KI - Künstliche Intelligenz
https://doi.org/10.1007/s13218-020-00655-w

SYSTEMS DESCRIPTION

LETHE: Forgetting and Uniform Interpolation for Expressive Description Logics

Patrick Koopmann[1]

Received: 30 December 2019 / Accepted: 25 March 2020

Abstract
Uniform interpolation and forgetting describe the task of projecting a given ontology into a user-specified vocabulary, that is, of computing a new ontology that only uses names from a specified set of names, while preserving all logical entailments that can be expressed with those names. This is useful for ontology analysis, ontology reuse and privacy. LETHE is a tool for performing uniform interpolation on ontologies in expressive description logics, and it can be used from the command line, using a graphical interface, and as a Java library. It furthermore implements methods for computing logical difference and performing abduction using uniform interpolation. We present the tool together with an evaluation on a varied corpus of realistic ontologies.

Keywords Description logics · Non-classical reasoning · Uniform interpolation · Forgetting

1 Introduction

Description logic (DL) ontologies are used in a range of application areas as a means to define terminological domain knowledge via concept and role names. Applications in medicine, biology and the semantic web often lead to the development of large and complex ontologies that cover wide areas of knowledge. Understanding and maintaining such complex ontologies becomes difficult without appropriate tool support. On the other hand, some information from existing ontologies might be useful for reuse in new ontologies, while one does not want to import the complexity of the whole ontology. Uniform interpolation, also studied under the name of forgetting, has the potential to approach these challenges [4, 14]. Given an ontology $\mathcal{O}$ and a signature Σ of concept and role names, a uniform interpolant for $\mathcal{O}$ over Σ is a new ontology that covers all logical entailments in Σ, while using no names that are outside of the signature Σ (see Fig. 1 for an example). Uniform interpolation can be used for ontology reuse by computing a specialised ontology that only deals with the names that are relevant for the new application. Furthermore, it can be used to make implicit, hidden relations between names visible, which can be helpful for ontology understanding and maintenance. In addition, uniform interpolation can be used to solve other non-classical reasoning problems relevant in the context of ontology maintenance, such as logical difference [19] and abduction [3, 15].

LETHE is a tool that can be used to compute uniform interpolants in different expressive DLs.[1] Internally, it uses a resolution method presented in [9] for $\mathcal{ALCH}$ TBoxes, and later extended to $\mathcal{SHQ}$ [10] and knowledge bases consisting of both a TBox and an ABox [11]. Since those publications, a few bugfixes, optimisations and new features have been implemented. This paper presents the current version of LETHE: the reasoning services it supports out of the box, the different user-interfaces, and an evaluation comparing LETHE with FAME [20], the other state-of-the-art uniform interpolation tool for expressive DLs.

2 Preliminaries

We first give an overview about the DLs relevant for LETHE, and then discuss the supported reasoning services.

Funded by the DFG grant 389793660 as part of TRR 248 (see https://www.perspicuous-computing.science/)

✉ Patrick Koopmann
patrick.koopmann@tu-dresden.de

[1] Institute of Theoretical Computer Science, Technische Universität Dresden, 01187 Dresden, Germany

[1] https://lat.inf.tu-dresden.de/~koopmann/LETHE.

Published online: 13 April 2020

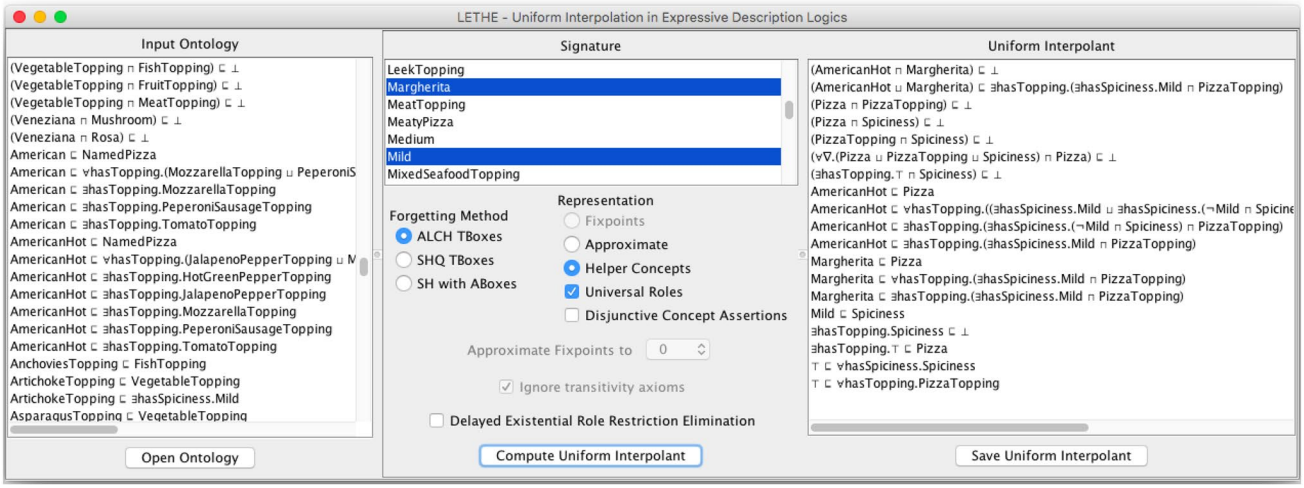

Fig. 1 Graphical user interface of LETHE, here showing a uniform interpolant of the pizza ontology for {AmericanHot, Margherita, Mild, Pizza, PizzaTopping, Spiciness, hasTopping, hasSpiciness}

