FAME(Q): An Automated Tool for Forgetting in Description Logics with Qualified Number Restrictions

Yizheng Zhao[1,2,3] and Renate A. Schmidt[3]

[1] National Key Laboratory for Novel Software Technology, Nanjing University, China
[2] School of Artificial Intelligence, Nanjing University, China
[3] School of Computer Science, The University of Manchester, UK

Abstract. In this paper, we describe FAME(Q), a Java-based implementation of a forgetting method developed for eliminating concept and role names from $\mathcal{ALCOQH}$-ontologies. FAME(Q) is presently the only tool for concept forgetting in description logics with qualified number restrictions and nominals, and the only tool for role forgetting in description logics with qualified number restrictions. FAME(Q) can be used as a stand-alone tool or a Java library for forgetting, or related tasks. An evaluation of FAME(Q) on a large corpus of biomedical ontologies shows that the tool is able to compute forgetting solutions in 90% of the test cases; in most cases, the solutions are computed within a few seconds.

1 Introduction

Forgetting is an ontology re-engineering technique that seeks to produce new ontologies from existing ones using only a subset of their signature while preserving all logical consequences up to the names in the subset. This is done by eliminating from an ontology a set of concept and role names (the *forgetting signature*) in such a way that all logical consequences are preserved up to the names in the remaining signature. The ontology produced by forgetting (the *forgetting solution*), can be seen as a *view* of the original ontology. In traditional databases, a view is a subset of a database, whereas in ontologies, a view is more than a subset; it may contain not only axioms contained in the original ontology, but also contains those entailed by the ontology (implicitly contained in the ontology). Forgetting is potentially useful for many ontology processing tasks such as ontology reuse, alignment, versioning, merging, debugging, repair, and logical difference computation [1,10,13,3,12,6,5,14,15]. Forgetting is also useful for other tasks such as information hiding and explanation generation [5,2].

At present, practical methods for forgetting in description logics with qualified number restrictions are the resolution-based approach of the LETHE system [8,7,9] and the one developed by [17,19]. FAME(Q) is a Java implementation of the latter, which computes uniform interpolants for $\mathcal{ALCOQH}$-ontologies. The method is a hybrid approach that makes use of both resolution and Ackermann's

Lemma. It is so far the only approach able to forget concept and role names in description logics with qualified number restrictions.

In this paper, we describe the forgetting method used by FAME(Q), the implementation of FAME(Q), and details of an evaluation of FAME(Q) on a large corpus of publicly accessible biomedical ontologies. The current version of FAME(Q) can be downloaded via `http://www.cs.man.ac.uk/~schmidt/sf-fame/`.

2 Forgetting for $\mathcal{ALCOQH}$-Ontologies

Let $\mathsf{N_C}$, $\mathsf{N_R}$ and $\mathsf{N_I}$ be (countably infinite and pairwise disjoint) sets of *concept names*, *role names* and *individual names* (*nominals*), respectively. *Concepts* in $\mathcal{ALCOQH}$ have one of the following forms:

$$\top \mid \bot \mid a \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \geq mr.C \mid \leq nr.C,$$

where $a \in \mathsf{N_I}$, $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, C and D denote arbitrary concepts, and $m \geq 1$ and $n \geq 0$ are natural numbers. Further concepts are defined as abbreviations: $\exists r.C = \geq 1r.C$, $\forall r.C = \leq 0r.\neg C$, $\neg \geq mr.C = \leq nr.C$ and $\neg \leq nr.C = \geq mr.C$, where $n = m - 1$. Concepts of the form $\geq mr.C$ and $\leq nr.C$ are referred to as *(qualified) number restrictions*.

An $\mathcal{ALCOQH}$-ontology is comprised of a TBox, an RBox and an ABox. A TBox is a finite set of axioms of the form $C \sqsubseteq D$ (*concept inclusions*), where C and D are concepts. An RBox is a finite set of axioms of the form $r \sqsubseteq s$ (*role inclusions*), where $r, s \in \mathsf{N_R}$. An ABox is a finite set of axioms of the form $C(a)$ (*concept assertions*) and $r(a, b)$ (*role assertions*), where $a, b \in \mathsf{N_I}$, $r \in \mathsf{N_R}$, and C is a concept.

Forgetting can be defined in two closely related ways. In particular, it can be defined as the dual of *uniform interpolation* or model-theoretically as *semantic forgetting* [6,17,4]. The two notions differ in the sense that uniform interpolation preserves all *logical consequences* whereas semantic forgetting preserves *semantic equivalence* up to certain names. The results of semantic forgetting (the *semantic solutions*), are in general stronger than those of uniform interpolation (the *uniform interpolants*). This means that semantic solutions always entail uniform interpolants, but the converse does not hold. Uniform interpolants are always expressible in the source logic, while semantic solutions are often not, and may require an extended target language to express them.

