The FAME Family – A Family of Reasoning Tools for Forgetting in Expressive Description Logics

Yizheng Zhao[1], Hao Feng[2], Ruba Alassaf[1], Warren Del-Pinto[1] and Renate A. Schmidt[1]

[1] The University of Manchester, UK
[2] North China University of Science and Technology, China

Abstract. We present the FAME family – a family of reasoning tools for forgetting in expressive description logics. *Forgetting* is a non-standard reasoning service that seeks to create restricted views of (description logic-based) ontologies by eliminating a set of concept and role names, namely the *forgetting signature*, from the ontologies in such a way that all logical consequences are preserved up to the remaining signature. The family has two members at present: (i) FAME 1.0 for computing semantic solutions of forgetting in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$, and (ii) FAME 2.0 for computing uniform interpolants in the description logic $\mathcal{ALCOQH}(\triangledown, \neg, \sqcap, \sqcup)$. They are Java-based implementations of respectively two Ackermann-based forgetting methods developed in our recent work. Both can be used as standalone tools or Java libraries for forgetting and related tasks. An empirical evaluation of FAME 1.0 and 2.0 on a large corpus of real-world ontologies shows that the tools are practical.

1 Introduction

Ontologies, exploiting Description Logics as the representational underpinning, provide a logic-based data model for representation of domain knowledge thereby supporting effective reasoning over domain knowledge for a range of real-world applications, most evidently for applications in life sciences, text mining, as well as the Semantic Web. However, with their growing utilisation, not only has the number of available ontologies increased considerably, but they are often large in size and are becoming more complex to manage. Moreover, modelling domain knowledge in the form of ontologies is labour-intensive work which is expensive from an implementation perspective. There is therefore a strong demand for technologies and automated tools for creating restricted views of ontologies so that existing ontologies can be reused to their full potential. *Forgetting* is a non-standard reasoning service that seeks to create restricted views of ontologies by eliminating a set of concept and role names, namely the *forgetting signature*, from the ontologies in such a way that all logical consequences are preserved up to the remaining signature. It allows users to focus on specific parts of ontologies that can be easily reused, or to zoom in on ontologies for in-depth analysis of certain subparts. Other uses of forgetting can be found in [9,16,7,13,20,18,4,3].

Forgetting can be defined in two closely related ways; it can be defined syntactically as the dual of *uniform interpolation* [19] (related notions include weak forgetting [21], consequence-based inseparability [15] and consequence-based conservative extensions [6]) and it can be defined model-theoretically as *semantic forgetting* [20] (related notions include strong forgetting [12], model inseparability [8], model conservative extensions [14] and second-order quantifier elimination [5]). The two notions differ in the sense that uniform interpolation preserves all *logical consequences* up to certain names whereas semantic forgetting preserves *equivalence* up to certain names. In this sense, semantic forgetting is a stronger notion of forgetting than uniform interpolation, and the results of semantic forgetting, namely the *semantic solutions*, are in general stronger than those of uniform interpolation, namely the *uniform interpolants*. Semantic solutions often require more expressivity than is available in the source language. For example, the semantic solution of forgetting the role name $\{r\}$ from the $\mathcal{ALC}$-ontology $\{A_1 \sqsubseteq \exists r.B, A_2 \sqsubseteq \forall r.\neg B\}$ is $\{A_1 \sqsubseteq \exists \triangledown.B, A_1 \sqcap A_2 \sqsubseteq \bot\}$, whereas the uniform interpolant is $\{A_1 \sqcap A_2 \sqsubseteq \bot\}$, which is weaker. Observe that the target language must include the universal role $\triangledown$ to represent the semantic solution. If a semantic solution is expressible in the source logic, then it is equivalent to the uniform interpolant, which means, in this case, the two notions coincide [21].

In this paper, we describe the FAME family – a family of reasoning tools for forgetting in expressive description logics. The FAME family has two members at present, namely, FAME 1.0 and FAME 2.0, which are Java-based implementations of respectively the methods of [23,24] for computing semantic solutions of forgetting in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$, and the methods of [25,26] for computing uniform interpolants in the description logic $\mathcal{ALCOQH}(\triangledown, \neg, \sqcap, \sqcup)$. The foundations for the methods are non-trivial generalisations of *Ackermann's Lemma* [1]. The universal role and the Boolean role constructors enrich the target languages, rendering them more expressive to represent forgetting solutions. FAME 1.0 is the first tool for computing semantic solutions of forgetting in description logics. FAME 2.0 is the first tool for computing uniform interpolants in description logics with qualified number restrictions plus nominals and ABoxes. Both tools have been evaluated on a large corpus of real-world ontologies, with the results showing that: (i) in more than 90% of the test cases the tools were successful, i.e., eliminated all specified concept and role names and the possibly introduced definers, and (ii) in more than 70% of the successful cases the elimination was done within seconds. The current versions of FAME 1.0 and 2.0 can be downloaded via `http://www.cs.man.ac.uk/~schmidt/sf-fame/`.

