What a Surprise! Computing Uniform Interpolants Can Be as Efficient as Computing Modules

Anonymous Author(s)

ABSTRACT

Uniform Interpolation (UI) refers to an advanced non-standard reasoning service designed to refine ontologies by creating signature-restricted modules. These modules, known as uniform interpolants, retain only "relevant names" while preserving their meanings in the absence of other names. UI holds significant potential across various domains where tailored ontology modules are required. However, realizing its full potential demands highly optimized techniques for generating such modules. Previous studies have identified notable challenges in generating uniform interpolants for $\mathcal{EL}$-ontologies, where their computation is substantially more complex and computationally demanding than standard subset modules. In certain instances, generating a uniform interpolant may be infeasible; when feasible, the size of a uniform interpolant can expand exponentially, potentially tripling the size of the original ontology.

Despite these obstacles, this paper introduces an advanced "forgetting" method tailored for computing uniform interpolants of $\mathcal{ELIO}$-ontologies with ABoxes. We show that with effective normalization and inference strategies, these uniform interpolants can be computed efficiently, matching the speed of standard module computation. A comprehensive evaluation using a prototype implementation of this method achieved a 100% success rate on two major benchmark datasets, Oxford-ISG and BioPortal, with results delivered within seconds. The efficiency of our approach is attributed to our novel linear strategy for introducing definers, in sharp contrast to existing strategies that lead to an exponential increase in definers and computational inefficiency. Our method is unique in its ability to create signature-restricted modules for large-scale ontologies, making it a vital addition to the community's toolkit.

CCS CONCEPTS

• **Theory of computation → Description logics**; **Automated reasoning**; • **Computing methodologies → Ontology engineering**.

KEYWORDS

Ontologies, Module Extraction, Uniform Interpolation, Forgetting

1 INTRODUCTION

In computer science and artificial intelligence, an ontology is a formal representation of a domain of interest, described using a specific vocabulary of names, known as the *signature* of the ontology [17]. This definition was originally introduced in the knowledge sharing and reuse effort, for efficient engineering of (distributed, cooperating, heterogeneous) knowledge-based systems. Coming from this, a core area of application for ontologies has been knowledge management. But rather than fully capturing knowledge about a particular domain, the idea of using ontologies in knowledge management is that one would agree on a common vocabulary and use it for the knowledge shared by formal means while making the best use of the reasoning services in the back-end. By constructing these formal representations, ontologies support the reproducible transmission of data across various information systems and ensure consistent coding of concepts, and thus have gained significant traction in several areas of science and industry, such as medicine, bioinformatics, energy, law, and software engineering, where they underpin critical knowledge-driven processes and decision-making.

However, a considerable challenge with ontologies is their tendency to become large and heterogeneous. This arises from their need to represent diverse and complex domains comprehensively. The extensive scale and detail of ontologies can create substantial obstacles to their practical reusability. These challenges stem from the complexities involved in managing and maintaining such intricate structures, which can be cumbersome and generate significant computational costs during reasoning processes. Consequently, sophisticated management and optimization are essential to ensure that ontologies remain functional and efficient, especially in applications where rapid processing and agility are crucial.

An effective strategy for managing large, heterogeneous ontologies is to extract a specific part, known as a*module*, that retains the essential functionality of the original ontology within a specific context, while notably reducing its overall size. This modular approach offers several advantages for the flexible reuse of ontologies. Firstly, it allows for more localized maintenance, enabling targeted adjustments directly within the relevant module. This targeted approach simplifies updates and modifications, streamlining the management of the ontology. Secondly, isolating specific parts of the ontology enables these modules to be repurposed or seamlessly integrated into various applications, enhancing the versatility of the ontology. Additionally, from a technical standpoint, this modular approach can lead to more efficient reasoning. In certain scenarios, only a specific module might be necessary for particular deductions, and the reasoning process can be distributed across multiple machines, each handling separate modules.

We have yet to establish a formal definition for a module of an ontology. Traditionally, a module is defined as a syntactic subset of the original ontology that retains certain properties. This definition often hinges on the specification of these properties via a designated signature of names, denoted as Σ. A key requirement is that the interpretations of the Σ-names within the module must strictly align with their interpretations within the original ontology. Specifically, for any axiom involving only the names in Σ, if it is entailed by the original ontology, it must also be entailed by the module, and vice versa. This reciprocal entailment ensures that the module remains a faithful, albeit smaller, representation of the original ontology's semantics.

Modules can therefore be used as safe replacements for the full ontology within the context of Σ. However, to satisfy the above conditions, modules often need to include names outside of Σ, as illustrated in the following example.

Example 1. *Consider an ontology $\mathcal{O}$ containing the following GCIs and let $\Sigma = \{Professor, Author\}$:*

$$\begin{array}{ll} \alpha_1 & Professor \sqsubseteq \exists writes.AcademicPaper \\ \alpha_2 & AcademicPaper \sqsubseteq Publication \\ \alpha_3 & \exists writes.AcademicPaper \sqsubseteq Researcher \\ \alpha_4 & \exists writes.Publication \sqsubseteq Author \end{array}$$

This ontology states that: α_1: Professors are required to write academic papers; α_2: Academic paper is a type of publication; α_3: Anyone who writes academic papers is considered a researcher; α_4: Anyone who writes publications is considered an author.

This ontology $\mathcal{O}$ entails that a professor is an author over Σ, which is captured by its subset Σ-module $\mathcal{M} = \{\alpha_1, \alpha_2, \alpha_4\}$. Notice that the names writes, AcademicPaper, Publication — which are outside of Σ — are included in the module to preserve this information.

This requirement can limit the reusability of ontologies in various applications. It is often impossible to strictly restrict the signature of the results, making modularization a technique unsuitable for tasks like information hiding [16] and privacy protection [15]. For example, in domains like medicine or the military, ontologies often contain sensitive data that must remain confidential during public dissemination, distribution, or sharing. This concern is also significant in commercial environments where protecting proprietary information is crucial. One effective approach to maintain confidentiality is to restrict the visibility of sensitive data within these ontologies, sharing only the part of the ontology that includes accessible names for authorized individuals. However, simply adapting a module from an ontology may not be sufficient, as it might still include names beyond the intended restricted scope, potentially compromising sensitive data.

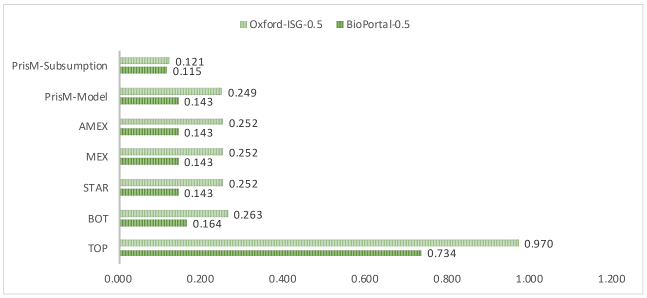


Figure 1: Ratio of Non-Σ Names in Computed Modules

To illustrate this restriction, we conducted a preliminary experiment using various modularization methods (discussed in detail later) on two widely used ontology repositories: Oxford-ISG[1] and NCBO BioPortal[2]. Specifically, our focus was on examining the inclusion of concept and role names outside the scope of Σ. In the BioPortal dataset, Σ averaged 940 names for extracting module $\mathcal{M}$, and in the Oxford-ISG dataset, it was 791. Figure 1 shows the ratio of concept and role names introduced from outside Σ to the total in $\mathcal{M}$. The results revealed that for Oxford-ISG, AMEX included 25.2% names outside Σ, compared to 14.3% in BioPortal. MEX and STAR yielded identical results to AMEX. PrisM-Subsumption resulted in 12.1% on Oxford-ISG and 11.5% on BioPortal, while PrisM-model had 24.9% and 14.3%, respectively. BOT yielded 26.3% on Oxford-ISG and 16.4% on BioPortal. TOP's results were remarkably high, at 97.0% and 73.4%, suggesting that TOP almost replicated the entire original ontology.

In situations where one needs to preserve the functionality of the source ontology but only wishes to use a specific subset of names, a signature-restricted module of the source ontology becomes highly advantageous. Uniform Interpolation (UI) [27] is one approach to creating such a module. Specifically, UI is a non-standard reasoning procedure that, given an ontology $\mathcal{O}$ and a sub-signature Σ of $\mathcal{O}$, computes a new ontology $\mathcal{V}$, called the "uniform interpolant", that employs only the names in Σ while preserving their original meanings in the absence of other names. We call UI "non-standard" because it cannot be reduced to the "standard" satisfiability or entailment tests (SAT problem). However, UI does more than merely extract a module from the original ontology. To maintain the semantic integrity of the Σ-names, new axioms must be derived from the original ontology. Consequently, uniform interpolants may contain significantly more axioms than the source ontology; think of it as a "rewritten module".

Example 2. *Reconsider the previous example. A rewritten Σ-module of $\mathcal{O}$ contains only:*

$$\alpha_5 \quad Professor \sqsubseteq Author$$

α_5 is a new logical entailment involving only the names in Σ, deduced from $\mathcal{O}$ through a number of inference steps.

Previous research has shown that UI poses greater computational challenges than modularization [3, 35, 49]. Therefore, despite UI being proven extremely useful for numerous ontology-based knowledge management tasks, including debugging and repair [41, 48], merging and alignment [29, 39, 51], versioning [19, 20, 45], semantic difference [22, 23, 31, 55], abduction and explanation generation [9, 26], and interactive ontology revision [37], its potential can only be fully realized with the development of a highly efficient algorithm (and its corresponding implementation) for computing such rewritten modules.