By $\mathsf{sig_C}(X)$ and $\mathsf{sig_R}(X)$ we denote respectively the sets of the concept names and role names that occur in X, where X ranges over concepts, roles, clauses, axioms, sets of clauses and sets of axioms (ontologies). By $\mathsf{sig}(X)$ we denote the union of $\mathsf{sig_C}(X)$ and $\mathsf{sig_R}(X)$.

Definition 1 (Uniform Interpolation for $\mathcal{ALCOQH}$). *Let $\mathcal{O}$ be an $\mathcal{ALCOQH}$-ontology and let $\mathcal{F} \subseteq \mathsf{sig_C}(\mathcal{O})$ be a set of concept and role names. An ontology $\mathcal{V}$ is an* $\mathcal{ALCOQH}$-uniform interpolant *of $\mathcal{O}$ for $\mathsf{sig}(\mathcal{O})\backslash\mathcal{F}$ iff the following conditions hold: (i) $\mathsf{sig}(\mathcal{V}) \subseteq \mathsf{sig}(\mathcal{O})\backslash\mathcal{F}$, and (ii) for any axiom α with $\mathsf{sig}(\alpha) \subseteq \mathsf{sig}(\mathcal{O})\backslash\mathcal{F}$, $\mathcal{V} \models \alpha$ iff $\mathcal{O} \models \alpha$. In this case, $\mathsf{sig}(\mathcal{O})\backslash\mathcal{F}$ is called the* interpolation signature, *i.e., the set of concept and role names to be preserved.*

Definition 1 says that uniform interpolants have the same logical consequences with the original ontologies up to the interpolation signature.

Definition 2 below says that semantic solutions preserve equivalence up to the interpretations of the names in the forgetting signature $\mathcal{F}$. We say that $\mathcal{I}$ and $\mathcal{I}'$ are *equivalent up to a set $\mathcal{F}$ of concept and role names*, or *$\mathcal{F}$-equivalent*, if $\mathcal{I}$ and $\mathcal{I}'$ coincide but differ possibly in the interpretations of the names in $\mathcal{F}$.

Definition 2 (Semantic Forgetting for $\mathcal{ALCOQH}$). *Let $\mathcal{O}$ be an $\mathcal{ALCOQH}$-ontology and let $\mathcal{F} \subseteq \mathsf{sig}(\mathcal{O})$ be a set of concept and role names. An ontology $\mathcal{V}$ is a* semantic solution *of forgetting $\mathcal{F}$ from $\mathcal{O}$ iff the following conditions hold: (i) $\mathsf{sig}(\mathcal{V}) \subseteq \mathsf{sig}(\mathcal{O}) \backslash \mathcal{F}$ and (ii) for any interpretation $\mathcal{I}$: $\mathcal{I} \models \mathcal{V}$ iff $\mathcal{I}' \models \mathcal{O}$, for some interpretation $\mathcal{I}'$ $\mathcal{F}$-equivalent to $\mathcal{I}$. $\mathcal{F}$ is called the* forgetting signature*, i.e., the set of concept and role names to be forgotten.*

3 The Forgetting Method

Next, we briefly describe the forgetting method implemented in FAME(Q). The method is mainly based on two calculi: a calculus for concept name elimination and a calculus for role name elimination. The former was presented in our recent work [19] and the latter in [17]. The method is terminating and sound.

Both calculi operate on $\mathcal{ALCOQH}$-ontologies in *clausal normal form*, which are obtained from axioms using the standard transformations based on logical equivalence such as $\neg\neg \geq mr.C = \geq mr.C$. In the following, we always use the notation $\mathcal{N}$ to denote a set of clauses (clausified from an $\mathcal{ALCOQH}$-ontology).

Definition 3 (Clausal Normal Form). *A* TBox literal *in $\mathcal{ALCOQH}$ is a concept of the form a, $\neg a$, A, $\neg A$, $\geq mr.C$ or $\leq nr.C$, where $a \in \mathsf{N_I}$, $r \in \mathsf{N_R}$, C is a concept, and $m > 1$ and $n > 0$ are natural numbers. A* TBox clause *in $\mathcal{ALCOQH}$ is a disjunction of a finite number of TBox literals. An* RBox clause *in $\mathcal{ALCOQH}$ is a disjunction of a role name and a negated role name. A clause is called an* $\mathcal{S}$-clause *if it contains $\mathcal{S}$, for any concept/role name $\mathcal{S}$ in $\mathsf{N_C} \cup \mathsf{N_R}$.*

Our method is a rounds-based method, where forgetting solutions (uniform interpolants and semantic solutions) are computed by iteratively eliminating the (concept and role) names in $\mathcal{F}$. We call the name under consideration for forgetting in the current round the *pivot*.