2 $\mathcal{ALCOIH}(\triangledown, \sqcap)$ and $\mathcal{ALCOQH}(\triangledown, \neg, \sqcap, \sqcup)$

Let $\mathsf{N_C}$, $\mathsf{N_R}$ and $\mathsf{N_I}$ be three countably infinite and pairwise disjoint sets of *concept names*, *role names* and *individual names* (*nominals*), respectively. *Roles* in $\mathcal{ALCOIH}(\triangledown, \sqcap)$ can be a role name $r \in \mathsf{N_R}$, the inverse r^- of a role name r, the universal role $\triangledown$, or can be formed with conjunction. *Concepts* in $\mathcal{ALCOIH}(\triangledown, \sqcap)$ have one of the following forms: $\top \mid \bot \mid a \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C$,

where $a \in \mathsf{N_I}$, $A \in \mathsf{N_C}$, C and D are arbitrary concepts, and R is an arbitrary role. *Roles* in $\mathcal{ALCOQH}(\triangledown, \neg, \sqcap, \sqcup)$ can be a role name $r \in \mathsf{N_R}$, the universal role $\triangledown$, or can be formed with negation, conjunction and disjunction. *Concepts* in $\mathcal{ALCOQH}(\triangledown, \neg, \sqcap, \sqcup)$ 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 $a \in \mathsf{N_I}$, $A \in \mathsf{N_C}$, C and D are arbitrary concepts, R is an arbitrary role, 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 called *(qualified) number restrictions*, which allow one to specify cardinality constraints on roles.

An $\mathcal{ALCOIH}(\triangledown, \sqcap)$- or $\mathcal{ALCOQH}(\triangledown, \neg, \sqcap, \sqcup)$-ontology comprises 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 any 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 any concept. ABox assertions are superfluous in description logics with nominals, because they can be expressed equivalently as concept inclusions (via nominals), namely, $C(a)$ as $a \sqsubseteq C$ and $r(a, b)$ as $a \sqsubseteq \exists r.b$. Hence, an $\mathcal{ALCOIH}(\triangledown, \sqcap)$- or $\mathcal{ALCOQH}(\triangledown, \neg, \sqcap, \sqcup)$-ontology is assumed to contain only TBox and RBox axioms. The semantics of $\mathcal{ALCOIH}(\triangledown, \sqcap)$ and $\mathcal{ALCOQH}(\triangledown, \neg, \sqcap, \sqcup)$ is defined as usual.

The forgetting methods used by FAME 1.0 and 2.0 work on TBox and RBox axioms in clausal normal form. A *TBox literal in* $\mathcal{ALCOIH}(\triangledown, \sqcap)$ is a concept of the form a, $\neg a$, A, $\neg A$, $\exists R.C$ or $\forall R.C$. A *TBox literal in* $\mathcal{ALCOQH}(\triangledown, \neg, \sqcap, \sqcup)$ is a concept of the form a, $\neg a$, A, $\neg A$, $\geq mR.C$ or $\leq nR.C$. A *TBox clause* is a disjunction of a finite number of TBox literals. An *RBox atom in* $\mathcal{ALCOIH}(\triangledown, \sqcap)$ is a role name, an inverted role name or the universal role. An *RBox atom in* $\mathcal{ALCOQH}(\triangledown, \neg, \sqcap, \sqcup)$ is a role name or the universal role. An *RBox clause* is a disjunction of an RBox atom and a negated RBox atom. TBox and RBox clauses are obtained from corresponding axioms using the standard clausal normal form transformations.

Let $\mathcal{S} \in \mathsf{N_C}$ ($\mathcal{S} \in \mathsf{N_R}$) be a designated concept (role) name. We say that an occurrence of $\mathcal{S}$ is *positive* (*negative*) in an axiom or clause if it is under an *even* (*odd*) number of negations. An axiom (clause) is called an *$\mathcal{S}$-axiom* (*$\mathcal{S}$-clause*) if it contains $\mathcal{S}$. An axiom (clause) is called an *$\mathcal{S}^+$-axiom* (*$\mathcal{S}^+$-clause*) or *$\mathcal{S}^-$-axiom* (*$\mathcal{S}^-$-clause*) if it contains positive (negative) occurrences of $\mathcal{S}$. $\mathcal{S}$ is said to be *pure* in a set $\mathcal{N}$ of clauses if either all occurrences of $\mathcal{S}$ in $\mathcal{N}$ are positive, or all occurrences of $\mathcal{S}$ in $\mathcal{N}$ are negative. It is said to be *impure* otherwise.

By $\mathsf{sig_C}(X)$, $\mathsf{sig_R}(X)$ and $\mathsf{sig_I}(X)$, we denote respectively the sets of the concept names, role names and individual names occurring in X, where X ranges over concepts, roles, clauses, axioms, sets of clauses and ontologies. By $\mathsf{sig}(X)$ we denote the union of $\mathsf{sig_C}(X)$ and $\mathsf{sig_R}(X)$. Let $\mathcal{S} \in \mathsf{N_C}$ ($\mathcal{S} \in \mathsf{N_R}$) be any concept (role) name, and let $\mathcal{I}$ and $\mathcal{I}'$ be any interpretations. We say that $\mathcal{I}$ and $\mathcal{I}'$ are *equivalent up to* $\mathcal{S}$, or *$\mathcal{S}$-equivalent*, if $\mathcal{I}$ and $\mathcal{I}'$ coincide but differ possibly in the interpretation of $\mathcal{S}$. More generally, 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 1 (Semantic Forgetting). *Let $\mathcal{O}$ be an ontology and let $\mathcal{F} \subseteq \mathsf{sig}(\mathcal{O})$ be the forgetting signature. An ontology $\mathcal{O}'$ is a* semantic solution *of forgetting $\mathcal{F}$ from $\mathcal{O}$ iff the following conditions hold: (i) $\mathsf{sig}(\mathcal{O}') \subseteq \mathsf{sig}(\mathcal{O})\backslash\mathcal{F}$, and (ii) for any interpretation $\mathcal{I}$: $\mathcal{I} \models \mathcal{O}'$ iff $\mathcal{I}' \models \mathcal{O}$, for some interpretation $\mathcal{I}'$ $\mathcal{F}$-equivalent to $\mathcal{I}$, i.e., O and $\mathcal{O}'$ are equivalent up to the interpretations of the names in $\mathcal{F}$.*