2.1 Description Logics

In the DLs we consider, *concepts* are constructed from the pair-wise disjoint sets $\mathsf{N_C}$, $\mathsf{N_R}$ and $\mathsf{N_I}$ of respectively *concept*, *role-* and *individual names* according to the following syntax rule:

$$C ::= \top \mid \bot \mid A \mid \neg C \mid C \sqcup C \mid C \sqcap D \mid \exists r.C \mid \forall r.C \mid \geq nr.C \mid \{a\},$$

where $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R} \cup \{\nabla\}$ and $n \in \mathbb{N}, n \geq 1$. ∇ denotes the *universal role*. A *knowledge base* (KB) is a finite set of *concept inclusions* (CIs) of the form $C \sqsubseteq D$, *role inclusions* (RIs) of the form $r \sqsubseteq s$, and *assertions* of the forms $C(a)$, $r(a, b)$ where C, D are concepts and $r, s \in \mathsf{N_R}$ and $a, b \in \mathsf{N_I}$. CIs, RIs and assertions are collectively called *axioms*. A KB without assertions is called *ontology* or *TBox*.

The basic DL $\mathcal{ALC}$ just supports the constructs $\top$, $\bot$, A, $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists r.C$ and $\forall r.C$, no universal roles, and only axioms of the form $C \sqsubseteq D$. $\mathcal{S}$ additionally allows for axiom $\mathsf{trans}(r)$. $\mathcal{EL}$ restricts $\mathcal{ALC}$ by only allowing $\top$, A, $C \sqcap D$ and $\exists r.C$ If $\mathcal{L}$ is a DL, $\mathcal{LH}$ denotes its extension with *role axioms* $r \sqsubseteq s$, $\mathcal{LQ}$ its extension with *number restrictions* $\geq nr.C$, $\mathcal{LO}$ its extension with nominals $\{a\}$, and $\mathcal{LU}$ its extension with universal roles. For instance, $\mathcal{ALCH}$ extends $\mathcal{ALC}$ with axioms of the form $r \sqsubseteq s$, and $\mathcal{SHQ}$ supports axioms of the forms $r \sqsubseteq s$, $\mathsf{trans}(r)$, and concepts of the form $\geq nr.C$.

The semantics of DLs is defined in terms of *interpretations* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, with the non-empty set $\Delta^{\mathcal{I}}$ as domain, and the *interpretation function* $\cdot^{\mathcal{I}}$ mapping each $a \in \mathsf{N_I}$ to $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, each $A \in \mathsf{N_C}$ to $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each $r \in \mathsf{N_R}$ to $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, $\nabla^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and which is extended to concepts by

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}}, \\
\bot^{\mathcal{I}} &= \emptyset, \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}, \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}}, \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}}, \\
(\exists r.C)^{\mathcal{I}} &= \{x \mid \text{there exists} \langle x, y \rangle \in r^{\mathcal{I}} \text{ s.t. } y \in C^{\mathcal{I}}\}, \\
(\forall r.C)^{\mathcal{I}} &= \{x \mid \text{for all} \langle x, y \rangle \in r^{\mathcal{I}}, y \in C^{\mathcal{I}}\} \\
(\geq nr.C)^{\mathcal{I}} &= \{x \mid \#\{\langle x, y \rangle \in r \mid y \in C\} \geq n\} \text{ and} \\
\{a\}^{\mathcal{I}} &= a^{\mathcal{I}}.
\end{aligned}$$

An axiom α is *satisfied in an interpretation* $\mathcal{I}$, in symbols $\mathcal{I} \vDash \alpha$, if $\alpha = C \sqsubseteq D$ and $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, $\alpha = r \sqsubseteq s$ and $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$, $\alpha = \mathsf{trans}(r)$ and $r^{\mathcal{I}}$ is transitive, $\alpha = A(a)$ and $a^{\mathcal{I}} \in A^{\mathcal{I}}$, or $\alpha = r(a, b)$ and $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$. If for a KB $\mathcal{K}$, $\mathcal{I} \vDash \alpha$ for all axioms $\alpha \in \mathcal{K}$, then $\mathcal{I}$ is a model of $\mathcal{K}$. An axiom α is *entailed* by a KB $\mathcal{K}$, in symbols $\mathcal{K} \vDash \alpha$, if $\mathcal{I} \vDash \alpha$ for all models $\mathcal{I}$ of $\mathcal{K}$.