The calculus for eliminating a concept name from a set $\mathcal{N}$ of clauses includes two purify rules and one combination rule.[4] The purify rules are applied when the pivot occurs only positively or only negatively in $\mathcal{N}$, i.e., the pivot is *pure* in $\mathcal{N}$. The purify rules say that if the pivot occurs only positively (negatively) in $\mathcal{N}$, it is eliminated by substitution with the top (bottom) concept. The combination rule is applied when the pivot occurs both positively and negatively in $\mathcal{N}$, i.e., the pivot is *impure* in $\mathcal{N}$. It is applicable iff $\mathcal{N}$ is in a specialized normal form called *A-reduced form*, if A is the concept pivot.

[4] In [19], the combination rule is named the Ackermann rule.

Definition 4 (A-Reduced Form). *Let $\mathcal{N}$ be a set of clauses. Let $A \in \mathsf{sig_C}(\mathcal{N})$. A clause is in* A-reduced form *if it has the form $C \sqcup A$, $C \sqcup \neg A$, $C \sqcup \geq mr.A$, $C \sqcup \geq mr.\neg A$, $C \sqcup \leq nr.A$ or $C \sqcup \leq nr.\neg A$, where $r \in \mathsf{N_R}$, C is a clause that does not contain A, and $m \geq 1$ and $n \geq 0$ are natural numbers. A set $\mathcal{N}$ of clauses is in* A-reduced form *if all A-clauses in $\mathcal{N}$ are in A-reduced form.*

A-clauses not in A-reduced form can be transformed into A-reduced form by introducing *definer names* (or *definers* for short). Once $\mathcal{N}$ is in A-reduced form, one can immediately apply the combination rule to $\mathcal{N}$ to eliminate A. For space reasons we do not present and describe the combination rule in this paper, but refer the reader to [19] for a comprehensive description of the rule.

The calculus for eliminating a role name from $\mathcal{N}$ includes two purify rules and five combination rules.[5] The purify rules are applied when the pivot is *pure* in $\mathcal{N}$. The combination rules are applied when the pivot is *impure* in $\mathcal{N}$. They are applicable iff $\mathcal{N}$ is in *r-reduced form*, where r is the pivot.

Definition 5 (r-Reduced Form). *Let $\mathcal{N}$ be a set of clauses. Let $r \in \mathsf{sig_R}(\mathcal{N})$. A TBox clause is in* r-reduced form *if it has the form $C \sqcup \geq mr.D$ or $C \sqcup \leq nr.D$, where C and D are concepts that do not contain r, and $m \geq 1$ and $n \geq 0$ are natural numbers. An RBox clause is in* r-reduced form *if it has the form $\neg s \sqcup r$ or $s \sqcup \neg r$, where $s \in \mathsf{N_R}$ and $s \neq r$. A set $\mathcal{N}$ of clauses is in* r-reduced form *if all r-clauses in $\mathcal{N}$ are in r-reduced form.*

r-clauses not in r-reduced form can be transformed into r-reduced form by introducing *definers* as in concept forgetting. Once $\mathcal{N}$ is in r-reduced form, we apply an appropriate combination rule to $\mathcal{N}$ to eliminate r. We refer the reader to [19] for presentation and a comprehensive description of the rules.

In order to be able to express more semantic solutions of concept forgetting, the target language is $\mathcal{ALCOQH}$ extended with the top role, role negation, role conjunction and role disjunction.

4 The Implementation

FAME(Q) is a Java-based implementation of the forgetting method described in the previous section. In this section we describe the implementation in detail, and discuss some of its notable features. For users' convenience, FAME(Q) provides a graphic user interface, shown in Figure 1. FAME 1.0 [18] is a preceding system for forgetting in description logics without number restrictions, but since the inference rules used by FAME(Q) are different from those in FAME 1.0, FAME(Q) is not simply an improvement of FAME 1.0, but is a novel system.

FAME(Q) has a modular design consisting of six main modules: Load Ontology, Parse into Own Data Structure, Role Forgetting, Concept Forgetting, Unparse into OWL Data Structure, and Save Ontology, which are linked as depicted in Figure 2. Each successively undertakes a particular task. FAME(Q) uses the OWL

[5] In [17], the purify rules are named Ackermann rules I and II, and the combination rules are named the Ackermann rules III, IV and V.

