

References

1. Ackermann, W.: Untersuchungen über das Eliminationsproblem der mathematischen Logik. Mathematische Annalen 110(1), 390–413 (1935)
2. Calvanese, D., De Giacomo, G., Lenzerini, M.: Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99. pp. 84–89. Morgan Kaufmann (1999)
3. Gabbay, D.M., Schmidt, R.A., Szałas, A.: Second Order Quantifier Elimination: Foundations, Computational Aspects and Applications, Studies in Logic, vol. 12. College Publications (2008)
4. Grau, B.C.: Privacy in ontology-based information systems: A pending matter. Semantic Web 1(1-2), 137–141 (2010)
5. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for very expressive description logics. Logic Journal of the IGPL 8(3), 239—263 (2000)
6. Kazakov, Y.: Consequence-Driven Reasoning for Horn $\mathcal{SHIQ}$ Ontologies. In: Boutilier, C. (ed.) Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-09). pp. 2040–2045. AAAI Press (2009)
7. Konev, B., Ludwig, M., Walther, D., Wolter, F.: The logical difference for the lightweight description logic $\mathcal{EL}$. Journal of Artificial Intelligence Research 44(1), 633–708 (2012)
8. Konev, B., Walther, D., Wolter, F.: The logical difference problem for description logic terminologies. In: Automated Reasoning, Lecture Notes of Computer Science, vol. 5195, pp. 259–274. Springer (2008)
9. Konev, B., Walther, D., Wolter, F.: Forgetting and uniform interpolation in large-scale description logic terminologies. In: Proceedings of the Internation Conference on Artificial Intelligence (IJCAI-09). pp. 830–835. AAAI Press (2009)
10. Koopmann, P., Schmidt, R.A.: Forgetting concept and role symbols in $\mathcal{ALCH}$-ontologies. In: Logic for Programming, Artificial Intelligence and Reasoning. Lecture Notes in Computer Science, vol. 8312, pp. 552–567. Springer (2013)
11. Koopmann, P., Schmidt, R.A.: Implementation and evaluation of forgetting in $\mathcal{ALC}$-ontologies. In: Proceedings of the 7th International Workshop on Modular Ontologies (WoMO'13). CEUR Workshop Proceedings, vol. 1081, pp. 37–48. CEUR-WS.org (2013)
12. Koopmann, P., Schmidt, R.A.: Uniform interpolation of $\mathcal{ALC}$-ontologies using fixpoints. In: Frontiers of Combining Systems. Lecture Notes in Computer Science, vol. 8152, pp. 87–102. Springer (2013)
13. Koopmann, P., Schmidt, R.A.: Count and forget: Uniform interpolation of $\mathcal{SHQ}$-ontologies. In: Automated Reasoning. Lecture Notes in Computer Science, vol. 8562, pp. 434–448. Springer (2014)
14. Koopmann, P., Schmidt, R.A.: Forgetting and uniform interpolation for $\mathcal{ALC}$-ontologies with ABoxes. In: Proceedings of the 27th International Workshop of Description Logics (DL 2014). CEUR Workshop Proceedings, vol. 1193, pp. 245–257. CEUR-WS.org (2014)
15. 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 (AAAI-15). vol. 1, pp. 175–181. AAAI-Press (2015)
16. Ludwig, M., Konev, B.: Practical uniform interpolation and forgetting for $\mathcal{ALC}$ TBoxes with applications to logical difference. In: Proc. KR'14. AAAI Press (2014)

17. 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: Proceedings of the Thirteenth International Conference (KR-12). pp. 286–296. AAAI Press (2012)
18. Lutz, C., Wolter, F.: Foundations for uniform interpolation and forgetting in expressive description logics. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-11). pp. 989–995. AAAI Press (2011)
19. Nikitina, N.: Forgetting in general $\mathcal{EL}$ terminologies. In: Proceedings of the 2011 International Workhop on Description Logics (DL2011). CEUR Workshop Proceedings, vol. 745, pp. 345–355. CEUR-WS.org (2011)
20. Nikitina, N., Rudolph, S.: (Non-) Succinctness of uniform interpolants of general terminologies in the description logic $\mathcal{EL}$. Artificial Intelligence 215, 120–140 (2014)
21. 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, Studies in Fuzziness and Soft Computing, vol. 24, pp. 307–328. Springer (1999)
22. Ortiz, M., Rudolph, S., Simkus, M.: Worst-case optimal reasoning for the Horn-DL fragments of OWL 1 and 2. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference (KR-10) (2010)
23. Schmidt, R.A., Hustadt, U.: A principle for incorporating axioms into the first-order translation of modal formulae. In: Automated Deduction–CADE-19, pp. 412–426. Springer (2003)
24. Simancík, F., Kazakov, Y., Horrocks, I.: Consequence-based reasoning beyond Horn ontologies. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI-11). vol. 22, pp. 1093–1098. AAAI Press (2011)
25. Simancík, F., Motik, B., Krötzsch, M.: Fixed parameter tractable reasoning in DLs via decomposition. In: Proceedings of the 24th International Workshop on Description Logics (DL 2011). CEUR Workshop Proceedings, vol. 745, pp. 400–410. CEUR-WS.org (2011)
26. Steigmiller, A., Glimm, B., Liebig, T.: Coupling tableau algorithms for expressive description logics with completion-based saturation procedures. In: Automated Reasoning, Lecture Notes of Computer Science, vol. 8562, pp. 449–463. Springer (2014)
27. Tobies, S.: Complexity results and practical algorithms for logics in knowledge representation. Ph.D. thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany (2001)
28. Wang, K., Wang, Z., Topor, R.W., Pan, J.Z., Antoniou, G.: Eliminating concepts and roles from ontologies in expressive descriptive logics. Computational Intelligence 30(2), 205–232 (2014)
29. Wang, Z., Wang, K., Topor, R.W., Pan, J.Z.: Forgetting for knowledge bases in DL-Lite. Annals of Mathematics and Artificial Intelligence 58(1–2), 117–151 (2010)