In addition to these classical concept constructors, a less common concept constructor we use is the *greatest fixpoint* $\nu X.C[X]$ [2], which corresponds to the limit of the sequence $\top, C[\top], C[C[\top]], \cdots$, Given a DL $\mathcal{L}$, we denote by $\mathcal{L}\nu$ the extension with greatest fixpoint operators. For formal details on the semantics of fixpoint operators, we refer to [2].

2.2 Uniform Interpolation and Related Tasks

Definition 1 (*Uniform interpolation*) Let $\mathcal{K}$ be a KB, $\mathcal{L}$ a DL, and Σ be a signature. Then, we call a KB $\mathcal{K}^{\mathcal{L},\Sigma}$ a *uniform* $\langle \mathcal{L}, \Sigma \rangle$*-interpolant* of $\mathcal{K}$ iff

1. $\mathsf{sig}(\mathcal{K}^{\mathcal{L},\Sigma}) \subseteq \Sigma$, and
2. for every $\mathcal{L}$ axiom α with $\mathsf{sig}(\alpha) \subseteq \Sigma$, we have $\mathcal{K} \models \alpha$ iff $\mathcal{K}^{\mathcal{L},\Sigma} \models \alpha$.

Note that we do not require $\mathcal{K}^{\mathcal{L},\Sigma}$ to be a $\mathcal{K}$ ontology itself. If the DL does not allow for fixpoints, already for acyclic ontologies, there can be signatures, for which no uniform interpolant exists in that DL [14]. On the other hand, for the DLs considered here, uniform interpolants of ontologies always exist in DLs with fixpoints. Furthermore, when interpolating KBs with assertions, a uniform interpolant may only exist if we allow for nominals in the result [8, 11]. We often speak of *the uniform interpolant*, referring the the logically strongest among the possible options. The dual notion of uniform interpolation is forgetting: The result of forgetting a name x from an ontology $\mathcal{O}$ is the uniform $\langle \mathcal{L}, \Sigma\rangle$-interpolant for $\Sigma = \mathsf{sig}(\mathcal{O})\backslash\{x\}$.

Fixpoint operators are not supported by the web ontology standard OWL, which is why Lethe offers two ways to eliminating them when producing the result: by either approximation, or by using auxiliary concept names, so-called definers, that simulate the behaviour of greatest fixpoints and make sure all logical entailments of the uniform interpolant are kept. (For details, see [9].)

If we want to reuse a uniform interpolant in a different context, it may be useful to compute it in a DL with universal roles. The following theorem is an easy consequence of [6, Theorem 3].

Theorem 1 *Let $\mathcal{L} \in \{\mathcal{ALC}, \mathcal{ALCQ}\}$, $\mathcal{O}$ an $\mathcal{L}$-ontology, Σ a signature, and $\mathcal{O}^{\Sigma}$ a uniform $\langle \mathcal{LU}, \Sigma\rangle$ -interpolant of $\mathcal{O}$. Then, for $\mathcal{L}$ -every ontology $\mathcal{O}_2$ with $(\mathsf{sig}(\mathcal{O}_2) \cap \mathsf{sig}(\mathcal{O}_1)) \subseteq \Sigma$ and every $\mathcal{L}$-axiom α s.t. $\mathsf{sig}(\alpha) \subseteq \Sigma$,we have $(\mathcal{O} \cup \mathcal{O}_2) \models \alpha$ iff $(\mathcal{O}^{\Sigma} \cup \mathcal{O}_2) \models \alpha$.*

Intuitively, if we want to reuse an $\mathcal{ALC}$- or an $\mathcal{ALCQ}$-ontology $\mathcal{O}$ in another context that speaks about Σ, we can replace $\mathcal{O}$ by its uniform $\langle \mathcal{LU}, \Sigma\rangle$-interpolant and still preserve all consequences over Σ. A corresponding property does not hold for uniform $\langle \mathcal{L}, \Sigma\rangle$-interpolants in general.

In addition to uniform interpolation, Lethe implements logical difference and abduction by reduction to uniform interpolation with some dedicated optimisations.

Definition 2 (*Logical difference*) Let $\mathcal{O}_1, \mathcal{O}_2$ be two ontologies, Σ a signature and $\mathcal{L}$ a DL. The *logical difference of $\mathcal{O}_1$ to $\mathcal{O}_2$ over* $\langle \Sigma, \mathcal{L}\rangle$ is the set of all $\mathcal{L}$-axioms α with $\mathsf{sig}(\alpha) \subseteq \Sigma$ s.t. $\mathcal{O}_1 \models \alpha$ and $\mathcal{O}_2 \not\models \alpha$. If $\Sigma = (\mathsf{sig}(\mathcal{O}_1) \cap \mathsf{sig}(\mathcal{O}_2))$, it is called *logical difference of $\mathcal{O}_1$ to $\mathcal{O}_2$ over $\mathcal{L}$*. A *representation* of the logical difference is an ontology $\mathsf{logDiff}(\mathcal{O}_1, \mathcal{O}_2, \Sigma)$ s.t. for every axiom α in the logical difference, $\mathsf{logDiff}(\mathcal{O}_1, \mathcal{O}_2, \Sigma) \cup \mathcal{O}_2 \models \alpha$.