This paper aims to achieve this goal. In particular, we develop a practical method for computing uniform interpolants of $\mathcal{ELIO}$-ontologies with ABoxes. The method employs a highly optimized "forgetting" procedure that computes uniform interpolants by singly and systematically eliminating from the original ontology the names not included in Σ. Nikitina and Rudolph [36] have shown that computing uniform interpolants for the $\mathcal{EL}$ faimily is extremely challenging — a finite uniform interpolant does not always exist, and when it does, it can be up to triple exponential in size relative to the input ontology. In the worst case, no shorter uniform interpolant exists. However, we show in this paper that this computational challenge should not constitute a fundamental technical obstacle for UI in practice, and with effective normalization and inference strategies, uniform interpolants can be computed as efficiently as modules. A comprehensive evaluation with a prototype implementation of our method shows exceptional success rates and efficiency on the benchmark datasets NCBO BioPortal and Oxford-ISG. These results demonstrate a significant computational advantage over SOTA UI

[1]https://www.cs.ox.ac.uk/isg/ontologies/
[2]https://bioportal.bioontology.org/

tools. Additionally, we conducted an extensive comparison with various prevalent modularization methods, focusing on metrics such as result size, computation time, and memory consumption. Our findings challenge the conventional view that modularization methods are inherently superior. Instead, we show that UI can not only compete with modularization techniques but, in practice, even surpass them across these metrics. By providing a computationally feasible way to handle large ontologies, our optimized UI method enables ontology curators to foster more effective knowledge sharing across various applications and domains and thus enhances the accuracy and efficiency of knowledge management.

A long version of this paper including all missing proofs, illustrative examples, and additional empirical results, as well as the source code for the prototype implementation alongside the datasets, can be found at https://github.com/anonymous-ai-researcher/cikm2024.

2 PRELIMINARIES

Modern ontologies are often formulated in the Web Ontology Language (OWL) based on *Description Logics* (DLs) [1, 2], a successful family of knowledge representation formalisms. DLs have several variants that differ in expressivity, depending on which logical constructors are set to be used to describe domain knowledge. A basic DL, called $\mathcal{EL}$, uses concept names, role names, and the logical constructors of $\exists$ and $\sqcap$ to construct compound concepts. Our focus in this paper is $\mathcal{ELIO}$, which extends $\mathcal{EL}$ with role inverse ($\mathcal{I}$) and nominal ($\mathcal{O}$). $\mathcal{ELIO}$ is more expressive than $\mathcal{EL}$ in the sense that there exists an $\mathcal{ELIO}$-concept C such that $C \not\equiv D$ holds for all $\mathcal{EL}$-concepts D. This additional expressivity provides more power and flexibility for making statements about domain knowledge, but comes with a computational cost. One important topic within DL research is exploring the trade-off between the expressive power of the language and the computational complexity of various reasoning tasks. The expressive power of DLs is invariably constrained to ensure that standard reasoning remains decidable.

Let $\mathsf{N_C}$, $\mathsf{N_R}$ and $\mathsf{N_I}$ be mutually disjoint, countably infinite sets of *concept*, *role* and *individual* names, respectively. *Roles* in $\mathcal{ELIO}$ can be a role name $\mathsf{r} \in \mathsf{N_R}$ or the inverse r^- of r. *Nominals* take the form $\{a\}$, where $a \in \mathsf{N_I}$ is an individual name. *Concepts* in $\mathcal{ELIO}$ have one of the following forms:

$$\top \mid \{\mathsf{a}\} \mid \mathsf{A} \mid C \sqcap D \mid \exists \mathsf{r}.C \mid \exists \mathsf{r}^-.C,$$

where $\mathsf{A} \in \mathsf{N_C}$, $\mathsf{r} \in \mathsf{N_R}$, and C and D range over concepts. We use r^- to denote s if $\mathsf{r} = s^-$ for $\mathsf{s} \in \mathsf{N_R}$ and identify $(\mathsf{r}^-)^-$ with r. We use the lowercase r to denote a role name and the uppercase R to denote a general role.

An $\mathcal{ELIO}$-ontology $\mathcal{O}$ is composed of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. A TBox $\mathcal{T}$ is a finite set of *axioms* of the form $C \sqsubseteq D$ (*general concept inclusion*, or GCI for short), where C and D are concepts. We use $C \equiv D$ as an abbreviation for the GCIs $C \sqsubseteq D$ and $D \sqsubseteq C$. An ABox $\mathcal{A}$ is a finite set of axioms of the form $C(a)$ (*concept assertions*) and the form $R(a, b)$ (role assertions), where $a, b \in \mathsf{N_I}$, C a concept, and R a role. We use $\mathsf{r}(a, b)$ to denote $\mathsf{r}^-(b, a)$ for r an inverse role. In a DL with nominals, ABox assertions can be internalized into TBox axioms, namely, $C(a)$ into $\{a\} \sqsubseteq C$ and $R(a, b)$ into $\{a\} \sqsubseteq \exists R.\{b\}$. Hence, in this paper, we assume w.l.o.g. that an $\mathcal{ELIO}$-ontology contains only GCIs.

Let $\mathcal{S} \in \mathsf{N_C} \cup \mathsf{N_R} \cup \mathsf{N_I}$ be a designated concept, role or individual name. A concept (GCI) is called an *$\mathcal{S}$-concept* (*$\mathcal{S}$-GCI*) if it contains $\mathcal{S}$. An occurrence of $\mathcal{S}$ is said to be *positive* (*negative*) in an $\mathcal{S}$-GCI if it occurs at the right-hand (left-hand) side of the GCI.

The semantics of $\mathcal{ELIO}$ is defined in terms of an *interpretation* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ is a nonempty set of elements, known as the *domain of the interpretation*, and $\cdot^{\mathcal{I}}$ denotes the *interpretation function* that maps every individual $a \in \mathsf{N_I}$ to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, every concept name $A \in \mathsf{N_C}$ to a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and every role name $r \in \mathsf{N_R}$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function $\cdot^{\mathcal{I}}$ is inductively extended to concepts as follows:

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}} \qquad (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$$
$$(\exists r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y.(x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$$
$$(r^-)^{\mathcal{I}} = \{(y, x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (x, y) \in r^{\mathcal{I}}\}$$

Let $\mathcal{I}$ be an interpretation. A GCI $C \sqsubseteq D$ is *true* in $\mathcal{I}$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. $\mathcal{I}$ is a *model* of an ontology $\mathcal{O}$, written $\mathcal{I} \models \mathcal{O}$, iff every GCI in $\mathcal{O}$ is *true* in $\mathcal{I}$. A GCI $C \sqsubseteq D$ is entailed by $\mathcal{O}$ (or a *logical entailment* of $\mathcal{O}$), written $\mathcal{O} \models C \sqsubseteq D$, iff $C \sqsubseteq D$ is true in every model $\mathcal{I}$ of $\mathcal{O}$.

A *signature* Σ is a finite subset of concept and role names from $\mathsf{N_C} \cup \mathsf{N_R}$. For any syntactic object X — which may include concepts, roles, GCIs, or ontologies — the sets $\mathrm{sig_C}(X)$ and $\mathrm{sig_R}(X)$ represent the concept and role names present in X, respectively. We further let $\mathrm{sig}(X) = \mathrm{sig_C}(X) \cup \mathrm{sig_R}(X)$.

Definition 1 (**Uniform Interpolation**). *Let $\mathcal{O}$ be an $\mathcal{ELIO}$-ontology and $\Sigma \subseteq sig(\mathcal{O})$ be a set of concept and role names called the interpolation signature. An $\mathcal{ELIO}$-ontology $\mathcal{V}$ is a uniform Σ-interpolant if the following conditions hold:*
(i) *$sig(\mathcal{V}) \subseteq \Sigma$, and*
(ii) *for any $\mathcal{ELIO}$-GCI $C \sqsubseteq D$ with $sig(C \sqsubseteq D) \subseteq \Sigma$, $\mathcal{V} \models C \sqsubseteq D$ iff $\mathcal{O} \models C \sqsubseteq D$.*

Interpolating an ontology $\mathcal{O}$ for a signature Σ can be achieved by *forgetting* the names not in Σ from $\mathcal{O}$ [28, 35, 52, 55, 56]. Therefore, interpolating $\mathcal{O}$ for Σ amounts to forgetting the names in $\mathrm{sig}(\mathcal{O})\backslash\Sigma$. In this sense, forgetting can be viewed as a dual problem to uniform interpolation.

Definition 2 (**Forgetting**). *Let $\mathcal{O}$ be an $\mathcal{ELIO}$-ontology and $\mathcal{S} \in sig(\mathcal{O})$ be a concept or role name. An $\mathcal{ELIO}$-ontology $\mathcal{V}$ is a* result of forgetting $\{\mathcal{S}\}$ from $\mathcal{O}$ *if the following conditions hold:*
(i) *$sig(\mathcal{V}) \subseteq sig(\mathcal{O})\backslash\{\mathcal{S}\}$, and*
(ii) *for any $\mathcal{ELIO}$-GCI $C \sqsubseteq D$ with $sig(C \sqsubseteq D) \subseteq sig(\mathcal{O})\backslash\{\mathcal{S}\}$, $\mathcal{V} \models C \sqsubseteq D$ iff $\mathcal{O} \models C \sqsubseteq D$.*
More generally, let $\mathcal{F} \subseteq sig(\mathcal{O})$ be a set of concept and role names, referred to as the forgetting signature. *An $\mathcal{ELIO}$-ontology $\mathcal{V}$ is a* result of forgetting $\mathcal{F}$ from $\mathcal{O}$ *if the following conditions hold:*
(i) *$sig(\mathcal{V}) \subseteq sig(\mathcal{O})\backslash\mathcal{F}$, and*
(ii) *for any $\mathcal{ELIO}$-GCI $C \sqsubseteq D$ with $sig(C \sqsubseteq D) \subseteq sig(\mathcal{O})\backslash\mathcal{F}$, $\mathcal{V} \models C \sqsubseteq D$ iff $\mathcal{O} \models C \sqsubseteq D$.*

Essentially, the forgetting process refines an ontology $\mathcal{O}$ into a more restricted perspective, $\mathcal{V}$, focusing sorely on a sub-signature Σ of $\mathcal{O}$. Here, Σ is defined as a subset of $\mathcal{O}$'s signature excluding $\mathcal{F}$, i.e., $\Sigma \subseteq \mathrm{sig}(\mathcal{O})\backslash\mathcal{F}$. Notably, within the $\mathcal{ELIO}$ framework,

$\mathcal{V}$ mirrors the behavior of $\mathcal{O}$, implying that they yield the same $\mathcal{ELIO}$-entailments w.r.t. Σ. It also follows from this definition that: (i) the result of forgetting $\mathcal{F}$ from $\mathcal{O}$ can be computed by eliminating single names in $\mathcal{F}$, regardless of the order of elimination, and (ii) the forgetting results are unique up to logical equivalence, meaning any two results, such as $\mathcal{V}_1$ and $\mathcal{V}_2$, derived from the same forgetting process, are logically equivalent despite potential differences in their explicit representations.