Definition 2 (Uniform Interpolation). *Let $\mathcal{O}$ be an ontology and let $\mathcal{F} \subseteq \mathsf{sig}(\mathcal{O})$ be the forgetting signature. An ontology $\mathcal{O}'$ is a* uniform interpolant *of $\mathcal{O}$ for $\mathsf{sig}(\mathcal{O})\backslash\mathcal{F}$ iff the following conditions hold: (i) $\mathsf{sig}(\mathcal{O}') \subseteq \mathsf{sig}(\mathcal{O})\backslash\mathcal{F}$, and (ii) for any axiom α with $\mathsf{sig}(\alpha) \subseteq \mathsf{sig}(\mathcal{O})\backslash\mathcal{F}$, $\mathcal{O}' \models \alpha$ iff $\mathcal{O} \models \alpha$, i.e., $\mathcal{O}'$ preserves all logical consequences over the names in $\mathsf{sig}(\mathcal{O})\backslash\mathcal{F}$.*

In this paper, the notation $\mathcal{F}$ is used to denote the forgetting signature, i.e., the set of concept and role names to be forgotten, and $\mathcal{F}_{\mathsf{C}}$ and $\mathcal{F}_{\mathsf{R}}$ are used to denote respectively the sets of the concept names and the role names in $\mathcal{F}$. The name in $\mathcal{F}$ under current consideration for forgetting is referred to as the *pivot*.

3 The Calculi

FAME 1.0 and 2.0 eliminate concept and role names in a goal-oriented manner. In this section, we briefly go through the calculi used respectively by FAME 1.0 and 2.0 for eliminating a concept and a role name from an $\mathcal{ALCOIH}(\triangledown, \sqcap)$- and an $\mathcal{ALCOQH}(\triangledown, \neg, \sqcap, \sqcup)$-ontology (in clausal form). For proofs and other details of the calculi, we refer to reader to the original work [23,24,25,26].

3.1 Eliminating One Concept Name in $\mathcal{ALCOIH}(\triangledown, \sqcap)$

The calculus used by FAME 1.0 for eliminating a concept name from a set of $\mathcal{ALCOIH}(\triangledown, \sqcap)$-clauses includes three types of rules: (i) two purify rules, (ii) two Ackermann rules, and (iii) four rewrite rules.

The purify rules are applied to eliminate a concept name $A \in \mathrm{sig}_{\mathsf{C}}(\mathcal{N})$ from a set $\mathcal{N}$ of clauses when A is pure in $\mathcal{N}$. Specifically, A is eliminated from $\mathcal{N}$ by substituting $\top$ ($\bot$) for every occurrence of A in $\mathcal{N}$ if all occurrences of A in $\mathcal{N}$ are positive (negative). This is referred to as *purification*. The Ackermann rules, shown in Figure 1, are applied to eliminate a concept name $A \in \mathrm{sig}_{\mathsf{C}}(\mathcal{N})$ from a set $\mathcal{N}$ of clauses when A is impure in $\mathcal{N}$. By $\mathcal{N}\backslash\{C_1 \sqcup A, ..., C_n \sqcup A\}$ we denote the set $\mathcal{N}$ excluding the clauses $C_1 \sqcup A, ..., C_n \sqcup A$. By $\mathcal{N}^A_\alpha$ we denote the clause set obtained from $\mathcal{N}$ by substituting α for every occurrence of A in $\mathcal{N}$, where α is a concept. While the purify rules can be applied at any time, the Ackermann rules can only be applied if $\mathcal{N}$ is in *A-reduced form*.

Ackermann$^{\mathsf{C},+}$

$$\frac{\mathcal{N}\backslash\{C_1 \sqcup A, ..., C_n \sqcup A\}, C_1 \sqcup A, ..., C_n \sqcup A}{\mathcal{N}\backslash\{C_1 \sqcup A, ..., C_n \sqcup A\}^{A}_{\neg C_1 \sqcup ... \sqcup \neg C_n}}$$

provided: (i) $A \in \mathrm{sig}_{\mathsf{C}}(\mathcal{N})$ is the pivot,
(ii) the C_i $(1 \leq i \leq n)$ do not contain A, and
(iii) A occurs only negatively in $\mathcal{N}\backslash\{C_1 \sqcup A, ..., C_n \sqcup A\}$.