Lethe uses computes representations of logical difference by checking for entailments of axioms in the uniform interpolant. Additional optimisations are used to restrict the number of reasoner calls and forgetting steps performed for comparing large ontologies with large syntactical overlap. Uniform interpolation is also used for computing representatives of logical differences in [12, 19].

Definition 3 (*Abduction*) Let $\mathcal{O}_1$ be an ontology, α an axiom s.t. $\mathcal{O} \not\models \alpha$ (the *observation*), and Σ a signature (the set of *abducibles*). Then, a *hypothesis for $\mathcal{O} \models \alpha$ in Σ* is an ontology $\mathcal{H}$ s.t.

1. $\mathcal{O} \cup \mathcal{H} \models \alpha$,
2. $\mathsf{sig}(\mathcal{H}) \subseteq \Sigma$, and
3. for every ontology $\mathcal{H}'$ satisfying the above conditions, $\mathcal{O} \cup \mathcal{H}' \models \mathcal{H}$.

Lethe solves this abduction problem for $\mathcal{ALCH}$ ontologies and signatures Σ s.t. $(\mathsf{sig}(\mathcal{O}) \cap \mathsf{N_R}) \subseteq \Sigma$. For an extension of our abduction setting (also using Lethe), see [3].

3 User Interfaces of Lethe

Lethe implements three different algorithms for computing uniform interpolants, one for $\mathcal{ALCH}$ ontologies based on [9] (ALCHForgetter), one for $\mathcal{SHQ}$ ontologies based on [10] (SHQForgetter) and one for forgetting in $\mathcal{SH}$ knowledge bases based on [8, 11] (KBForgetter). The general approach and implementation idea of these methods is described in Sect. 4. Uniform interpolation is always performed by forgetting one name after the other. While the logic supported by KBForgetter is more general than the one by $\mathcal{ALCH}$-Forgetter, the implementations differ substantially, and thus may perform differently well on the same input.

Graphical User Interface. With the application of ontology analysis in mind, Lethe comes with a simple graphical user interface that can be used to quickly try out the tool (see Fig. 1 illustrating uniform interpolation with the *pizza ontology*[2]). Ontologies in OWL syntax can be loaded and are displayed in DL syntax. The user then selects the target signature, the method to be used, and whether greatest fixpoint operators should approximated or simulated with helper concepts. During computation, the user is presented with a progressbar where he sees the current name being forgotten. The first 80–90% of names are usually forgotten very fast, while the more difficult names are forgotten in

[2] https://github.com/owlcs/pizza-ontology/.

the end. If the user does not want to wait, he can cancel the forgetting process, in which case he sees the currently computed uniform interpolant.

Console Interface. LETHE furthermore allows to be used as a command line tool. Here, the user can also set a time out, after which the partial uniform interpolant is saved if the computation did not terminate yet. A second command is provided for computing logical differences.

Use as Java Library. Probably the most relevant for practical applications is the possibility to use LETHE as a Java library. Though implemented in Scala, LETHE provides for a facade supporting standard Java data structures that is compatible with the OWL API 5.1.7 [5]. The use of this facade is documented on the website. Classes and interfaces are provided for the three different forgetting methods, and for performing uniform interpolation, logical difference computation and abduction with abducibles.

4 Method and Implementation

4.1 General Method

In order to forget a specific name, LETHE performs the following steps:

1. normalise the input,
2. compute all inferences on the name to forget,
3. filter out occurrences of the name, and
4. denormalise.

For illustration, we describe the method for $\mathcal{ALC}$ TBoxes—the underlying idea is the same for the more expressive DLs (for details, see [8–11]). In our normal form, every axiom is of the form

$$\top \sqsubseteq L_1 \sqcup \ldots \sqcup L_n \qquad L_i ::= A \mid \neg A \mid \exists r.D \mid \forall r.D,$$

where $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, and D is taken from a special set $\mathsf{N_D} \subseteq \mathsf{N_C}$ of *definers*. We call the L_i *literals* and normalised axioms *clauses*, usually omit the leading $\top \sqsubseteq$ and treat them as sets, that is, no literal occurs twice and the order is not important. We further have the restriction that no clause contains more than one literal of the form $\neg D$, where $D \in \mathsf{N_D}$.