3 RELATED WORK

3.1 Forgetting and Existing Forgetting Methods

Forgetting can be formalized in two closely related ways. Initially, Lin and Reiter [30] formalized forgetting within first-order logic from a *model-theoretic* perspective, where forgetting a predicate P in a logical theory $\mathcal{T}$ yields a new theory $\mathcal{T}'$ characterized by models that concur with those of $\mathcal{T}$, except possibly in the interpretation of P. They further identified that for finite theories, the problem of forgetting a predicate amounts to the elimination of existential second-order quantifiers over predicates [11]. This alignment implies that the results of forgetting extend beyond first-order definability and they can be computed using second-order quantifier elimination methods such as Scan [10], Dls [46], Sqema [6], Msqel [44], etc. Lin and Reiter's notion of forgetting was later termed "strong forgetting" by Zhang and Zhou [54], who proposed from a *deductive* perspective an alternative "weak forgetting" for first-order logic. Here, the process of weakly forgetting a predicate P in a logical theory $\mathcal{T}$ yields the set $\mathcal{T}'$ comprising all first-order logical consequences of $\mathcal{T}$ irrelevant to P.[3] They further distinguished that weak forgetting yields results $\mathcal{T}_1$ that are generally weaker than those of strong forgetting $\mathcal{T}_2$, i.e., $\mathcal{T}_2 \models \mathcal{T}_1$, but these two notions coincide, i.e, $\mathcal{T}_2 \equiv \mathcal{T}_1$, when $\mathcal{T}_2$ is first-order definable. While always first-order definable, $\mathcal{T}_1$ may feature an infinite set of first-order formulas. In logic, weak forgetting has been explored as a problem analogous to, or potentially the dual of, uniform interpolation [8, 18, 50], a notion closely related to Craig interpolation [7], yet with more stringent constraints.

The definitions of strong and weak forgetting have been generalized to various DLs. In these contexts, they are characterized in terms of (model-theoretic or deductive) *inseparability* and *conservative extension* [13, 14, 21, 34]. Research in this domain has been focused on the following problems:

(i) Determining whether a DL $\mathcal{L}$ is closed under forgetting;
(ii) Analyzing the computational complexity of deciding if a forgetting result exists for $\mathcal{L}$; when it does, characterizing the dimensional attributes of such results;
(iii) Investigating the computational complexity of deriving forgetting results for $\mathcal{L}$;
(iv) Developing and optimizing practical methodologies to compute results of forgetting for $\mathcal{L}$.

Significant theoretical findings include:

(i) Very few DLs are known to be closed under forgetting, whether it be under the strong or weak notion, even when the source language is $\mathcal{EL}$ having very limited expressivity. This means that for a forgetting problem with a source language of $\mathcal{EL}$, a forgetting result within the expressivity of $\mathcal{EL}$ does not necessarily exist [34]. This also applies to $\mathcal{ALC}$ [13];
(ii) Determining whether a result exists for strong forgetting is undecidable on $\mathcal{EL}$ and $\mathcal{ALC}$ [21];
(iii) Determining whether a result exists for weak forgetting is ExpTime-complete on $\mathcal{EL}$ [32, 36] and 2ExpTime-complete on $\mathcal{ALC}$ [35];
(iv) For $\mathcal{EL}$ and $\mathcal{ALC}$, the result of weak forgetting can be triple exponential in size compared to the source ontology in the worst-case scenario [32, 35, 36].

Forgetting, despite its significant computational challenge, has been universally recognized for its tremendous potential in ontology-based knowledge processing. There has been a concerted effort toward developing and automating effective techniques for computing results of forgetting, particularly in various DLs.

Currently, the primary practical methods for forgetting are Lethe and Fame. Lethe [25] utilizes the classic resolution calculus [42] to compute uniform interpolants and handles ontologies in $\mathcal{ALC}$ and several its extensions. On the other hand, Fame [57] employs a monotonicity property known as Ackermann's Lemma to facilitate strong forgetting, catering specifically to $\mathcal{ALCOIH}$ ontologies. Although Nui and another resolution-based approach [53] designed for $\mathcal{EL}$ and $\mathcal{SHQ}$ were developed, they are no longer accessible. Consequently, this paper focuses on Lethe as the SOTA UI method and establishes it as our comparison baseline.

3.2 Existing Modularization Methods

As with forgetting, subset modules can be categorized into *model susbet modules* (model-theoretic notion) and *subsumption subset modules* (deductive notion). Based on specific properties, they can be further divided into four types: *plain*, *self-contained*, *weakly depleting*, and *strongly depleting*. Due to space constraints, we will not elaborate on these types here; instead, this information is included in the long version of this paper for reference. A module $\mathcal{M}$ is called a *minimal* Σ-module of $\mathcal{O}$ if there is no proper subset of $\mathcal{M}$ that is also a Σ-module of $\mathcal{O}$. In other words, every GCI in $\mathcal{M}$ is necessary for preserving the logical entailments over Σ.

Specifically, MEX[4] employs a specialized method to extract minimal, self-contained, and strongly depleting modules from acyclic $\mathcal{ELI}$-ontologies in polynomial time [22]. The same method has been extended to DL-Lite ontologies in [24]. AMEX[5] further extends MEX to acyclic $\mathcal{ALCIQ}$ ontologies, maintaining self-containment and strong depletingness, though sacrificing efficiency [12]. Locality-based methods (LBMs) can extract self-contained and strongly depleting modules from $\mathcal{SROIQ}$-ontologies in polynomial time and are included as interfaces in the OWL API[6] [14]. There are various types of LBMs according to how they test the locality of each GCI, namely syntactic $\bot$ locality (BOT), syntactic $\top$ locality (TOP) and their nested version, syntactic $\bot\top^*$ locality (STAR). LBMs have two main extensions, namely reachability-based methods (RBMs) and Datalog-based methods (DBMs). RBMs, implemented by CEL for extracting modules from $\mathcal{SRIQ}$-ontologies [38], are divided into $\bot$,

[3]A sentence is deemed irrelevant to a predicate P if it is logically equivalent to another sentence that does not contain P.

[4]https://cgi.csc.liv.ac.uk/ konev/software/
[5]https://github.com/sadger/module-extraction
[6]http://owlapi.sourceforge.net

$\top$ and $\bot\top^*$. $\bot$-reachability is identical to $\bot$-locality if the original ontologies are in a certain normal form. Modules extracted by the other types of reachability are usually subsets of those extracted by corresponding LBMs and are neither self-contained nor strongly depleting, meaning RBMs cannot capture all the information related to Σ. DBMs, implemented by PrisM[7] [43], can extract self-contained and weakly depleting modules for $\mathcal{SROIQ}$-ontologies in polynomial time.

Except for MEX, all other approaches focus on model modules and are limited to computing approximations of such modules. In contrast, subsumption modules receive less attention because the deductive notion does not guarantee *robustness under replacement*, an important property for safely importing modules into other contexts [21]. To the best of our knowledge, the methods proposed by [5] and [40] are the only approaches for extracting subsumption modules, but neither is currently accessible. Hence, in this paper, we consider TOP, BOT, STAR, MEX, AMEX, and PrisM as our baselines for comparison.

4 NORMALIZATION OF $\mathcal{ELIO}$-ONTOLOGIES

Given an $\mathcal{ELIO}$-ontology $\mathcal{O}$ and a forgetting signature $\mathcal{F}$,[8], our method computes the result of forgetting by iteratively eliminating single names from $\mathcal{F}$. The single name elimination is achieved by two independent calculi that operate on specialized normal forms tailored to $\mathcal{ELIO}$-ontologies.

4.1 A-Normal Form (A-NF)

Definition 3 (A-**Normal Form**). *We say that a GCI is in* A-normal form *(or* A-NF*) if it has one of the following forms, where*

	A-NF		*A-NF*
I	$C \sqsubseteq \mathsf{A}$	*IV*	$\mathsf{A} \sqcap E \sqsubseteq F$
II	$C \sqsubseteq \exists R.(\mathsf{A} \sqcap D)$	*V*	$\exists S.(\mathsf{A} \sqcap E) \sqcap F \sqsubseteq G$

(i) R and S are general roles, and (ii) C, D, E, F, and G are general concepts (not necessarily concept names) that do not contain A. *An $\mathcal{ELIO}$-ontology $\mathcal{O}$ is in* A-NF *if every* A*-GCI in $\mathcal{O}$ is in A-NF.*

The transformation of an $\mathcal{ELIO}$-ontology $\mathcal{O}$ into A-NF can be done by exhaustive application of the following normalization rules (NR1 – NR5) to every A-GCI in $\mathcal{O}$ that has yet to be in A-NF. Below, X, Y, Y_1, and Y_2 are identified as $\mathcal{ELIO}$-concepts. Additionally, the term "definer", denoted by Z, refers to newly introduced concept names that act as "abbreviations" for compound concepts during normalization.