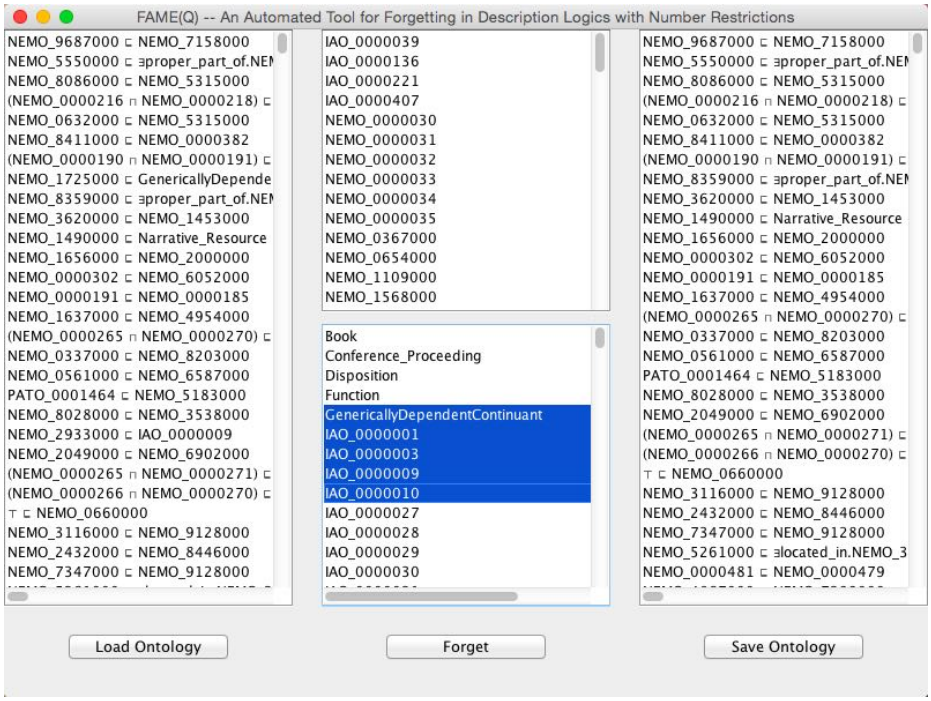

Fig. 1: Graphic User Interface of FAME(Q)

API Version 3.5.6[6] for the tasks of loading, parsing, unparsing and saving ontologies. The ontology to be loaded must be an OWL/XML file, or a URL pointing to an OWL/XML file. Internally (during the forgetting process), FAME(Q) uses own data structures to store and manipulate data so it can be processed efficiently.

4.1 Forgetting Process

Central to FAME(Q) are the Role Forgetting process and Concept Forgetting process, in which the role names and concept names in $\mathcal{F}$ are eliminated. FAME(Q) eliminates role names and concept names in a focused manner, that is, it performs role forgetting and concept forgetting separately. Although FAME(Q) can eliminate role and concept names in any specified order, it defaults to eliminating role names first. This is because during the concept forgetting process, role negation and role disjunction may be introduced, and the calculus for role name elimination does not support these two role constructs.

The role forgetting process is an iteration of several rounds in each of which a role name in the forgetting signature $\mathcal{F}$ is eliminated using the calculus for role name elimination. The concept forgetting process has two phases executed in sequence. In the first phase concept names in $\mathcal{F}$ are eliminated using only the

[6] http://owlcs.github.io/owlapi/

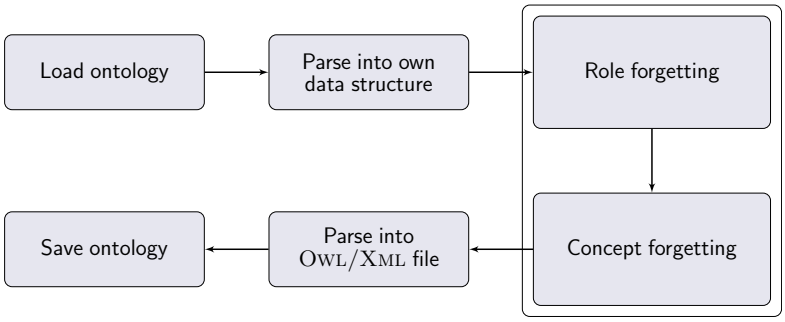

Fig. 2: Top-level design of FAME(Q)

purify rules. These iterations are intended to eliminate those concept names that are pure in $\mathcal{N}$. This is because purification does not require the ontology to be normalized or in reduced form, and thus is relatively cheap. Another reason is that purification introduces the top concept into clauses which are immediately simplified or eliminated; this makes subsequent forgetting less challenging. The second phase contains several rounds in each of which a concept name in the forgetting signature $\mathcal{F}$ is eliminated using not only the combination rule, but also the purify rules. This guarantees that all concept names in $\mathcal{F}$ are considered for elimination from $\mathcal{F}$.