In Step 2, we make use of the calculus shown in Fig. 2. In ***r*-Prop**, the definer D_{12} is a definer representing $D_1 \sqcap D_2$, which we introduce if not existent yet. In ***r*-Res**, $\mathcal{O}$ refers to the current set of clauses. Here, LETHE uses HermiT to decide the entailment. The rules ***A*-Res** and ***r*-Res** are used to perform the inferences on the symbol to forget (concept name A or role name r). Since a clause may contain at most

$$A\text{-}\mathbf{Res}: \frac{C_1 \sqcup A \quad C_2 \sqcup \neg A}{C_1 \sqcup C_2} \qquad r\text{-}\mathbf{Prop}: \frac{C_1 \sqcup \forall r.D_1 \quad C_2 \sqcup \mathsf{Q}r.D_2}{C_1 \sqcup C_2 \sqcup \mathsf{Q}r.D_{12}}$$

$$r\text{-}\mathbf{Res}: \frac{C_1 \sqcup \exists r.D_1 \quad C_2 \sqcup \forall r.D_2 \quad \ldots \quad C_n \sqcup \forall r.D_n}{C_1 \sqcup \ldots \sqcup C_n}, \quad \text{where } \mathcal{O} \models D_1 \sqcap \ldots D_n \sqsubseteq \bot.$$

Fig. 2 Forgetting calculus for $\mathcal{ALC}$

one negative definer literal, the rules are not applicable if the premises contain different negative definer literals, for instance $\neg D_1$ and $\neg D_2$. The rule ***r*-Prop** is a so-called *combination rule* and may *combine* the different definers D_1 and D_2 into a new definer D_{12}, resulting in new clauses which contain $\neg D_{12}$ instead of $\neg D_1$ and $\neg D_2$, which makes new inferences with ***A*-Res** and ***r*-Res** possible (recall that clauses may contain at most one negative definer literal). The calculi for more expressive DLs have additional combination rules that reflect the additional expressivity. Our method makes sure that in the worst case, at most exponentially many new definers are introduced, which ensures termination of the forgetting procedure.

In the denormalisation step, the definers are eliminated again using standard rewriting rules. It is in this step that we may introduce fixpoint operators into the ontology. Alternatively, if fixpoints are not desired in the output, we keep the definers for which the corresponding fixpoint cannot be simplified away (see below), or we approximate the fixpoint expression by unfolding the fixpoint expression up to a certain depth.

4.2 Implementation of Forgetting Calculus

The implemented forgetting methods use different strategies of determining when a combination rule has to be applied: ALCHForgetter keeps a map for each definer that stores its "distance" to the name to be forgotten. A combination rule then only combines definers that have the same distance. SHQForgetter and KBForgetter instead use a "lazy" approach: they first apply resolution unrestricted, allowing more than one negative definer. If a clause with negative definers $\neg D_1, \ldots, \neg D_n$ is inferred, clauses containing $D_1, \ldots, D_n$ under a role restriction are determined, and combination rules are tried to introduce a definer representing $D_1 \sqcap \ldots \sqcap D_n$. In the first approach, we try to predict when definer combination is necessary. In the second approach, we apply combination on demand. In addition, we use a set of usual techniques from resolution methods, such as indexing, forward- and backward subsumption.

4.3 Implementation of Uniform Interpolation

To compute the uniform interpolant, we apply Steps 1–4 for each name in the ontology that is not in the desired signature.

These steps are only applied to the axioms that contain the name to be forgotten, which are then replaced by the forgetting result. It turns out that the order in which we forget is very crucial to the performance: our heuristics take into account the positive and negative occurrences of the name to be forgotten and we generally start with the least frequent ones.

In addition, we use pre- and post-processing to reduce the number of axioms to be processed, and to improve the shape of the computed uniform interpolant. First, we use *module extraction* as in [18], and as implemented in the OWL API, to compute a subset of the ontology that contains all relevant axioms for the uniform interpolant. Second, we use purification to quickly forget all names which occur either only positively or only negatively, in which case they can be replaced by respectively $\top$ and $\bot$. As post-processing, we use a set of *beautification rules* that improve the syntactic shape of the axioms, by detecting tautological or contradictory subexpressions (including fixpoints), detecting redundancies or applying associativity. A cheaper version of beautification is used during the forgetting phase to keep the size of the current uniform interpolant small. A more expensive form is applied at the end to make the final uniform interpolant more human-readable and to keep the expressivity of the used DL small. For instance, an $\mathcal{EL}$ ontology might be preferable over an equivalent $\mathcal{ALC}$ ontology, if this transformation is possible in simple steps.