Ackermann$^{\mathsf{C},-}$

$$\frac{\mathcal{N}\backslash\{C_1 \sqcup \neg A, ..., C_n \sqcup \neg A\}, C_1 \sqcup \neg A, ..., C_n \sqcup \neg A}{\mathcal{N}\backslash\{C_1 \sqcup \neg A, ..., C_n \sqcup \neg A\}^{A}_{C_1 \sqcap ... \sqcap C_n}}$$

provided: (i) $A \in \mathrm{sig}_{\mathsf{C}}(\mathcal{N})$ is the pivot,
(ii) A does not occur in the C_i $(1 \leq i \leq n)$, and
(iii) A occurs only positively in $\mathcal{N}\backslash\{C_1 \sqcup \neg A, ..., C_n \sqcup \neg A\}$.

Fig. 1: Ackermann rules for eliminating $A \in \mathrm{sig}_{\mathsf{C}}(\mathcal{N})$ from a set $\mathcal{N}$ of clauses

Definition 3 (A-Reduced Form for $\mathcal{ALCOIH}(\triangledown, \sqcap)$). *For $A \in \mathsf{N_C}$ the pivot, a clause is in A^+-reduced form if it has the form $C \sqcup A$, and is in A^--reduced form if it has the form $C \sqcup \neg A$, where C is a clause that does not contain A. A set $\mathcal{N}$ of clauses is in A-reduced form if all A^+-clauses in $\mathcal{N}$ are in A^+-reduced form, or all A^--clauses in $\mathcal{N}$ are in A^--reduced form.*

The rewrite rules attempt to transform an A-clause (not in A-reduced form) into an equivalent one in A-reduced form. Note that using the present calculus, a concept name cannot always be eliminated. This is because there is a gap in the scope of the rewrite rules, which means that transforming a clause set into A-reduced form is not always possible.

Theorem 1. *Let $\mathcal{I}$ be an $\mathcal{ALCOIH}(\triangledown, \sqcap)$-interpretation. For $A \in \mathsf{N_C}$ the pivot, when an Ackermann/a purify rule is applicable to a set $\mathcal{N}$ of $\mathcal{ALCOIH}(\triangledown, \sqcap)$-clauses, the conclusion of the rule is true in $\mathcal{I}$ iff for some interpretation $\mathcal{I}'$ A-equivalent to $\mathcal{I}$, the premises are true in $\mathcal{I}'$, i.e., the conclusion of the rule is a semantic solution of forgetting A from $\mathcal{N}$.*

Theorem 2. *The rewrite rules preserve equivalence.*

3.2 Eliminating One Role Name in $\mathcal{ALCOIH}(\triangledown, \sqcap)$

Slightly different from the calculus described above for concept name elimination, the calculus used by FAME 1.0 for eliminating a role name from a set of $\mathcal{ALCOIH}(\triangledown, \sqcap)$-clauses includes four types of rules: (i) two purify rules, (ii) one Ackermann rule, (iii) three rewrite rules, and (iv) four definer introduction rules.

Ackermann$^{\mathsf{R}}$

$$\frac{\mathcal{N}, \overbrace{C_1 \sqcup \exists(r \sqcap Q_1).D_1, \ldots, C_m \sqcup \exists(r \sqcap Q_m).D_m}^{\mathcal{P}^+_{\mathcal{T}}(r)}, \overbrace{\neg s_1 \sqcup r, \ldots, \neg s_v \sqcup r}^{\mathcal{P}^+_{\mathcal{R}}(r)}, \overbrace{E_1 \sqcup \forall r.F_1, \ldots, E_n \sqcup \forall r.F_n}^{\mathcal{P}^-_{\mathcal{T}}(r)}, \overbrace{\neg r \sqcup t_1, \ldots, \neg r \sqcup t_w}^{\mathcal{P}^-_{\mathcal{R}}(r)}}{\mathcal{N}, \mathbf{Block}(\mathcal{P}^-(r), C_1 \sqcup \exists(r \sqcap Q_1).D_1), ..., \mathbf{Block}(\mathcal{P}^-(r), C_m \sqcup \exists(r \sqcap Q_m).D_m), \mathbf{Block}(\mathcal{P}^-(r), \neg s_1 \sqcup r), ..., \mathbf{Block}(\mathcal{P}^-(r), \neg s_v \sqcup r)}$$

provided: (i) $r \in \mathsf{N_R}$ is the pivot,
(ii) r does not occur in $\mathcal{N}$, and
(iii) the premises are in r-reduced form.

Notation in the Ackermann$^{\mathsf{R}}$ rule ($1 \leq i \leq m$, $1 \leq j \leq n$, $1 \leq k \leq v$, $1 \leq l \leq w$):

1. $\mathbf{Block}(\mathcal{P}^-(r), C_i \sqcup \exists(r \sqcap Q_i).D_i)$ denotes the union of the following sets:

Ground-tier: $\{C_i \sqcup \exists(t_1 \sqcap \ldots \sqcap t_w \sqcap Q_i).D_i\}$

1st-tier: $\bigcup_{1 \leq j \leq n} \{C_i \sqcup E_j \sqcup \exists(t_1 \sqcap \ldots \sqcap t_w \sqcap Q_i).(D_i \sqcap F_j)\}$

2nd-tier: $\bigcup_{1 \leq j_1 \leq j_2 \leq n} \{C_i \sqcup E_{j_1} \sqcup E_{j_2} \sqcup \exists(t_1 \sqcap \ldots \sqcap t_w \sqcap Q_i).(D_i \sqcap F_{j_1} \sqcap F_{j_2})\}$

...

nth-tier: $\{C_i \sqcup E_1 \sqcup \ldots \sqcup E_n \sqcup \exists(t_1 \sqcap \ldots \sqcap t_w \sqcap Q_i).(D_i \sqcap F_1 \sqcap \ldots \sqcap F_n)\}$.