Once a name has been eliminated from the clause set $\mathcal{N}$, it is removed from the forgetting signature $\mathcal{F}$. A name that cannot be eliminated in the current round may become eliminable after the elimination of another name [16]. The elimination rounds are therefore implemented in a do-while loop. The break condition checks if there were names eliminated in the previous rounds. If so, FAME(Q) repeats the iterations, attempting to eliminate the remaining names. The loop terminates when $\mathcal{F}$ becomes empty or no names were eliminated in the previous rounds.

Introduced definers are eliminated as part of the concept forgetting process using the calculus for concept name elimination. Unlike regular concept names, there is no guarantee that all definers can be eliminated. If the original ontology contains cyclic dependencies over the names in the forgetting signature $\mathcal{F}$, it may not be possible to eliminate all definers, see [17,19] for examples. This means the forgetting method is incomplete.

Our method can eliminate any concept and role names, though this is at the cost that the definers introduced may not be all eliminated. If the ontology computed by FAME(Q) does not contain any definers, we say that FAME(Q)/the forgetting is successful. In the successful cases, the forgetting solution is a uniform interpolant or a semantic solution.[7] If it is a uniform interpolant, it can be saved as an OWL/XML file. If it is a semantic solution, generally it cannot be

[7] Because in some cases a uniform interpolant and a semantic solution coincide, when we say a forgetting solution is a semantic solution, we means it is only a semantic solution but not a uniform interpolant.

saved as an OWL/XML file, because of extra expressivity such as role negation/-conjunction/disjunction being not supported by the OWL API. In these cases, the forgetting solutions are represented in the data structure of FAME(Q).

4.2 Frequency Count

A frequency counter is used in FAME(Q) to check the existence of each name of $\mathcal{F}$ in $\mathcal{N}$ and in each clause of $\mathcal{N}$, and count the frequency of positive and negative occurrences of the name in $\mathcal{N}$ and in each clause of $\mathcal{N}$. Algorithm 1 below computes the frequency counts of positive occurrences of a concept name, where AtomicConcept denotes a concept name, GreaterThan and LessThan denote the ternary number restriction operators $\geq$ and $\leq$, respectively, and Conjunction and Disjunction denote the n-nary operators of $\sqcap$ and $\sqcup$, respectively. The first operand of GreaterThan and LessThan is a positive integer and a non-negative integer, respectively, the second operand is a role name, and the third operand is a concept. The operands of Conjunction and Disjunction are concepts. Operands are stored in a list, an ordered collection of objects allowing duplicate values. We used a list, not a set, because the insertion order is preserved in a list, and allows positional access and insertion of elements. The algorithms (for counting negative frequency of a concept name and for counting positive frequency and negative frequency of a role name) were implemented similarly.

4.3 Definer Reuse

FAME(Q) reuses definers whenever possible. For example, consider the case of forgetting $A \in \mathsf{sig_C}(\mathcal{N})$ from $\mathcal{N}$, when a concept has been replaced by a specific definer in an A-clause, it is replaced uniformly by the definer in all A-clauses. We do not introduce new definer names for the same concept in other A-clauses. On the other hand, if a concept C has been replaced by a definer D, the concept of $\neg C$ is replaced (if necessary) by $\neg$D, rather than a fresh definer, that is, FAME(Q) introduces definers in a conservative manner (as few as possible). This significantly improves the efficiency of FAME(Q). Definers and the concepts replaced by them (the corresponding concepts) are stored as keys and values respectively in a Java HashMap, which allows for easy insertion and retrieval of paired elements.

5 The Evaluation

In order to understand the practicality and usefulness of FAME(Q), we evaluated the current version on a corpus of ontologies taken from the NCBO BioPortal repository,[8] a resource currently including more than 600 ontologies originally developed for clinical research. The corpus was based on a snapshot of the repository taken in March 2017 [11], containing 396 OWL API compatible ontologies. Statistical information about these ontologies can be found in [18].