5 Evaluation

We evaluate Lethe and compare it with Fame, the other state-of-the-art tool for uniform interpolation in expressive DLs. While being faster, later versions of Fame compute uniform interpolants in a very expressive DL not supported by OWL, which explains why we could not produce OWL files for most inputs with Fame 2.0, the latest version. For this reason, we used the $\mathcal{ALCOIQ}$-forgetter of Fame 1.0 in our experiments.[3]

Evaluations of older versions of Lethe [8–11] and comparisons with other tools [1, 20, 21] can be found in the literature. Since then, additional optimisations and features have been implemented, as well as some bugs fixed. The evaluation presented differs in three further aspects from earlier evaluations. (1) We compute uniform interpolants with universal roles, which are now directly supported by Lethe. Fame always does this, and since it makes forgetting roles much easier, this provides for a fairer comparison. Furthermore, universal roles in the uniform interpolant can be useful in practice (see Theorem 1). (2) We do not discard

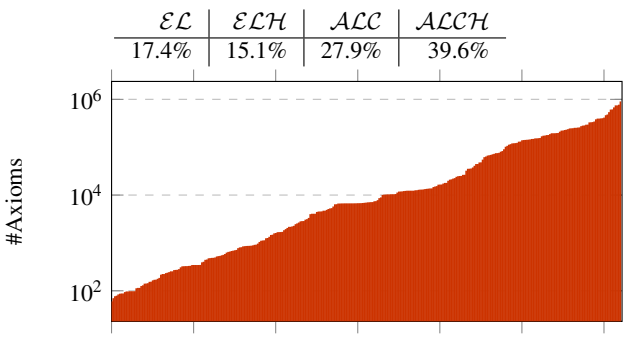

Fig. 3 Distribution of sizes and expressivity of the input ontologies

computations that caused timeouts, but instead evaluate the uniform interpolants computed within the given time frame, since in many applications, a fast computed uniform interpolant with a few more symbols is sufficient and preferable over long waiting times. (3) We use different heuristics for selecting samples of signatures.

System Specification. The experiments where run on an Intel Core i5-4590 CPU machine with 3.30 GHz and 32 GB RAM, using Debian/GNU Linux 9 and OpenJDK 11.0.5.

Corpus. We use the ontologies from the OWL Reasoner Evaluation 2015 [17], for the track *DL Classification*, which has been balanced in terms of size, expressivity and complexity of ontologies. From each ontology, we removed axioms outside of $\mathcal{ALCH}$, where we translated n-ary equivalence and disjointness axioms, as well as domain and range axioms, into corresponding $\mathcal{ALCH}$ concept inclusions. From the resulting corpus, we removed all ontologies that had less than 100 names and more than 100,000 axioms. Figure 3 shows sizes and expressivity of the ontologies in the resulting corpus of 198 ontologies.

Signatures. We focused on uniform interpolants for small signatures, which are particularly useful for ontology analysis, and thus selected signatures of 100 names for each computed uniform interpolant. We used different strategies to select signatures: (1) fully *random signatures* by selecting each name with equal probability, (2) *weighted signatures* by selecting each name weighted with the frequency of its occurrences in the ontology, and (3) *coherent signatures* by selecting names related to each other using genuine modules [18]. A genuine module is a module extracted for the signature of some axiom, and thus has as signature names that are related to that axiom in the ontology, and are consequently related to each other. To obtain a coherent signature, we took the union of randomly selected genuine modules until the overall signature size was above 100, and then randomly selected names from the resulting signature.

[3] http://www.cs.man.ac.uk/~schmidt/sf-fame/.

	$\mathcal{EL}$	$\mathcal{ALC}$	$\mathcal{ALCI}$
random	34.6%	58.1%	7.3%
weighted	36.1%	55.9%	7.9%
coherent	33.8%	63.0%	3.1%

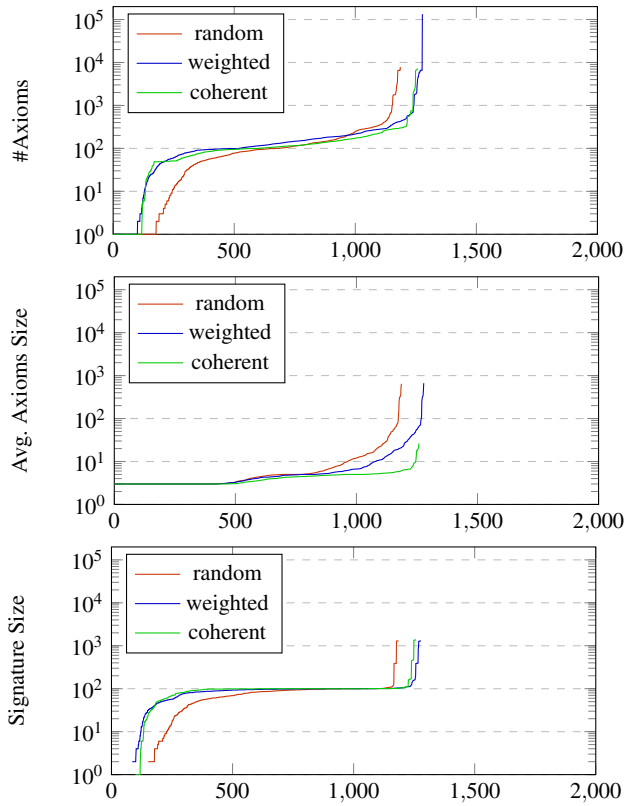

Fig. 4 Uniform interpolants computed by FAME

	$\mathcal{EL}$	$\mathcal{ELH}$	$\mathcal{ALC}$	$\mathcal{ALCH}$
random	29.3%	1.6%	67.1%	1.9%
weighted	24.1%	2.3%	71.4%	2.1%
coherent	39.7%	3.4%	56.2%	0.7%