NR1 For each instance of a GCI $X \sqsubseteq Y_1 \sqcap Y_2$, if either Y_1 or Y_2 contains A, replace it with $X \sqsubseteq Y_1$ and $X \sqsubseteq Y_2$;

NR2 For each instance of $X \sqsubseteq Y$, if A occurs more than once in it and a concept of the form $\exists R.C$ is present at the surface level of the left-hand side X, where R is a role and C is a concept containing A, replace C with a new definer $Z \in \mathsf{N_C}$ and add $C \sqsubseteq Z$ to $\mathcal{O}$;

NR3 For each instance of $X \sqsubseteq Y$, if A occurs more than once in it and a concept of the form $\exists R.C$ is present at the surface level of the right-hand side Y, where R is a role and C is a concept containing A, replace C with a new definer $Z \in \mathsf{N_C}$ and add $C \sqsubseteq Z$ to $\mathcal{O}$;

NR4 For each instance of $X \sqsubseteq Y$, if A occurs exactly once in it and a concept of the form $\exists R.C$ is present at the surface level of the left-hand side X, where R is a role — and provided that C contains A but is not in the form of $\mathsf{A} \sqcap E$ as specified by A-NF V or VI, replace C with a new definer $Z \in \mathsf{N_C}$ and add $C \sqsubseteq Z$ to $\mathcal{O}$;

NR5 For each instance of $X \sqsubseteq Y$, if A occurs exactly once in it and a concept of the form $\exists R.C$ is present at the surface level of the right-hand side Y, where R is a role — and provided that C contains A but is not in the form of $\mathsf{A} \sqcap D$ as specified by A-NF II or III, replace C with a new definer $Z \in \mathsf{N_C}$ and add $C \sqsubseteq Z$ to $\mathcal{O}$;

In developing a normalization method, it is essential to ensure the following properties for its effectiveness and efficiency:

(i) *Soundness* of the method: we must ensure that the resulting $\mathcal{O}'$ computed by the method holds the same logical consequences as the original one $\mathcal{O}$ w.r.t. their shared signature. This is necessary to satisfy Condition (ii) of Definition 2. Here, the shared signature corresponds to the signature of $\mathcal{O}$ — $\mathrm{sig}(\mathcal{O})$ — as the normalization process may involve the introduction of fresh definers.

(ii) *Completeness* of the method: we must ensure that any $\mathcal{ELIO}$-ontology can be transformed into A-NF using the method;

(iii) *Efficiency* of the method: we must ensure that the normalization process is terminating and completes in polynomial time.

Lemmas 1 and 2 below state the soundness of the normalization methods, while Lemma 3 states their termination and completeness.

Lemma 1. *Let $\mathcal{O}$ be an $\mathcal{ELIO}$-ontology and $\mathcal{O}'$ the normalized one obtained from $\mathcal{O}$ using NR1 – NR5. Then we have*

$$\mathcal{O} \models C \sqsubseteq D \text{ iff } \mathcal{O}' \models C \sqsubseteq D,$$

for any $\mathcal{ELIO}$-GCI $C \sqsubseteq D$ with $sig(C \sqsubseteq D) \subseteq sig(O)$.

4.2 R-Normal Form (R-NF)

Definition 4 (R-**Normal Form**). *We say that a GCI is in* r-normal form *(or* r-NF*) if it has one of the following forms, where (i)*

	r-NF		*r-NF*
I	$C \sqsubseteq \exists r.D$	*III*	$E \sqcap \exists r.F \sqsubseteq G$
II	$C \sqsubseteq \exists r^-.D$	*IV*	$E \sqcap \exists r^-.F \sqsubseteq G$

$\mathsf{r} \in N_R$, and (ii) C, D, E, F, and G are concepts that do not contain r. *An $\mathcal{ELIO}$-ontology $\mathcal{O}$ is in* r-NF *if every* r*-GCI in $\mathcal{O}$ is in r-NF.*

One can compute the r-NF of a given $\mathcal{ELIO}$-ontology $\mathcal{O}$ using a slightly adjusted approach for A-NF transformation.

Lemma 2. *Let $\mathcal{O}$ be an $\mathcal{ELIO}$-ontology and $\mathcal{O}'$ the normalized one obtained from $\mathcal{O}$ using the slightly adjusted normalization rules for r-NF transformation. Then we have*

$$\mathcal{O} \models C \sqsubseteq D \text{ iff } \mathcal{O}' \models C \sqsubseteq D,$$

for any $\mathcal{ELIO}$-GCI $C \sqsubseteq D$ with $sig(C \sqsubseteq D) \subseteq sig(O)$.

[7] https://github.com/anaphylactic/PrisM

[8] Note that $\mathcal{F}$ is not necessarily confined to a subset of $\mathrm{sig}(\mathcal{O})$, as Definition 2 indicates that forgetting a name not present in $\mathcal{O}$ leads to no change to $\mathcal{O}$. Hence, forgetting a name not present in $\mathcal{O}$ yields $\mathcal{O}$ itself as the result.

Lemma 3. *Let $\mathcal{O}$ be an $\mathcal{ELIO}$-ontology. Then $\mathcal{O}$ can be transformed into A-NF or r-NF $\mathcal{O}'$ by a linear number of applications of the corresponding normalization rules. In addition, the size of the resulting ontology $\mathcal{O}'$ is linear in the size of $\mathcal{O}$.*

5 DEFINER INTRODUCTION STRATEGY

Our forgetting method introduces a novel normal form and utilizes a non-traditional, yet notably cost-effective, linear definer introduction strategy for normalization. This strategy significantly enhances the method's efficiency.

We delve into the intricacies of how it employs definers to facilitate the normalization of ontologies. Lethe works on clauses of the form $L_1 \sqcup \ldots \sqcup L_k$, where each L_i $(1 \leq i \leq k)$ is a TBox literal:

$$\mathsf{A} \mid \neg\mathsf{A} \mid \exists\mathsf{r}.Z \mid \exists\mathsf{r}^{-}.Z \mid \forall\mathsf{r}.Z \mid \forall\mathsf{r}^{-}.Z,$$

where $\mathsf{r} \in \mathsf{N_R}$ and $\mathsf{A}, Z \in \mathsf{N_C}$. A salient observation is that Lethe mandates every Z (any subconcept immediately below a $\exists$- or $\forall$-restriction) to be a definer, at any stage of the forgetting process. In contrast, our method allows for a more flexible specification of Z. This profoundly affects the general applicability of the inference rules employed by these methods, and by extension, has significant implications for their inferential efficiency.

For a deeper algorithmic understanding of Lethe's strategy for introducing definers, we fix some notations. By $\mathsf{sig_D}(\mathcal{O})$ we denote the set of the definers introduced in $\mathcal{O}$, by $\mathsf{Sub}^{\forall}_{\exists}(\mathcal{O})$ the set of all subconcepts of the form $\exists\mathsf{r}^{(-)}.X$ or $\forall\mathsf{r}^{(-)}.X$ in $\mathcal{O}$, where $\mathsf{r} \in \mathsf{N_R}$ and X an arbitrary concept, and by $\mathsf{Sub}_X(\mathcal{O})$ the set of all subconcepts X in $\mathcal{O}$ with $\exists\mathsf{r}^{(-)}.X \in \mathsf{Sub}^{\forall}_{\exists}(\mathcal{O})$ or $\forall\mathsf{r}^{(-)}.X \in \mathsf{Sub}^{\forall}_{\exists}(\mathcal{O})$.

In the Lethe framework, which exploits a definer reuse strategy (i.e., Lethe consistently utilizes a definer to refer to the identical subconcepts), an injective function f can be defined over $\mathsf{sig_D}(\mathcal{O})$, namely $f : \mathsf{sig_D}(\mathcal{O}) \rightarrow \mathsf{Sub}_X(\mathcal{O})$. f also exhibits surjectivity, given Lethe's exhaustive manner to introduce defines — Lethe mandates every subconcept immediately below an $\exists$- or $\forall$-restriction to be a definer. On the other hand, within our method's framework, f is defined as non-surjective. However, for both methods, the number of definers — denoted as $|\,\mathsf{sig_D}(\mathcal{O})|$ — relevant to the normalization of $\mathcal{O}$, is bounded by $\mathrm{O}(n)$. Here, n corresponds to the count of $\exists$- and $\forall$-restrictions present in $\mathcal{O}$. This implies a linear growth in the introduction of definers.

Lethe employs a saturation-based reasoning approach to facilitate the elimination of single names from an ontology $\mathcal{O}$. This process involves the generation of new entailments, which are subsequently added to $\mathcal{O}$, using a generalized resolution calculus, referred to as Res, as described in [25]. Lethe transforms $\mathcal{O}$ into normal form through a bifurcated methodology. The initial phase, termed as the *pre-resolution phase*, witnesses Lethe's inaugural computation of $\mathcal{O}$'s normal form to fire up Res. As previously examined, this stage features a *linear* and *static* introduction of definers.

However, transitioning to the second *intra-resolution stage*, Lethe exhaustively applies Res's inference rules on $\mathcal{O}$ until saturation is achieved at $\mathsf{Res}(\mathcal{O})$, where new entailments are propagated from existing ones. For example, applying the $\forall\exists$*-role propagation* rule to $C_1 \sqcup \forall r.D_1$ and $C_2 \sqcup \exists r.D_2$ yields a new entailment $C_1 \sqcup C_2 \sqcup \exists r.(D_1 \sqcap D_2)$, for which Lethe has to introduce a fresh definer $D_{12} \in \mathsf{N_C}$ to replace $D_1 \sqcap D_2$ for normalization, and then adds $\neg D_{12} \sqcup (D_1 \sqcap D_2)$ to $\mathcal{O}$ — definers are *dynamically* introduced as Res iterates over $\mathcal{O}$. Following this process, an additional injective yet non-surjective function f' emerges over $\mathsf{sig_D}(\mathsf{Res}(\mathcal{O}))$, defined as $f : \mathsf{sig_D}(\mathsf{Res}(\mathcal{O})) \rightarrow \mathsf{Sub}_X(\mathsf{Res}(\mathcal{O}))$, and the size of the codomain $|\mathsf{Sub}_X(\mathsf{Res}(\mathcal{O}))| = 2^{|\mathsf{Sub_D}(\mathcal{O})|}$. The number of the definers for the normalization of $\mathsf{Res}(\mathcal{O})$ is bounded by $\mathrm{O}(2^n)$, where n is the number of $\exists$- and $\forall$-restrictions in $\mathcal{O}$. Thus, Lethe introduces definers at an exponential rate during the intra-resolution stage. In contrast, our forgetting method restricts its normalization endeavors within the pre-resolution stage, indicating a linear trajectory in the introduction of definers during its entire forgetting span.

Definers are extraneous to the desired signature and thus should be excluded from the result of forgetting $\mathcal{F}$ from $\mathcal{O}$. Consequently, in the worst case, Lethe will be tasked with discarding as many as $(2^n) + |\mathcal{F}|$ names and executing Res for $\mathrm{O}(2^n) + |\mathcal{F}|$ iterations to compute the forgetting result. In contrast, our method introduces a maximum of n definers, and in the worst case, only needs to activate the forgetting calculus (described next) $n + |\mathcal{F}|$ times.