[8] https://bioportal.bioontology.org/

Algorithm 1: POSITIVE(A, cls)

```
   Input  : a concept name A
            a clause cls
   Output: an integer i
 1 if cls instance of AtomicConcept then
 2 |   if cls equals to A then
 3 |   |   return 1;
 4 |   else
 5 |   |   return 0;
 6 else if cls instance of Negation then
 7 |   Clause operand = cls.getOperands().get(0);
 8 |   return NEGATIVE(A, operand);
 9 else if cls instance of GreaterThan or LessThan then
10 |   Clause operand = cls.getOperands().get(1);
11 |   return POSITIVE(A, operand);
12 else if cls instance of Conjunction or Disjunction then
13 |   initialize Integer sum to 0;
14 |   List<Clause> operand_list = cls.getOperands();
15 |   foreach clause operand in operand_list do
16 |   |   sum = sum + POSITIVE(A, operand);
17 |   end
18 |   return sum;
19 else
20 |   return 0;
```

Table 1 lists the types of axioms handled by FAME(Q). All these can be encoded as SubClassOf axioms. Axioms not expressible in $\mathcal{ALCOQH}$ were removed from each ontology as FAME(Q) only accommodated $\mathcal{ALCOQH}$-ontologies.

To reflect real-world application scenarios, we evaluated the performance of FAME(Q) for forgetting different numbers of concept names and role names from each ontology. We considered the cases of forgetting 10%, 30% and 50% of concept and role names from the signature of each ontology. LETHE was the only existing tool for forgetting in description logics with number restrictions; it handled $\mathcal{ALCQH}$ but only for concept forgetting. We compared the results of concept forgetting computed by FAME(Q) with those by LETHE on the $\mathcal{ALCQH}$-fragments. The fragments were obtained similarly as for the $\mathcal{ALCOQH}$-fragments. In order to allow a fair comparison with LETHE which was evaluated on randomly chosen forgetting signatures we did the same. The experiments were run on a desktop with an Intel® Core™ i7-4790 processor, four cores running at up to 3.60 GHz, and 8 GB of DDR3-1600 MHz RAM. The experiments were run 100 times on each ontology and we averaged the results in order to verify the accuracy of our findings. A timeout of 1000 seconds was imposed on each run.

The results obtained from forgetting 10%, 30% and 50% of concept names and role names from the $\mathcal{ALCOQH}$-ontologies are shown in Table 2, where one can observe that, on average, FAME(Q) was successful in nearly 90% of the test

	Type of Axiom	Representation
TBox	SubClassOf(C1 C2)	SubClassOf(C1 C2)
	EquivalentClasses(C1 C2)	SubClassOf(C1 C2), SubClassOf(C2 C1)
	DisjointClasses(C1 C2)	SubClassOf(C1 ObjectComplementOf(C2))
	DisjointUnion(C C1...Cn)	EquivalentClasses(C ObjectUnionOf(C1...Cn)) DisjointClasses(C1...Cn)
	SubObjectPropertyOf(R1 R2)	SubObjectPropertyOf(R1 R2)
	EquivalentObjectProperties(R1 R2)	SubObjectPropertyOf(R1 R2) SubObjectPropertyOf(R2 R1)
	ObjectPropertyDomain(R C)	SubClassOf(ObjectSomeValuesFrom(R owl:Thing), C)
	ObjectPropertyRange(R C)	SubClassOf(owl:Thing ObjectAllValuesFrom(R C))
ABox	ClassAssertion(C a)	SubClassOf(a C)
	ObjectPropertyAssertion(R a1 a2)	SubClassOf(a1 ObjectSomeValuesFrom(R a2))

Table 1: Types of axioms that can be handled by FAME(Q)

Settings		Results				
Forgetting	**Forget %**	**Time**	**Timeout**	**Success Rate**	**N_D Left**	$\forall, \neg, \sqcap, \sqcup$
Concept Forgetting	10%	3.1 sec.	1.3%	96.2%	2.5%	10.6%
	30%	9.0 sec.	4.0%	89.7%	6.3%	31.6%
	50%	14.2 sec.	7.5%	83.7%	8.8%	53.3%
	Avg.	8.8 sec.	4.3%	89.8%	5.9%	31.8%
Role Forgetting	10%	4.0 sec.	1.5%	96.7%	1.8%	18.3%
	30%	9.1 sec.	4.6%	90.1%	5.3%	25.3%
	50%	15.2 sec.	7.8%	82.7%	9.5%	41.9%
	Avg.	9.5 sec.	4.7%	89.8%	5.5%	25.4%

Table 2: Results of concept and role forgetting computed by FAME(Q)

cases (89.8% for both concept forgetting and role forgetting). In most successful cases, the forgetting solutions were computed within 10 seconds (8.8 seconds for concept forgetting and 9.5 seconds for role forgetting). The column headed N_D Left shows the percentages of the test cases where the definers were present in the resulting ontologies. The column headed $\forall, \neg, \sqcap, \sqcup$ shows the percentages of the test cases where the forgetting solutions involved role constructs.