2. $\mathbf{Block}(\mathcal{P}^-(r), \neg s_k \sqcup r)$ denotes the following set:

$\{E_1 \sqcup \forall s_k.F_1, \ldots, E_n \sqcup \forall s_k.F_n, \neg s_k \sqcup t_1, \ldots, \neg s_k \sqcup t_w\}$.

$\star$ $E_j = \begin{cases} \bot & \text{if } \mathcal{P}^-_{\mathcal{T}} = \emptyset \\ E_j & \text{otherwise,} \end{cases}$ $F_j = \begin{cases} \top & \text{if } \mathcal{P}^-_{\mathcal{T}} = \emptyset \\ F_j & \text{otherwise, and} \end{cases}$ $t_l = \begin{cases} \triangledown & \text{if } \mathcal{P}^-_{\mathcal{R}} = \emptyset \\ t_l & \text{otherwise.} \end{cases}$

Fig. 2: Ackermann rule for eliminating $r \in \mathrm{sig}_{\mathsf{R}}(\mathcal{N})$ from a set $\mathcal{N}$ of clauses

The purify rules eliminate a role name $r \in \mathrm{sig}_{\mathsf{R}}(\mathcal{N})$ from a set $\mathcal{N}$ of clauses when r is pure in $\mathcal{N}$. Specifically, r is eliminated from $\mathcal{N}$ by substituting the universal role (the empty role) for every occurrence of r in $\mathcal{N}$ if all occurrences of r in $\mathcal{N}$ are positive (negative). When r is impure in $\mathcal{N}$, we apply the Ackermann rule, shown in Figure 2, to $\mathcal{N}$ to eliminate r. The Ackermann rule is applicable to $\mathcal{N}$ to eliminate r iff $\mathcal{N}$ is in *r-reduced form.*

Definition 4 (r-Reduced Form for $\mathcal{ALCOIH}(\triangledown, \sqcap)$). *For $r \in \mathsf{N_R}$ the pivot, a TBox clause is in r-reduced form if it has the form $C \sqcup \exists(r \sqcap Q).D$ or $C \sqcup \forall r.D$, where C (D) is a clause (concept) that does not contain r, and Q is a role that does not contain r. An RBox clause is in r-reduced form if it has the form $\neg S \sqcup r$ or $\neg r \sqcup S$, where $S \neq r$ is a role name, or $S \neq r^-$ is an inverted role name. A set $\mathcal{N}$ of clauses is in r-reduced form if all r-clauses in $\mathcal{N}$ are in r-reduced form.*

The rewrite rules and the definer introduction rules [22] attempt to transform an r-clause (not in r-reduced form) into r-reduced form. The transformation is not always possible; it fails in cases where r is reflexive, i.e., $r \sqsubseteq r^-$.

Theorem 3. *Let $\mathcal{I}$ be any $\mathcal{ALCOIH}(\triangledown, \sqcap)$-interpretation. For $r \in \mathsf{N_R}$ the pivot, when the Ackermann/a purify rule is applicable to a set $\mathcal{N}$ of $\mathcal{ALCOIH}(\triangledown, \sqcap)$-clauses, the conclusion of the rule is true in $\mathcal{I}$ iff for some interpretation $\mathcal{I}'$ r-equivalent to $\mathcal{I}$, the premises are true in $\mathcal{I}'$, i.e., the conclusion of the rule is a semantic solution of forgetting r from $\mathcal{N}$.*

Theorem 4. *The rewrite rules preserve equivalence. The definer introduction rules preserve equivalence up to the interpretation of the introduced definers.*

The calculi used by FAME 2.0 for eliminating a concept (role) name from a set of $\mathcal{ALCOQH}(\triangledown, \neg, \sqcap, \sqcup)$-clauses are analogous to the present calculus used by FAME 1.0 for role name elimination, with slight differences in the specifications of reduced forms and generalisations of Ackermann rules. For space reasons, we do not describe them in this paper. For those who are interested in the details of the calculi used by FAME 2.0, we refer to [25,26].

4 The Implementation

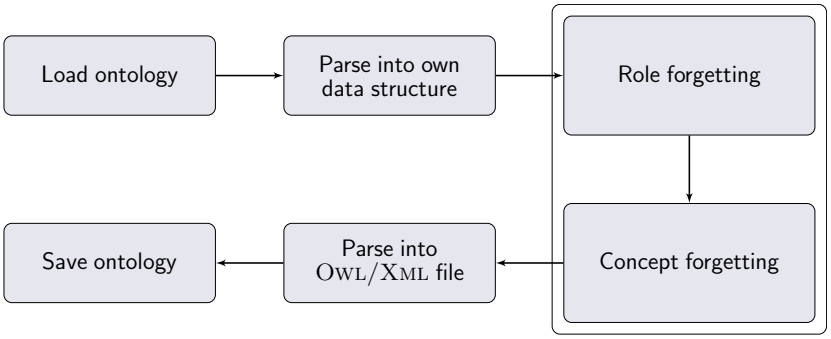

Fig. 3: Top-level design of FAME 1.0 and 2.0

FAME 1.0 and 2.0 were implemented in Java and have the same architecture, as shown in Figure 3. Hence in this section, we use the name "FAME" to refer to both versions. FAME uses the OWL API Version 3.5.6[3] for the tasks of loading, parsing and saving ontologies. The ontology to be loaded must be specified as an OWL/XML file, or as a URL pointing to an OWL/XML file, though, internally, FAME uses own data structure for efficiency.