6 SINGLE NAME ELIMINATION

6.1 Calculus for Eliminating A

Our forgetting method exploits a two-step calculus to eliminate a single concept name A from $\mathcal{O}$:

Step I: computes A-NF of $\mathcal{O}$ as described above;

Step II: eliminates A by exhaustive application of the inference rules in Figure 2 to the normalized $\mathcal{O}$.

While Lethe and Fame work on the clausal representation of $\mathcal{O}$, our method directly addresses GCIs.

IR1. $C \sqsubseteq \mathsf{A}, \mathsf{A} \sqcap E \sqsubseteq G \implies C \sqcap E \sqsubseteq G$

IR2. $C \sqsubseteq \mathsf{A}, \exists R.(\mathsf{A} \sqcap E) \sqcap F \sqsubseteq G \implies \exists R.(C \sqcap E) \sqcap F \sqsubseteq G$

IR3. $C \sqsubseteq \exists R.(\mathsf{A} \sqcap D), \mathsf{A} \sqcap E_1 \sqsubseteq G_1, \ldots, \mathsf{A} \sqcap E_n \sqsubseteq G_n$
$\implies C \sqsubseteq \exists R.(E_1 \sqcap \ldots \sqcap E_n \sqcap G_1 \sqcap \ldots \sqcap G_n \sqcap D)$
provided that: $\mathcal{O} \models \mathsf{A} \sqcap D \sqsubseteq E_1 \sqcap \cdots \sqcap E_n$
$C \sqsubseteq \exists R.(\mathsf{A} \sqcap D), \mathsf{A} \sqcap E \sqsubseteq G \implies C \sqsubseteq \exists R.D$
provided that: $\mathcal{O} \not\models \mathsf{A} \sqcap D \sqsubseteq E$

IR4. $C \sqsubseteq \exists R.(\mathsf{A} \sqcap D), \exists S.(\mathsf{A} \sqcap E) \sqcap F \sqsubseteq G$
$\implies C \sqsubseteq \exists R.D, C \sqcap F \sqsubseteq G$
provided that: $\mathcal{O} \models \mathsf{A} \sqcap D \sqsubseteq E$ and $\mathcal{O} \models R \sqsubseteq S$
$C \sqsubseteq \exists R.(\mathsf{A} \sqcap D), \exists S.(\mathsf{A} \sqcap E) \sqcap F \sqsubseteq G \implies C \sqsubseteq \exists r.D$
provided that: $\mathcal{O} \not\models \mathsf{A} \sqcap D \sqsubseteq E$ or $\mathcal{O} \not\models R \sqsubseteq S$

Figure 2: Inference rules for eliminating A

Given an $\mathcal{ELIO}$-ontology in A-NF. We start with the A-GCIs in $\mathcal{O}$ and add implied GCIs using appropriate inference rules shown in Figure 2. The rules are not concrete rules, but rule schemata. To create a concrete instance of such a rule schema, the meta-variables A, C, D, E, F, and G must be replaced by concrete concepts, and the meta-variables R and S by concrete roles. Each rule is bifurcated into two parts: the *premises* on the lefthand side of the notation $\Longrightarrow$ and the *conclusion* on the righthand side. The exhaustive application of the inference rules aims to reveal all implicit logical entailments concerning sig($\mathcal{O}$)\{A}, and add these new entailments into the current $\mathcal{O}$. This process continues until $\mathcal{O}$ is saturated w.r.t. A. We say that $\mathcal{O}$ is saturated w.r.t. A if no inference rule can be applied to it (applying any of the inference rules only leads to the addition of a GCI that is already present in $\mathcal{O}$).

Revealing implicit entailments from $\mathcal{O}$ involves combining every positive occurrence (GCIs taking the form A-NF I or II) of A with every negative occurrence (GCIs taking the form A-NF III or IV). This results in four distinct combination cases, labeled as IR1 — IR4, respectively, as depicted in Figure 2. The idea of the combination follows the classic binary resolution principle [42]. This principle, applied to pairs of clauses containing complementary literals (e.g., $A \vee B$ and $\neg B \vee C$), derives a new clause (e.g., $A \vee C$) by combining all the literals of both clauses. It is easy to verify that the resolvent $\{A \vee C\}$ is equivalent to the premises $\{A \vee B, \neg B \vee C\}$ up to the excluded name B.

While unlike propositional resolution, which directly resolves on propositional variables, in $\mathcal{ELIO}$, the resolution targets a concept name A that may fall under an $\exists$-restriction, or occur as a conjunct of $\sqcap$. The complementarity between these literals only becomes evident via their first-order translations; see [58] for an example. Our inference rule directly resolves the literals of the forms $\exists$ and $\sqcap$ on r, yielding a forgetting result in DL. Through the exhaustive application of these inference rules until saturation, followed by the removal of all A-GCIs from the saturated $\mathcal{O}$, the result is a refined ontology, denoted as $\mathcal{O}^{-\mathsf{A}}$, devoid of any occurrences of A.

Lemma 4. *Let $\mathcal{O}$ be an $\mathcal{ELIO}$-ontology in A-NF, and $\mathcal{O}^{-A}$ the ontology obtained by eliminating A from $\mathcal{O}$ using the inference rules in Figure 2, then we have:*

$$\mathcal{O} \models C \sqsubseteq D \textit{ iff } \mathcal{O}^{-A} \models C \sqsubseteq D,$$

for any $\mathcal{ELIO}$-GCI $C \sqsubseteq D$ with sig$(C \sqsubseteq D) \subseteq$ sig$(\mathcal{O})\backslash\{A\}$.

Lemma 4 establishes the partial soundness of the calculus. Specifically, the derived ontology $\mathcal{O}^{-\mathsf{A}}$ fulfills the second condition necessary for it to be the result of forgetting {A} from $\mathcal{O}$. However, $\mathcal{O}^{-\mathsf{A}}$ may include definers which fall outside the scope of sig($\mathcal{O}$)\{A}, potentially failing to fulfill the first condition. A discussion on this will follow shortly.

6.2 Calculus for Eliminating r

The calculus for role name elimination parallels that for concept name elimination, which proceeds in two steps as well.

Step I: computes the r-NF of $\mathcal{O}$ as described previously;
Step II: eliminates r by exhaustive application of the inference rules in Figure 3 to the normalized $\mathcal{O}$.

Revealing implicit entailments from $\mathcal{O}$ involves the combination of every positive occurrence (GCIs taking the form r-NF I or II) of r with every negative occurrence (GCIs taking the form r-NF III and IV) (i.e., resolving on r). This results in four distinct combination cases, labeled as IR5, IR6, IR7, and IR8, as depicted in Figure 3. By the exhaustive application of these inference rules until saturation, followed by the removal of all r-GCIs, the result is a refined ontology, denoted as $\mathcal{O}^{-\mathsf{r}}$, devoid of any occurrences of r.

IR5. $C \sqsubseteq \exists \mathsf{r}.D, F \sqcap \exists \mathsf{r}.E \sqsubseteq G \implies F \sqcap C \sqsubseteq G$
provided that: $\mathcal{O} \models \exists \mathsf{r}.D \sqsubseteq \exists \mathsf{r}.E$

IR6. $C \sqsubseteq \exists \mathsf{r}.D, F \sqcap \exists \mathsf{r}^-.E \sqsubseteq G \implies F \sqcap C \sqsubseteq G$
provided that: $\mathcal{O} \models \exists \mathsf{r}.D \sqsubseteq \exists \mathsf{r}^-.E$

IR7. $C \sqsubseteq \exists \mathsf{r}^-.D, F \sqcap \exists \mathsf{r}.E \sqsubseteq G \implies F \sqcap C \sqsubseteq G$
provided that: $\mathcal{O} \models \exists \mathsf{r}^-.D \sqsubseteq \exists \mathsf{r}.E$

IR8. $C \sqsubseteq \exists \mathsf{r}^-.D, F \sqcap \exists \mathsf{r}^-.E \sqsubseteq G \implies F \sqcap C \sqsubseteq G$
provided that: $\mathcal{O} \models \exists \mathsf{r}.D \sqsubseteq \exists \mathsf{r}^-.E$

Figure 3: Inference rule for eliminating r

Lemma 5. *Let $\mathcal{O}$ be an $\mathcal{ELIO}$-ontology in r-NF, and $\mathcal{O}^{-r}$ the ontology obtained from eliminating r from $\mathcal{O}$ using the inference rules in Figure 3, then we have:*

$$\mathcal{O} \models C \sqsubseteq D \textit{ iff } \mathcal{O}^{-r} \models C \sqsubseteq D,$$

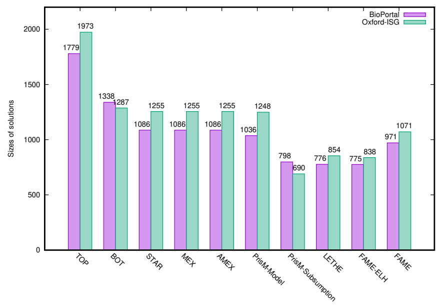
for any $\mathcal{ELIO}$-GCI $C \sqsubseteq D$ with sig$(C \sqsubseteq D) \subseteq$ sig$(\mathcal{O})\backslash\{r\}$.

Note that an external DL reasoner is utilized to check the side conditions of the inference rules in both figures. It is known that subsumption checking in $\mathcal{ELIO}$ is ExpTime-complete [2].