According to the results FAME(Q) was considerably faster than LETHE on the $\mathcal{ALCQH}$-fragments; on average, it was 8 times faster. An important reason is that LETHE introduces definers in a systematic and exhaustive manner. The column headed N_D Intro shows the percentages of the test cases where definers were introduced during the forgetting process. It can be seen that LETHE introduced definers in nearly 100% of the test cases. In addition, FAME(Q) attained notably better success rates over LETHE (90.5% over 79.0%). Most failures of LETHE were due to the timeout.

Another advantage is that solutions computed by FAME(Q) are in general stronger than those by LETHE. Often, a stronger solution means a better one. For

Settings		Results					
Tool	**Forget%**	**Time**	**Timeout**	**Success Rate**	**$\mathsf{N_D}$ Intro**	$\forall, \neg, \sqcap, \sqcup$	**Fixpoints**
FAME(Q) $\mathcal{ALCQH}$	10%	2.9 sec.	1.0%	96.2%	16.3%	10.6%	0.0%
	30%	7.5 sec.	3.5%	89.7%	27.2%	31.6%	0.0%
	50%	7.4 sec.	6.7%	83.7%	35.8%	53.3%	0.0%
	Avg.	8.1 sec.	3.4%	89.8%	5.9%	31.8%	0.0%
LETHE $\mathcal{ALCQH}$	10%	25.2 sec.	7.4%	92.6%	97.2%	0.0%	11.4%
	30%	59.5 sec.	20.5%	79.5%	100.0%	0.0%	14.9%
	50%	91.7 sec.	35.1%	64.9%	100.0%	0.0%	18.2%
	Avg.	58.8 sec.	21.0%	79.0%	99.1%	0.0%	14.8%

Table 3: Results of concept forgetting computed by FAME(Q) and LETHE

example, the solution of forgetting the concept name $\{\mathsf{Male}\}$ from the ontology

$$\{\mathsf{A} \sqsubseteq \geq 2\mathsf{hasSon}.\mathsf{Male},\ \mathsf{A} \sqsubseteq \geq 3\mathsf{hasDaughter}.\neg\mathsf{Male},\\ \mathsf{hasSon} \sqsubseteq \mathsf{hasChild},\ \mathsf{hasDaughter} \sqsubseteq \mathsf{hasChild}\}$$

computed by LETHE is

$$\{\mathsf{A} \sqsubseteq \geq 2\mathsf{hasSon}.\top,\ \mathsf{A} \sqsubseteq \geq 3\mathsf{hasDaughter}.\top,\\ \mathsf{hasSon} \sqsubseteq \mathsf{hasChild},\ \mathsf{hasDaughter} \sqsubseteq \mathsf{hasChild}\},$$

while the solution of FAME(Q) includes an additional axiom

$$\mathsf{A} \sqsubseteq \geq 5(\mathsf{hasSon} \sqcup \mathsf{hasDaughter}).\top,$$

where role disjunction is used. Upon the solution of LETHE, if we further forget the role names hasSon and hasDaughter, the uniform interpolant is $\{\mathsf{A} \sqsubseteq \geq 3\mathsf{hasChild}.\top\}$, while on the intermediary solution of FAME(Q), the solution is $\{\mathsf{A} \sqsubseteq \geq 5\mathsf{hasChild}.\top\}$, which is stronger and closer to the fact: A has at least 5 children. This shows an advantage of semantic forgetting where extra expressivity allows intermediary information ($\mathsf{A} \sqsubseteq \geq 5(\mathsf{hasSon} \sqcup \mathsf{hasDaughter}).\top$) to be captured which produces a better solution.

For users such as SNOMED CT and NCIT who do not have the flexibility to easily switch to a more expressive language, or are bound by the application, the available support and tooling, to a specific language, FAME(Q) is not satisfactory. Tracking the logical difference between different versions of ontologies is an application where the target language should coincide with the source language. In these cases, LETHE would be more suited.