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

Algorithm 1: FORGET(r_sig, c_sig, clause_set)

Input : a set r_sig of role names to be forgotten
a set c_sig of concept names to be forgotten
a set clause_set of clauses
Output: a set clause_set of clauses (after forgetting)

```
1  do
2      if r_sig is empty and c_sig is empty then
           /* clause_set does not contain any names in r_sig or c_sig; in
              this case, clause_set is a forgetting solution              */
3          return clause_set
4      end
5      initialising final int sig_size_before to (r_sig.size() + c_sig.size())
6      initialising Set⟨Name⟩ pure_sig to null
       /* get from r_sig and c_sig all names that are pure in clause_set   */
7      pure_sig := getPureNames(r_sig, c_sig, clause_set)
       /* apply Purify to clause_set to eliminate names in pure_sig        */
8      clause_set := purify(pure_sig, clause_set)
       /* simplify all axioms in clause_set                                */
9      clause_set.getSimplified()
10     initialising Set⟨Clause⟩ sub_clause_set to null
11     foreach RoleName role in r_sig do
           /* get from clause_set all axioms that contain role             */
12         sub_clause_set := getSubset(role, clause_set)
           /* remove from clause_set all axioms in sub_clause_set          */
13         clause_set.removeAll(sub_clause_set);
           /* attempt to transform sub_clause_set into r-reduced form      */
14         sub_clause_set.getRReducedForm(role, sub_clause_set);
           /* check whether sub_clause_set is in r-reduced form            */
15         if isRReducedForm(role, role_clause_set) then
               /* apply Ackermann^R to sub_clause_set to eliminate role    */
16             sub_clause_set := ackermann(role, sub_clause_set)
               /* simplify all axioms in sub_clause_set                    */
17             sub_clause_set.getSimplified()
               /* add the resulting set sub_clause_set back to clause_set  */
18             clause_set.addAll(sub_clause_set)
               /* remove role from r_sig                                   */
19             r_sig.remove(role)
               /* add all introduced definer names to c_sig                */
20             c_sig.addAll(sub_clause_set.getDefiners())
21         else
               /* add the unchanged set sub_clause_set back to clause_set  */
22             clause_set.addAll(sub_clause_set)
23     end
24     Similar for loop over the concept names in the present c_sig
25     initialising final int sig_size_after to (r_sig.size() + c_sig.size())
26 while sig_size_before != sig_size_after
   /* clause_set still contains names in r_sig or c_sig                    */
27 return clause_set
```

FAME defaults to performing role forgetting first. This is because during the role forgetting process concept definer names may be introduced (to facilitate the normalisation of the input ontology). These definer names, regarded as regular concept names, can thus be eliminated as part of subsequent concept forgetting.

Given an ontology $\mathcal{O}$ and a forgetting signature $\mathcal{F} = \{r_1, ..., r_m, A_1, ..., A_n\}$, where $r_i \in \mathsf{sig_R}(\mathcal{O})$ $(1 \leq i \leq m)$ and $A_j \in \mathsf{sig_C}(\mathcal{O})$ $(1 \leq j \leq n)$, the forgetting process in FAME consists of four phases: (i) the conversion of $\mathcal{O}$ into a set $\mathcal{N}$ of clauses (clausification), (ii) the role forgetting phase, (iii) the concept forgetting phase, and (iv) the conversion of the resulting set $\mathcal{N}'$ of clauses into an ontology $\mathcal{O}'$ (declausification). The concept (role) forgetting phase is an iteration of several rounds in which the concept (role) names in $\mathcal{F}$ are eliminated using the corresponding calculus described in the previous section and the work of [25,26].

The main algorithm used by FAME is shown in Algorithm 1. FAME performs purification prior to other steps (Lines 6–9). This is because the purify rules do not require the clause set to be normalised or to be transformed into a reduced form, and they can be applied at any time. Moreover, applying the purify rules to a clause set often results in numerous syntactic redundancies, tautologies and contradictions inside the clauses which are immediately eliminated or simplified, leading to a much reduced set with fewer clauses and fewer names. This makes subsequent forgetting less difficult. The $\mathsf{getSubset}(\mathcal{S}, \mathcal{O})$ method extracts from the clause set $\mathcal{O}$ all axioms that contain the name $\mathcal{S}$. $\mathcal{S}$ can thus be eliminated from this subset, rather than from the entire set $\mathcal{O}$. Subsequent simplifications are then performed on the resulting subset (i.e., the elimination and the simplification are performed *locally*). This significantly reduces the search space and has improved the efficiency of FAME (compared to the early prototypes). It is found that a name that could not be eliminated by FAME might become eliminable after the elimination of another name [24]. We therefore impose a do-while loop on the iterations of the elimination rounds. The breaking condition checks if there were names eliminated during the previous elimination rounds. If so, FAME repeats the iterations, attempting to eliminate the remaining names. The loop terminates when the forgetting signature becomes empty or no names were eliminated during the previous elimination rounds.