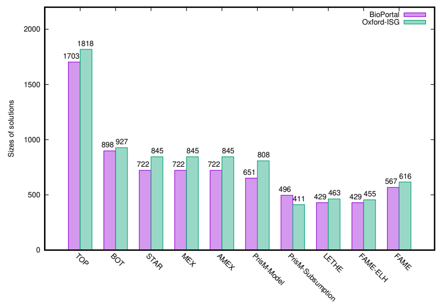
6.3 The Forgetting Process

Our method takes an $\mathcal{ELIO}$-ontology $\mathcal{O}$ and a forgetting signature $\mathcal{F}$ including the concept and role names to be eliminated from $\mathcal{O}$. Notably, the elimination sequence is flexible, accommodating any order specified by the user. The single-name elimination procedure is iteratively applied to each name in $\mathcal{F}$, each time yielding an intermediate result. Throughout this process, definers may be introduced. As per Definition 2, definers should not be present in the final result of forgetting. Therefore, once all names in $\mathcal{F}$ have been addressed, any introduced definers must also be eliminated from the intermediate result. However, there is no guarantee of completely removing all definers. For example, consider an attempt to forget A from an $\mathcal{ELIO}$-ontology $\{\mathsf{A} \sqsubseteq \exists r^-.\mathsf{A}\}$, which exhibits cyclic behavior over A. The result is $\{\mathsf{D}_1 \sqsubseteq \exists r^-.\mathsf{D}_1\}$, with D1 $\in \mathsf{N_D}$ as a new definer. If one tried to forget D_1 from this result, it would yield a GCI of the same pattern $\{\mathsf{D}_2 \sqsubseteq \exists r^-.\mathsf{D}_2\}$, with D2 $\in \mathsf{N_D}$ as a newer definer. This would result in an endless introduction of definers. While cyclic situations might be tackled using fixpoints [4], as shown by Lethe, mainstream reasoning tools and the OWL API do not support fixpoints. Our method does not adopt the extension of fixpoints as a solution. Instead, it guarantees the termination of forgetting by giving up forgetting D_1, leaving it in the resulting ontology, and declaring an unsuccessful forgetting attempt. This highlights the inherent unsolvability of the forgetting problem.

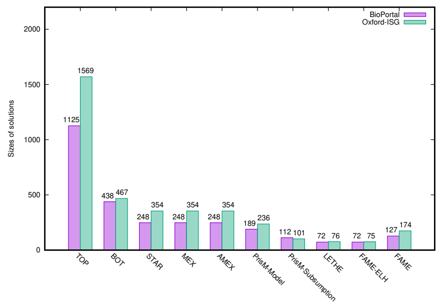
Theorem 1. *Given any $\mathcal{ELIO}$-ontology $\mathcal{O}$ and any forgetting signature $\mathcal{F} \subseteq$ sig$(\mathcal{O})$ as input, our forgetting method always terminates and returns an $\mathcal{ELIO}$-ontology $\mathcal{V}$. If $\mathcal{V}$ does not contain any definers, then it is a result of forgetting $\mathcal{F}$ from $\mathcal{O}$ and a uniform Σ-interpolant of $\mathcal{O}$ for $\Sigma =$ sig$(\mathcal{O})\backslash\mathcal{F}$.*



(a) Forgetting 50% of names in sig$(\mathcal{O})$

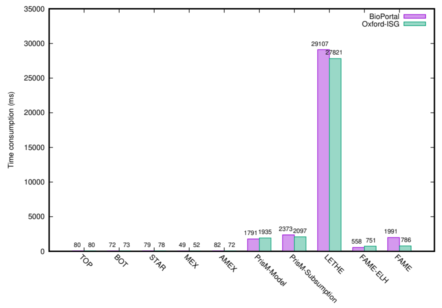


(b) Forgetting 70% of names in sig$(\mathcal{O})$

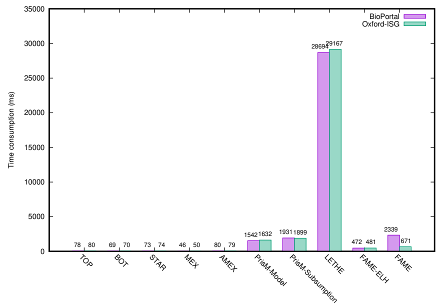


(c) Forgetting 90% of names in sig$(\mathcal{O})$

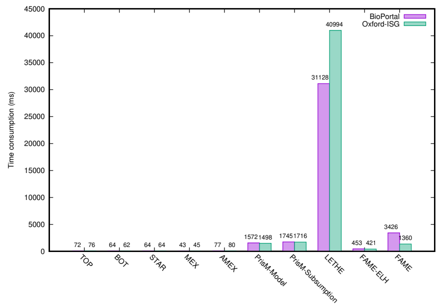
Figure 4: Average |Onto| in output ontologies



(a) Forgetting 50% of names in sig$(\mathcal{O})$

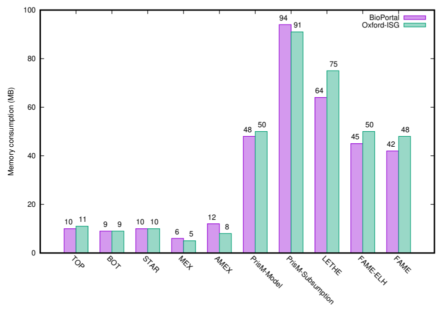


(b) Forgetting 70% of names in sig$(\mathcal{O})$

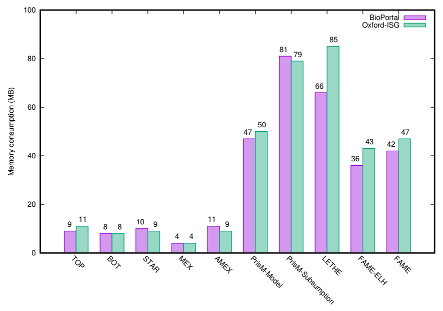


(c) Forgetting 90% of names in sig$(\mathcal{O})$

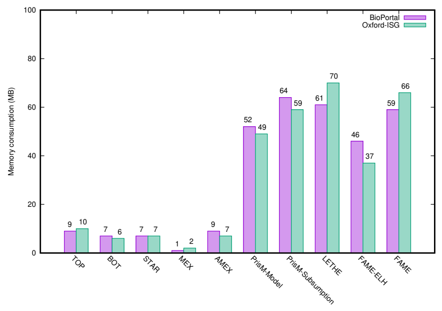
Figure 5: Average Time Consumption



(a) Forgetting 50% of names in sig$(\mathcal{O})$



(b) Forgetting 70% of names in sig$(\mathcal{O})$



(c) Forgetting 90% of names in sig$(\mathcal{O})$

Figure 6: Average Memory Consumption

7 EXPERIMENTS

We have developed a prototype (Proto) of our forgetting method in Java using OWL API Version 5.1.7[9]. To evaluate its performance and practicality, we compared this prototype against the SOTA forgetting and UI method LETHE[10], and a bunch of standard modularization tools introduced previously, using two large corpora of real-world ontologies. The first corpus, taken from the Oxford ISG Library[11], included a wide range of ontologies from multiple sources. The second corpus, a March 2017 snapshot from NCBO BioPortal[12], specifically featured biomedical ontologies.

From the Oxford ISG, we selected 488 ontologies with GCI counts not exceeding 10,000. We excluded those with cyclic dependencies or lacking ∃-restrictions or inverse roles to remove trivial cases; this left us with 177 ontologies. We then distilled these ontologies to their $\mathcal{ELIO}$-fragments by discarding GCIs not expressible in

[9]http://owlcs.github.io/owlapi/
[10]https://lat.inf.tu-dresden.de/ koopmann/LETHE/
[11]http://krr-nas.cs.ox.ac.uk/ontologies/lib/
[12]https://zenodo.org/records/439510

$\mathcal{ELIO}$. Applying the same strategy to BioPortal, we obtained a collection of 76 ontologies. A comprehensive breakdown of the refined ontologies is available in the extended version of this paper.

The creation of the forgetting signature $\mathcal{F}$ varies according to the specific requirements of different tasks. To address this variability, we designed three evaluation configurations to forget 50%, 70%, and 90% of the concept and role names in the signature of each ontology. We utilized a shuffling algorithm to ensure randomized selection of $\mathcal{F}$. Our experiments were conducted on a laptop equipped with an Intel Core i7-9750H processor with 6 cores, capable of reaching up to 2.70 GHz, and 12 GB of DDR4-1600 MHz RAM. For consistent performance assessment, we set a maximum runtime of 300 seconds and a heap space limit of 9GB. An experiment was deemed *successful* if it met the following criteria: (i) all names specified in $\mathcal{F}$ were successfully eliminated; (ii) no definers were present in the output, if introduced during the process; (iii) completion within the 300-second limit; and (iv) operation within the set 9GB space limit. We repeated the experiments 100 times for each test case and took the average to validate our findings.

A notable result is that our method achieved a 100% success rate across all evaluation tracks. In contrast, Lethe had average success rates of 86.57%, 74.88%, and 70.16% on Oxford-ISG ontologies for forgetting 50%, 70%, and 90% of names, respectively. On the BioPortal dataset, these figures were 85.14%, 73.11%, and 69%, respectively. Most failures were due to timeouts and memory overflows, underscoring the importance of computational efficiency and memory consumption for the success of a forgetting method.

Next, we delve into the inherent properties of our method and the nature of its forgetting results by conducting a comprehensive comparison with various prevalent modularization methods, focusing on metrics such as result size, computation time, and memory consumption.

Regarding result size (see Figure 4), surprisingly, as a forgetting method, our prototype's output did not reflect the theoretical projections of a triple exponential size increase compared to the input ontologies. On the contrary, the forgetting results demonstrated impressive compactness in comparison to the input ontologies, surpassing even the results produced by modularization techniques; these modules are mere syntactic subsets of the input ontologies. Note that many works, e.g., [35, 36], emphasize the impracticality of forgetting and UI due to its exponential space complexity; however, this is completely contrary to the situation in practice.

Regarding time consumption, the results in Figure 5 were equally astonishing. Our prototype, when performing the same forgetting tasks, outpaced LETHE by what could be described as "light years". Moreover, its speed was even on par with the fastest modularization methods, despite the well-known fact that the computational complexity of forgetting is generally significantly greater than that of modularization [3, 49].

Regarding memory consumption (see Figure 6), forgetting typically required more memory during computation than modularization. This is due to the need to compute a large number of new entailments in the forgetting process. However, when comparing different forgetting methods, our prototype still outperformed the rest. It generally consumed only 60 - 80% of the memory used by other forgetting methods, primarily due to its effective control over definer introduction.