6 Conclusions

This paper describes the tool of FAME(Q) for forgetting in $\mathcal{ALCOQH}$-ontologies. FAME(Q) is at present the only tool able to forget concept and role names in description logics with number restrictions. Compared to LETHE, a tool that can perform concept forgetting in $\mathcal{ALCQH}$, FAME(Q) fared better with respect to success rates and time efficiency on $\mathcal{ALCQH}$-fragments of realistic ontologies.

References

1. J. Bicarregui, T. Dimitrakos, D. M. Gabbay, and T. S. E. Maibaum. Interpolation in practical formal development. *Logic Journal of the IGPL*, 9(2):231–244, 2001.
2. W. M. Del-Pinto and R. A. Schmidt. ABox Abduction via Forgetting in $\mathcal{ALC}$. In *Proc. AAAI'19*. AAAI Press, 2019. To appear.
3. T. Eiter, G. Ianni, R. Schindlauer, H. Tompits, and K. Wang. Forgetting in managing rules and ontologies. In *Web Intelligence*, pages 411–419. IEEE Computer Society, 2006.
4. D. M. Gabbay, R. A. Schmidt, and A. Szałas. *Second Order Quantifier Elimination: Foundations, Computational Aspects and Applications*. College Publications, 2008.
5. B. C. Grau and B. Motik. Reasoning over ontologies with hidden content: The import-by-query approach. *J. Artif. Intell. Res.*, 45:197–255, 2012.
6. B. Konev, D. Walther, and F. Wolter. Forgetting and Uniform Interpolation in Large-Scale Description Logic Terminologies. In *Proc. IJCAI'09*, pages 830–835. IJCAI/AAAI Press, 2009.
7. P. Koopmann. *Practical Uniform Interpolation for Expressive Description Logics*. PhD thesis, University of Manchester, UK, 2015.
8. P. Koopmann and R. A. Schmidt. Count and Forget: Uniform Interpolation of $\mathcal{SHQ}$-Ontologies. In *Proc. IJCAR'14*, volume 8562 of *Lecture Notes in Computer Science*, pages 434–448. Springer, 2014.
9. P. Koopmann and R. A. Schmidt. LETHE: Saturation-Based Reasoning for Non-Standard Reasoning Tasks. In *Proc. DL'15*, volume 1387 of *CEUR Workshop Proceedings*, pages 23–30. CEUR-WS.org, 2015.
10. J. Lang, P. Liberatore, and P. Marquis. Propositional independence: Formula-variable independence and forgetting. *J. Artif. Intell. Res.*, 18:391–443, 2003.
11. N. Matentzoglu and B. Parsia. BioPortal Snapshot 30.03.2017, Mar. 2017.
12. G. Qi, Y. Wang, P. Haase, and P. Hitzler. A Forgetting-based Approach for Reasoning with Inconsistent Distributed Ontologies. In *Proc. WoMO'08*, volume 348 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.
13. K. Wang, G. Antoniou, R. W. Topor, and A. Sattar. Merging and Aligning Ontologies in DL-Programs. In *Proc. RuleML'05*, volume 3791 of *Lecture Notes in Computer Science*, pages 160–171. Springer, 2005.
14. K. Wang, Z. Wang, R. W. Topor, J. Z. Pan, and G. Antoniou. Eliminating concepts and roles from ontologies in expressive descriptive logics. *Computational Intelligence*, 30(2):205–232, 2014.
15. Y. Zhao, G. Alghamdi, R. A. Schmidt, H. Feng, G. Stoilos, D. Juric, and M. Khodadadi. Tracking Logical Difference in Large-Scale Ontologies: A Forgetting-Based Approach. In *Proc. AAAI'19*. AAAI Press, 2019.
16. Y. Zhao and R. A. Schmidt. Forgetting Concept and Role Symbols in $\mathcal{ALCOIH}\mu^+(\nabla, \sqcap)$-Ontologies. In *Proc. IJCAI'16*, pages 1345–1352. IJCAI/AAAI Press, 2016.
17. Y. Zhao and R. A. Schmidt. Role Forgetting for $\mathcal{ALCOQH}(\nabla)$-Ontologies Using An Ackermann-Based Approach. In *Proc. IJCAI'17*, pages 1354–1361. IJCAI/AAAI Press, 2017.
18. Y. Zhao and R. A. Schmidt. FAME: An Automated Tool for Semantic Forgetting in Expressive Description Logics. In *Proc. IJCAR'18*, volume 10900 of *Lecture Notes in Computer Science*, pages 19–27. Springer, 2018.
19. Y. Zhao and R. A. Schmidt. On Concept Forgetting in Description Logics with Qualified Number Restrictions. In *Proc. IJCAI'18*, pages 1984–1990. IJCAI/AAAI Press, 2018.