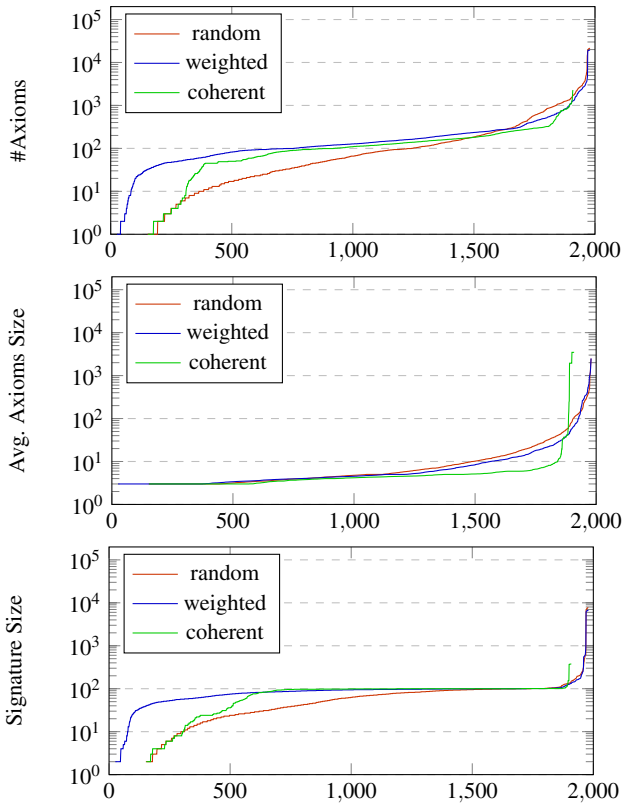

Fig. 5 Uniform interpolants computed by LETHE

The results of the evaluation are shown in Figs. 5 and 4. For the coherent signatures, LETHE produced an out of memory error each time for 7 of the ontologies. Apart from these cases, LETHE always computed a uniform interpolant, and in most cases completely for the desired signature. FAME does not have a timeout functionality as LETHE, and was terminated after the timeout passed, which is why we have less results for it. Still, one can see that LETHE was more often able to compute the uniform interpolant for the required signature size of 100. Note that FAME can also produce uniform interpolants with more than 100 symbols, due to definer symbols used to simulate fixpoints, and because the method used is *incomplete* and sometimes fails to forget some of the names.

6 Related Work

For an overview on forgetting in logics see [4]. Theoretical properties of uniform interpolation in expressive description logics have been investigated in [14]. The main competing tool for uniform interpolation in expressive DLs is FAME [20, 21], which we used in our evaluation. Which tool is recommended to use depends on the application, as FAME can be faster and supports more expressive DLs. The faster versions however often fail to compute results in OWL, as interpolants may use non-classical constructs. Another difference to LETHE is that the method underlying FAME is incomplete in the sense that it is not guaranteed to compute a uniform interpolant for every given signature. A very recent tool that can be used for forgetting in expressive DLs is DLS-FORGETTER [1], which applies the DLS algorithm on first order logic formulae. More similar to LETHE is the resolution-based method for $\mathcal{ALC}$ presented in [12], which however is not able to deal with cyclic ontologies. For the light-weight DL $\mathcal{EL}$, there exist an implemented method for acyclic terminologies [7], and one for general ontologies [13].

7 Outlook

We are currently investigating further reasoning services that are based on forgetting, and which will be implemented in future versions of Lethe. Specifically, we are looking at module extraction, abduction, and using forgetting to explain entailments in an ontology. Regarding abduction, we want to support arbitrary signatures and ABox assertions. For this generalised abduction problem, we have to adapt the forgetting procedure as well, as it has for instance to handle negated role assertions. Furthermore, we noticed that uniform interpolants are often smaller than modules extracted with the OWL API, but in some cases also larger and with more complex axioms. Another line of research to pursue is to develop a method that sits in between uniform interpolation and module extraction, and is optimised to compute small and simple ontologies that captures the entailments of a given signature, similar to [16].

Acknowledgements Open Access funding provided by Projekt DEAL.