Table 1: Definers introduced during forgetting

Ontology Name	Lethe (50%)	Proto (50%)	Lethe (70%)	Proto (70%)	Lethe (90%)	Proto (90%)
DUO	1	0	3	0	3	0
ARO	1304	0	2559	0	3162	5
PCAO	15	0	24	0	45	0
AMPHX	190	0	831	0	1334	656
FAO	4	0	15	0	21	0
DMTO	287	0	1307	0	1244	0
HIO	6	0	38	0	47	0
HSAPDV	75	0	475	388	421	16
LMHA	70	0	254	0	486	51
PMDO	2	0	12	0	16	0
MMUSDV	122	29	92	0	173	3
HANCESTRO	44	0	118	0	178	0
EMAPA	4088	0	9295	0	18035	2302
PREO	5	0	17	0	22	0
AEO	18	0	46	0	101	0
EOL	1	0	1	0	1	0
LHN	30	4	34	3	34	0
ORDO	15892	0	35407	78	40911	1898
ORNASEQ	3	0	2	0	5	0
COGAT	78	0	272	0	376	0

Table 1 presents a detailed comparison of the definers introduced by Lethe and our prototype across several sample forgetting tasks; full results can be consulted in the long version of this paper. For any given task, our prototype consistently introduced significantly fewer definers than Lethe. Specifically, in the Oxford-ISG cases, when forgetting 50%, 70%, and 90% of the names in the ontology, Lethe required definers in 65.0%, 64.1%, and 60.7% of the tasks, respectively. In contrast, our prototype needed definers in only 23.1%, 22.2%, and 12.0% of the tasks. Similarly, in the BioPortal cases, Lethe necessitated definers in 26.3% of tasks across all three forgetting settings, whereas our prototype significantly reduced this need to just 9.2%, 3.9%, and 2.6%, respectively. These statistics highlight the superior efficiency of our prototype in minimizing the introduction of definers. Since these definers must eventually be forgotten from the end output, the large number of definers introduced imposes a significant computational burden on the entire forgetting process. This is the primary reason our method was much faster than Lethe and could even rival the efficiency of modularization methods.

8 CONCLUSION AND FUTURE WORK

Uniform interpolation is an advanced reasoning procedure that can be used to extract signature-restricted modules. However, researchers tend to believe that its computational properties make it less practical compared to modularization methods. In this paper, we present a highly efficient forgetting method for computing uniform interpolants of $\mathcal{ELIO}$-ontologies with ABoxes. We demonstrate that uniform interpolants can be computed as efficiently as subset modules through sophisticated normalization and definer introduction strategies. Additionally, an empirical evaluation showcases its algorithmic ascendancy over the SOTA UI method. This provides the community with a powerful tool for fostering knowledge reuse while respecting the signature constraints.

Our immediate next step for future work is to extend the current forgetting method to accommodate more expressive DLs, such as $\mathcal{ALC}$ and its major decidable extensions [47].

REFERENCES

[1] Grigoris Antoniou and Frank van Harmelen. 2004. *Web Ontology Language: OWL.* Springer Berlin Heidelberg, 67–92.

[2] Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. 2017. *An Introduction to Description Logic.* Cambridge University Press.

[3] Elena Botoeva, Boris Konev, Carsten Lutz, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. 2016. Inseparability and Conservative Extensions of Description Logic Ontologies: A Survey. In *Proc. Reasoning Web'16 (LNCS, Vol. 9885).* Springer, 27–89.

[4] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. 1999. Reasoning in Expressive Description Logics with Fixpoints based on Automata on Infinite Trees. In *Proc. IJCAI'99.* Morgan Kaufmann, 84–89.

[5] Jieying Chen, Michel Ludwig, and Dirk Walther. 2018. Computing Minimal Subsumption Modules of Ontologies. In *Proc. GCAI'18 (EPiC Series in Computing, Vol. 55).* EasyChair, 41–53.

[6] Willem Conradie, Valentin Goranko, and Dimiter Vakarelov. 2006. Algorithmic correspondence and completeness in modal logic. I. The core algorithm SQEMA. *Log. Methods Comput. Sci.* 2, 1 (2006).

[7] William Craig. 1957. Three Uses of the Herbrand-Gentzen Theorem in Relating Model Theory and Proof Theory. *J. Symb. Log.* 22, 3 (1957), 269–285.

[8] Giovanna D'Agostino and Marco Hollenberg. 2000. Logical Questions Concerning The μ-Calculus: Interpolation, Lyndon and Los-Tarski. *J. Symb. Log.* 65, 1 (2000), 310–332.

[9] Warren Del-Pinto and Renate A. Schmidt. 2019. ABox Abduction via Forgetting in $\mathcal{ALC}$. In *Proc. AAAI'19.* AAAI Press, 2768–2775.

[10] Dov M. Gabbay and Hans Jürgen Ohlbach. 1992. Quantifier Elimination in Second-Order Predicate Logic. *South African Computer Journal* 7 (1992), 35–43.

[11] Dov M. Gabbay, Renate A. Schmidt, and Andrzej Szalas. 2008. *Second-Order Quantifier Elimination - Foundations, Computational Aspects and Applications.* Studies in logic : Mathematical logic and foundations, Vol. 12. College Publications.

[12] William Gatens, Boris Konev, and Frank Wolter. 2014. Lower and Upper Approximations for Depleting Modules of Description Logic Ontologies. In *Proc. ECAI'14 (Frontiers in Artificial Intelligence and Applications, Vol. 263).* IOS Press, 345–350.

[13] Silvio Ghilardi, Carsten Lutz, and Frank Wolter. 2006. Did I Damage My Ontology? A Case for Conservative Extensions in Description Logics. In *Proc. KR'06.* AAAI Press, 187–197.

[14] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. 2008. Modular Reuse of Ontologies: Theory and Practice. *J. Artif. Intell. Res.* 31 (2008), 273–318.

[15] Bernardo Cuenca Grau and Egor V. Kostylev. 2016. Logical Foundations of Privacy-Preserving Publishing of Linked Data. In *Proc. AAAI'16.* AAAI Press, 943–949.

[16] B. C. Grau and B. Motik. 2012. Reasoning over Ontologies with Hidden Content: The Import-by-Query Approach. *J. Artif. Intell. Res.* 45 (2012), 197–255.

[17] Nicola Guarino, Daniel Oberle, and Steffen Staab. 2009. What Is an Ontology? In *Handbook on Ontologies.* Springer, 1–17.

[18] Andreas Herzig and Jérôme Mengin. 2008. Uniform Interpolation by Resolution in Modal Logic. In *Proc. JELIA'08 (LNCS, Vol. 5293).* Springer, 219–231.

[19] Michel C. A. Klein and Dieter Fensel. 2001. Ontology versioning on the Semantic Web. In *Proc. SWWS'01.* 75–91.

[20] Michel C. A. Klein, Dieter Fensel, Atanas Kiryakov, and Damyan Ognyanov. 2002. Ontology Versioning and Change Detection on the Web. In *Proc. EKAW'02 (LNCS, Vol. 2473).* Springer, 197–212.

[21] Boris Konev, Carsten Lutz, Dirk Walther, and Frank Wolter. 2013. Model-theoretic inseparability and modularity of description logic ontologies. *Artif. Intell.* 203 (2013), 66–103.

[22] Boris Konev, Dirk Walther, and Frank Wolter. 2008. The Logical Difference Problem for Description Logic Terminologies. In *Proc. IJCAR'14 (LNCS, Vol. 5195).* Springer, 259–274.

[23] Boris Konev, Dirk Walther, and Frank Wolter. 2009. Forgetting and Uniform Interpolation in Large-Scale Description Logic Terminologies. In *Proc. IJCAI'09.* IJCAI/AAAI Press, 830–835.

[24] Roman Kontchakov, Frank Wolter, and Michael Zakharyaschev. 2010. Logic-based ontology comparison and module extraction, with an application to DL-Lite. *Artif. Intell.* 174, 15 (2010), 1093–1141.

[25] Patrick Koopmann. 2015. *Practical Uniform Interpolation for Expressive Description Logics.* Ph. D. Dissertation. The University of Manchester, UK.

[26] Patrick Koopmann, Warren Del-Pinto, Sophie Tourret, and Renate A. Schmidt. 2020. Signature-Based Abduction for Expressive Description Logics. In *Proc. KR'20.* 592–602.

[27] Patrick Koopmann and Renate A. Schmidt. 2013. Uniform Interpolation of $\mathcal{ALC}$-Ontologies Using Fixpoints. In *Proc. FroCos'13 (LNCS, Vol. 8152).* Springer, 87–102.

[28] Patrick Koopmann and Renate A. Schmidt. 2015. Saturated-Based Forgetting in the Description Logic $\mathcal{SIF}$. In *Proc. DL'15 (CEUR Workshop Proc., Vol. 1350).*

[29] Patrick Lambrix and He Tan. 2008. Ontology Alignment and Merging. In *Anatomy Ontologies for Bioinformatics, Principles and Practice.* Computational Biology, Vol. 6. Springer, 133–149.

[30] Fangzhen Lin and Ray Reiter. 1994. Forget It!. In *Proc. AAAI Fall Symposium on Relevance.* AAAI Press, 154–159.

[31] Michel Ludwig and Boris Konev. 2014. Practical Uniform Interpolation and Forgetting for $\mathcal{ALC}$ TBoxes with Applications to Logical Difference. In *Proc. KR'14.* AAAI Press, 318–327.

[32] Carsten Lutz, Inanç Seylan, and Frank Wolter. 2012. An Automata-Theoretic Approach to Uniform Interpolation and Approximation in the Description Logic EL. In *Proc. KR'12.* AAAI Press, 286–296.

[33] Carsten Lutz, Dirk Walther, and Frank Wolter. 2007. Conservative Extensions in Expressive Description Logics. In *Proc. IJCAI'07.* AAAI/IJCAI Press, 453–458.

[34] Carsten Lutz and Frank Wolter. 2010. Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. *J. Symb. Comput.* 45, 2 (2010), 194–228.

[35] Carsten Lutz and Frank Wolter. 2011. Foundations for Uniform Interpolation and Forgetting in Expressive Description Logics. In *Proc. IJCAI'11.* IJCAI/AAAI Press, 989–995.

[36] Nadeschda Nikitina and Sebastian Rudolph. 2014. (Non-)Succinctness of uniform interpolants of general terminologies in the description logic $\mathcal{EL}$. *Artif. Intell.* 215 (2014), 120–140.

[37] Nadeschda Nikitina, Sebastian Rudolph, and Birte Glimm. 2011. Reasoning-Supported Interactive Revision of Knowledge Bases. In *Proc. IJCAI'11.* IJCAI/AAAI, 1027–1032.