5 The Evaluation

The current versions of FAME 1.0 and 2.0 were evaluated on a large corpus of ontologies taken from the NCBO BioPortal repository, a resource that currently includes more than 600 ontologies originally developed for clinical research. The corpus for evaluation was based on a snapshot of the repository taken in March 2017 [17], containing 396 OWL API compatible ontologies.

The expressivity of the ontologies in the snapshot ranges from $\mathcal{EL}$ and $\mathcal{ALC}$ to $\mathcal{SHOIN}$ and $\mathcal{SROIQ}$. Since FAME 1.0 (2.0) can handle ontologies as expressive as $\mathcal{ALCOIH}(\triangledown, \sqcap)$ ($\mathcal{ALCOQH}(\triangledown, \neg, \sqcap, \sqcup)$), we adjusted these ontologies to the language of $\mathcal{ALCOIH}(\triangledown, \sqcap)$ ($\mathcal{ALCOQH}(\triangledown, \neg, \sqcap, \sqcup)$). This involved simple reformulations as summarised in Table 1, which also lists the types of axioms that

	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

	Maximum	Minimum	Mean	Median	90th percentile
$\#(\mathcal{O})$	1833761	100	4651	1096	12570
$\#\mathsf{sig}_{\mathsf{C}}(\mathcal{O})$	847760	36	2110	502	5598
$\#\mathsf{sig}_{\mathsf{R}}(\mathcal{O})$	1390	0	54	12	144
$\#\mathsf{sig}_{\mathsf{I}}(\mathcal{O})$	87879	0	216	0	206

Table 2: Statistics of the ontologies used for evaluation

can be handled by FAME 1.0 (2.0). Concepts not expressible in $\mathcal{ALCOIH}(\triangledown, \sqcap)$ ($\mathcal{ALCOQH}(\triangledown, \neg, \sqcap, \sqcup)$) were replaced by $\top$. Table 2 shows statistical information about the adjusted ontologies, where $\#(\mathcal{O})$ denotes the number of axioms in the ontologies, and $\#\mathsf{sig}_{\mathsf{C}}(\mathcal{O})$, $\#\mathsf{sig}_{\mathsf{R}}(\mathcal{O})$ and $\#\mathsf{sig}_{\mathsf{I}}(\mathcal{O})$ denote respectively the numbers of concept names, role names and individual names in the ontologies.

To reflect real-world application scenarios, we evaluated the performance of FAME 1.0 and 2.0 for eliminating different numbers of concept and role names from each ontology. In particular, we considered the cases of eliminating 10%, 40% and 70% of concept and role names in the signature of each ontology. The names to be forgotten were randomly chosen. The experiments were run on a desktop computer 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 used on each run.

The results obtained from eliminating different numbers of concept and role names from the $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontologies (computed by FAME 1.0) and from the $\mathcal{ALCOQH}(\triangledown, \neg, \sqcap, \sqcup)$-ontologies (computed by FAME 2.0) are shown respectively in Table 3 and Table 4. The most encouraging results are that: on average, (i) FAME 1.0 (2.0) was successful in more than 90% of the test cases, i.e., eliminated all specified concept and role names and the possibly introduced definers, and (ii) in most of the $\mathcal{ALCOIH}(\triangledown, \sqcap)$-cases forgetting solutions were computed within one second, and in most of the $\mathcal{ALCOQH}(\triangledown, \neg, \sqcap, \sqcup)$-cases forgetting so-

Settings	Results				
$\#\mathcal{F}_C$ (avg)	Time (sec)	Timeouts	Success Rate	Nominal	Clause Growth
211 (10%)	0.307	1.8%	94.9%	7.6%	-10.3%
844 (40%)	0.895	3.4%	93.4%	17.4%	-41.2%
1477 (70%)	1.364	6.6%	90.2%	24.7%	-72.4%
$\#\mathcal{F}_R$ (avg)	Time (sec)	Timeouts	Success Rate	Definer	Clause Growth
5 (10%)	0.309	0.0%	100.0%	0.0%	0.9%
22 (40%)	0.977	2.5%	97.5%	0.0%	3.5%
38 (70%)	1.891	6.6%	93.4%	0.0%	6.7%

Table 3: Evaluation results of FAME 1.0

Settings	Results				
$\#\mathcal{F}_C$ (avg)	Time (sec)	Timeouts	Success Rate	Definer	Semantic
214 (10%)	1.265	1.0%	96.5%	2.5%	25.0%
856 (40%)	3.901	3.8%	90.4%	5.8%	8.8%
1498 (70%)	7.272	7.6%	82.3%	10.1%	5.3%
$\#\mathcal{F}_R$ (avg)	Time (sec)	Timeouts	Success Rate	Definer	Clause Growth
6 (10%)	0.778	0.0%	100.0%	0.0%	3.5%
26 (40%)	2.654	3.5%	96.5%	0.0%	14.8%
45 (70%)	4.906	9.1%	90.9%	0.0%	23.7%

Table 4: Evaluation results of FAME 2.0

lutions were computed within four seconds (the current versions include several significant improvements over the prototypes used in [23,24,25,26]).