References

1. Alassaf R, Schmidt RA (2019) Dls-forgetter: an implementation of the DLS forgetting calculus for first-order logic. In: Proceedings of GCAI 2019, EPiC series in computing, vol 65. EasyChair, pp 127–138
2. Calvanese D, De Giacomo G, Lenzerini M (1999) Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In: Proceedings of IJCAI 1999. Morgan Kaufmann, pp 84–89
3. Del-Pinto W, Schmidt RA (2019) ABox abduction via forgetting in $\mathcal{ALC}$. In: Proceedings of AAAI 2019. AAAI Press, pp 2768–2775. https://doi.org/10.1609/aaai.v33i01.33012768
4. Eiter T, Kern-Isberner G (2019) A brief survey on forgetting from a knowledge representation and reasoning perspective. KI 33(1):9–33. https://doi.org/10.1007/s13218-018-0564-6
5. Horridge M, Bechhofer S (2011) The OWL API: a Java API for OWL ontologies. Semant Web 2(1):11–21
6. Konev B, Lutz C, Walther D, Wolter F (2009) Formal properties of modularisation. Modular ontologies: concepts, theories and techniques for knowledge modularization, LNCS, vol 5445. Springer, Berlin, pp 25–66. https://doi.org/10.1007/978-3-642-01907-4_3
7. Konev B, Walther D, Wolter F (2009) Forgetting and uniform interpolation in large-scale description logic terminologies. In: Proceedings of IJCAI 2009, pp 830–835
8. Koopmann P (2015) Practical uniform interpolation for expressive description logics. Ph.D. thesis, University of Manchester, UK
9. Koopmann P, Schmidt RA (2013) Forgetting concept and role symbols in $\mathcal{ALCH}$-ontologies. Logic for programming, artificial intelligence, and reasoning–LPAR-19, LNCS, vol 8312. Springer, Berlin, pp 552–567. https://doi.org/10.1007/978-3-642-45221-5_37
10. Koopmann P, Schmidt RA (2014) Count and forget: uniform interpolation of $\mathcal{SHQ}$-ontologies. In: Automated reasoning—7th international joint conference, IJCAR 2014, LNCS, vol 8562. Springer, pp 434–448. https://doi.org/10.1007/978-3-319-08587-6_34
11. Koopmann P, Schmidt RA (2015) Uniform interpolation and forgetting for $\mathcal{ALC}$ ontologies with ABoxes. In: Proceedings of AAAI 2015. AAAI Press, pp 175–181
12. Ludwig M, Konev B (2014) Practical uniform interpolation and forgetting for $\mathcal{ALC}$ tboxes with applications to logical difference. In: Principles of knowledge representation and reasoning: proceedings of KR 2014. AAAI Press
13. Ludwig M, Walther D (2016) Towards a practical decision procedure for uniform interpolants of $\mathcal{EL}$-TBoxes—a proof-theoretic approach. In: Proceedings of GCAI 2016, EPiC series in computing, vol 41. EasyChair, pp 147–160
14. Lutz C, Wolter F (2011) Foundations for uniform interpolation and forgetting in expressive description logics. In: Proceedings of IJCAI 2011. IJCAI/AAAI, pp 989–995. https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-170
15. Möller R, Özçep ÖL, Haarslev V, Nafissi A, Wessel M (2016) Abductive conjunctive query answering wrt ontologies. KI 30(2):177–182. https://doi.org/10.1007/s13218-015-0399-3
16. Nikitina N, Glimm B (2012) Hitting the sweetspot: economic rewriting of knowledge bases. The semantic web–ISWC 2012, LNCS, vol 7649. Springer, Berlin, pp 394–409. https://doi.org/10.1007/978-3-642-35176-1_25
17. Parsia B, Matentzoglu N, Gonçalves RS, Glimm B, Steigmiller A (2017) The OWL reasoner evaluation (ORE) 2015 competition report. J Autom Reason 59(4):455–482. https://doi.org/10.1007/s10817-017-9406-8
18. Vescovo CD, Parsia B, Sattler U, Schneider T (2011) The modular structure of an ontology: atomic decomposition. In: Proceedings of IJCAI 2011. IJCAI/AAAI, pp 2232–2237. https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-372
19. Zhao Y, Alghamdi G, Schmidt RA, Feng H, Stoilos G, Juric D, Khodadadi M (2019) Tracking logical difference in large-scale ontologies: a forgetting-based approach. In: Proceedings of AAAI 2019 AAAI Press, pp 3116–3124. https://doi.org/10.1609/aaai.v33i01.33013116
20. Zhao Y, Schmidt RA (2018) FAME: an automated tool for semantic forgetting in expressive description logics. In: Automated reasoning—9th international joint conference, IJCAR 2018, LNCS, vol 10900. Springer, pp 19–27. https://doi.org/10.1007/978-3-319-94205-6_2
21. Zhao Y, Schmidt RA (2019) FAME(Q): an automated tool for forgetting in description logics with qualified number restrictions. Automated deduction–CADE 27, LNCS, vol 11716. Springer, Berlin, pp 568–579. https://doi.org/10.1007/978-3-030-29436-6_34