Because of one rewrite rule in the calculus for concept name elimination in $\mathcal{ALCOIH}(\triangledown, \sqcap)$ [24], fresh nominals might be introduced during the forgetting process, but these nominals cannot be eliminated from the forgetting solutions using the methods of FAME 1.0. The column headed Nominal in Table 3 suggests however that forgetting solutions containing fresh nominals only occurred in a small number of cases ($\leq 25\%$). The column headed Definer shows the percentages of cases where the introduced definers could not be all eliminated from the forgetting solutions. Observe that, in concept forgetting, there was a decrease in the number of clauses in the forgetting solutions (compared to the input clause set), but in role forgetting, there was on the contrary an increase; see the Clause Growth column. As elaborated previously, the uniform interpolant computed by FAME 2.0 in concept forgetting could also be a semantic solution. The column headed Semantic shows that on average, in 13.2% of the test cases, the uniform interpolant was also a semantic solution. This means that semantic solutions of concept forgetting are rare for $\mathcal{ALCOQH}(\triangledown, \neg, \sqcap, \sqcup)$.

The most closely related tools to the FAME family are LETHE [10,11] and the tool developed by [13]. Both use resolution-based methods for computing uniform interpolants for $\mathcal{ALC}$ TBoxes, and in the case of LETHE several extensions of $\mathcal{ALC}$ TBoxes. An empirical comparison of FAME 1.0 and 2.0 with LETHE has shown that FAME 1.0 and 2.0 are considerably faster than LETHE [2].

References

1. W. Ackermann. Untersuchungen über das Eliminationsproblem der mathematischen Logik. *Mathematische Annalen*, 110(1):390–413, 1935.
2. R. Alassaf and R. A. Schmidt. A Preliminary Comparison of the Forgetting Solutions Computed using SCAN, LETHE and FAME. In *Proc. SOQE'17*, volume 2013 of *CEUR Workshop Proceedings*, pages 21–26. CEUR-WS.org, 2017.
3. E. Botoeva, B. Konev, C. Lutz, V. Ryzhikov, F. Wolter, and M. Zakharyaschev. Inseparability and Conservative Extensions of Description Logic Ontologies: A Survey. In *Proc. RW'16*, volume 9885 of *LNCS*, pages 27–89. Springer, 2016.
4. W. Del-Pinto and R. A. Schmidt. Forgetting-Based Abduction in $\mathcal{ALC}$. In *Proc. SOQE'17*, volume 2013 of *CEUR Workshop Proceedings*, pages 27–35. CEUR-WS.org, 2017.
5. D. M. Gabbay, R. A. Schmidt, and A. Szałas. *Second Order Quantifier Elimination: Foundations, Computational Aspects and Applications*. College Publications, 2008.
6. S. Ghilardi, C. Lutz, and F. Wolter. Did I Damage My Ontology? A Case for Conservative Extensions in Description Logics. In *Proc. KR'06*, pages 187–197. AAAI Press, 2006.
7. 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.
8. B. Konev, C. Lutz, D. Walther, and F. Wolter. Model-theoretic inseparability and modularity of description logic ontologies. *Artif. Intell.*, 203:66–103, 2013.
9. 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.
10. P. Koopmann. *Practical Uniform Interpolation for Expressive Description Logics*. PhD thesis, University of Manchester, UK, 2015.
11. 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.
12. F. Lin and R. Reiter. Forget it! In *Proc. AAAI Fall Symposium on Relevance*, pages 154–159. AAAI Press, 1994.
13. M. Ludwig and B. Konev. Practical Uniform Interpolation and Forgetting for $\mathcal{ALC}$ TBoxes with Applications to Logical Difference. In *Proc. KR'14*. AAAI Press, 2014.
14. C. Lutz, D. Walther, and F. Wolter. Conservative Extensions in Expressive Description Logics. In *Proc. IJCAI'07*, pages 453–458. AAAI/IJCAI Press, 2007.
15. C. Lutz and F. Wolter. Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. *J. Symb. Comput.*, 45(2):194–228, 2010.
16. C. Lutz and F. Wolter. Foundations for Uniform Interpolation and Forgetting in Expressive Description Logics. In *Proc. IJCAI'11*, pages 989–995. IJCAI/AAAI Press, 2011.
17. N. Matentzoglu and B. Parsia. BioPortal Snapshot 30.03.2017, Mar. 2017.
18. N. Nikitina and S. Rudolph. (Non-)Succinctness of uniform interpolants of general terminologies in the description logic $\mathcal{EL}$. *Artif. Intell.*, 215:120–140, 2014.
19. A. Visser. *Bisimulations, Model Descriptions and Propositional Quantifiers*. Logic Group Preprint Series. Department of Philosophy, Utrecht University, 1996.
20. 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.
21. Y. Zhang and Y. Zhou. Forgetting Revisited. In *Proc. KR'10*. AAAI Press, 2010.

22. Y. Zhao. *Automated Semantic Forgetting for Expressive Description Logics*. PhD thesis, The University of Manchester, UK, 2017.
23. Y. Zhao and R. A. Schmidt. Concept Forgetting in $\mathcal{ALCOI}$-Ontologies Using An Ackermann Approach. In *Proc. ISWC'15*, volume 9366 of *LNCS*, pages 587–602. Springer, 2015.
24. 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.
25. 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.
26. 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.