[38] Riku Nortje, Katarina Britz, and Thomas Meyer. 2013. Reachability Modules for the Description Logic $\mathcal{SRIQ}$. In *Proc. LPAR'19'.* 636–652.

[39] Natalya Fridman Noy and Mark A. Musen. 2000. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proc. AAAI/IAAI'00.* AAAI Press/The MIT Press, 450–455.

[40] Koopmann P. and Jieying Chen. 2020. Deductive Module Extraction for Expressive Description Logics. In *IJCAI'20.* ijcai.org, 1636–1643.

[41] Márcio Moretto Ribeiro and Renata Wassermann. 2009. Base Revision for Ontology Debugging. *J. Log. Comput.* 19, 5 (2009), 721–743.

[42] John Alan Robinson. 1965. A Machine-Oriented Logic Based on the Resolution Principle. *J. ACM* 12, 1 (1965), 23–41.

[43] Ana Armas Romero, Mark Kaminski, Bernardo Cuenca Grau, and Ian Horrocks. 2016. Module Extraction in Expressive Ontology Languages via Datalog Reasoning. *J. Artif. Intell. Res.* 55 (2016), 499–564.

[44] Renate A. Schmidt. 2012. The Ackermann approach for modal logic, correspondence theory and second-order reduction. *J. Appl. Log.* 10, 1 (2012), 52–74.

[45] Dan Schrimpsher, Zhiqiang Wu, Anthony M. Orme, and Letha H. Etzkorn. 2010. Dynamic ontology version control. In *Proc. ACMse'10.* ACM, 25.

[46] Andrzej Szalas. 1993. On the Correspondence between Modal and Classical Logic: An Automated Approach. *J. Log. Comput.* 3, 6 (1993), 605–620.

[47] Stephan Tobies. 2001. *Complexity results and practical algorithms for logics in knowledge representation.* Ph. D. Dissertation. RWTH Aachen University, Germany.

[48] Nicolas Troquard, Roberto Confalonieri, Pietro Galliani, Rafael Peñaloza, Daniele Porello, and Oliver Kutz. 2018. Repairing Ontologies via Axiom Weakening. In *Proc. AAAI'18.* AAAI Press, 1981–1988.

[49] Chiara Del Vescovo, Matthew Horridge, Bijan Parsia, Uli Sattler, Thomas Schneider, and Haoruo Zhao. 2020. Modular Structures and Atomic Decomposition in Ontologies. *J. Artif. Intell. Res.* 69 (2020), 963–1021.

[50] Albert Visser. 1996. *Bisimulations, Model Descriptions and Propositional Quantifiers.* Utrecht University.

[51] Kewen Wang, Grigoris Antoniou, Rodney Topor, and Abdul Sattar. 2005. Merging and Aligning Ontologies in dl-Programs. In *Proc. RuleML'05 (LNCS, Vol. 3791).* Springer, 160–171.

[52] Kewen Wang, Zhe Wang, Rodney W. Topor, Jeff Z. Pan, and Grigoris Antoniou. 2014. Eliminating Concepts and Roles from Ontologies in Expressive Descriptive Logics. *Computational Intelligence* 30, 2 (2014), 205–232.

[53] Yue Xiang, Xuan Wu, Chang Lu, and Yizheng Zhao. 2022. Creating Signature-Based Views for Description Logic Ontologies with Transitivity and Qualified Number Restrictions. In *Proc. WWW'22.* ACM, 808–817.

[54] Yan Zhang and Yi Zhou. 2010. Forgetting Revisited. In *KR.* AAAI Press.

[55] Yizheng Zhao, Ghadah Alghamdi, Renate A. Schmidt, Hao Feng, Giorgos Stoilos, Damir Juric, and Mohammad Khodadadi. 2019. Tracking Logical Difference in Large-Scale Ontologies: A Forgetting-Based Approach. In *Proc. AAAI'19.* AAAI Press, 3116–3124.

[56] Yizheng Zhao and Renate A. Schmidt. 2016. Forgetting Concept and Role Symbols in $\mathcal{ALCOIH}\mu^{+}(\nabla,\sqcap)$-Ontologies. In *Proc. IJCAI'16.* IJCAI Press, 1345–1353.

[57] Yizheng Zhao and Renate A. Schmidt. 2018. FAME: An Automated Tool for Semantic Forgetting in Expressive Description Logics. In *Proc. IJCAR'18 (LNCS, Vol. 10900).* Springer, 19–27.

[58] Yizheng Zhao and Renate A. Schmidt. 2018. On Concept Forgetting in Description Logics with Qualified Number Restrictions. In *Proc. IJCAI'18.* IJCAI/AAAI Press, 1984–1990.

A APPENDIX

A.1 Missing Proofs

We define the *frequency* $\mathsf{fq}(\mathsf{A}, C)$ of A in an A-concept C inductively as follows:

- $\mathsf{fq}(\mathsf{A}, X) = \begin{cases} 1, & \text{if } X = \mathsf{A}; \\ 0, & \text{if } X \in \mathsf{N_C} \text{ and } X \neq \mathsf{A}, \end{cases}$
- $\mathsf{fq}(\mathsf{A}, \exists R.E) = \mathsf{fq}(\mathsf{A}, E)$,
- $\mathsf{fq}(\mathsf{A}, E \sqcap F) = \mathsf{fq}(\mathsf{A}, E) + \mathsf{fq}(\mathsf{A}, F)$.

We define the *frequency* $\mathsf{fq}(\mathsf{A}, X \sqsubseteq Y)$ of A in a GCI $X \sqsubseteq Y$:

- $\mathsf{fq}(\mathsf{A}, X \sqsubseteq Y) = \mathsf{fq}(\mathsf{A}, X) + \mathsf{fq}(\mathsf{A}, Y)$.

Let A^* be a designated occurrence of A in C. We define the *role depth* $\mathsf{dp}(\mathsf{A}^*, C)$ of A^* in C inductively as follows:

- $\mathsf{dp}(\mathsf{A}^*, C) = 0$, if C is of the form $\mathsf{A}^* \sqcap D$, where D is an arbitrary concept,
- $\mathsf{dp}(\mathsf{A}^*, C) = \mathsf{dp}(\mathsf{A}^*, E) + 1$, if C is of the form $D \sqcap \exists R.E$, where $r \in \mathsf{N_R}$, E is a concept that contains A^*, and D is an arbitrary concept,

i.e., $\mathsf{dp}(\mathsf{A}^*, C)$ counts the number of $\exists$-restrictions guarding A^* in C. The *role depth* $\mathsf{dp}(\mathsf{A}, C)$ of A in C is defined as the sum of the *role depth* of all occurrences of A in C. The *role depth* $\mathsf{dp}(\mathsf{A}, X \sqsubseteq Y)$ of A in a GCI $X \sqsubseteq Y$ is defined as:

- $\mathsf{dp}(\mathsf{A}, X \sqsubseteq Y) = \mathsf{dp}(\mathsf{A}, X) + \mathsf{dp}(\mathsf{A}, Y)$.

Proposition 1. *Any* A-*GCI* $X \sqsubseteq Y$ *with* $\mathsf{dp}(\mathsf{A}, X \sqsubseteq Y) = 0$ *is in* A-*normal form.*

Proof. $\mathsf{dp}(\mathsf{A}, X \sqsubseteq Y) = 0$ indicates that A is not guarded by any $\exists$-restriction in $X \sqsubseteq Y$, ensuring that A appears only at the surface level of the concepts X and Y. This leads to three possible cases (a simpler representation of $X \sqsubseteq Y$ is assumed):

- $\mathsf{A} \notin \mathsf{sig_C}(X)$ and $\mathsf{A} \in \mathsf{sig_C}(Y)$ (Form I)
- $\mathsf{A} \in \mathsf{sig_C}(X)$ and $\mathsf{A} \notin \mathsf{sig_C}(Y)$ (Form III)
- $\mathsf{A} \in \mathsf{sig_C}(X)$ and $\mathsf{A} \in \mathsf{sig_C}(Y)$

The last case can be generalized as $\mathsf{A} \sqcap C \sqsubseteq \mathsf{A} \sqcap D$, with C, D being A-free concepts. This is equivalent to $\mathsf{A} \sqcap C \sqsubseteq \mathsf{A}$ and $\mathsf{A} \sqcap C \sqsubseteq D$; the former is a tautology and the latter aligns with Form III. □

Proposition 2. *Any* A-*GCI* $X \sqsubseteq Y$ *with* $\mathsf{fq}(\mathsf{A}, X \sqsubseteq Y) = 1$ *and* $\mathsf{dp}(\mathsf{A}, X \sqsubseteq Y) = 1$ *is in* A-*normal form.*

Proof. $\mathsf{fq}(\mathsf{A}, X \sqsubseteq Y) = 1$ ensures a single occurrence of A in $X \sqsubseteq Y$ and $\mathsf{dp}(\mathsf{A}, X \sqsubseteq Y) = 1$ indicates that A is guarded by a single $\exists$-restriction. This leads to two possible cases:

- $\mathsf{A} \notin \mathsf{sig_C}(X)$ and $\mathsf{A} \in \mathsf{sig_C}(Y)$ (Form II)
- $\mathsf{A} \in \mathsf{sig_C}(X)$ and $\mathsf{A} \notin \mathsf{sig_C}(Y)$ (Form IV)

Both cases are already in A-normal form. □

Proposition 3. *For any* A-*GCI* $X \sqsubseteq Y$, $\mathsf{fq}(\mathsf{A}, X \sqsubseteq Y) - \mathsf{dp}(\mathsf{A}, X \sqsubseteq Y) \leq 1$.

Proof. For any A-GCI $X \sqsubseteq Y$ with a depth $\mathsf{dp}(\mathsf{A}, X \sqsubseteq Y) = m$ ($m \geq 0$), A appears in $X \sqsubseteq Y$ at most m+1 times. As per Proposition 1, A can occur unguarded (not within an $\exists$-restriction) only once in $X \sqsubseteq Y$. Given A's depth of m in $X \sqsubseteq Y$, there can be at most m guarded occurrences of A. Thus, A appears in $X \sqsubseteq Y$ a maximum of $m + 1$ times. □

We define $\mathsf{NLZ}_1(\mathcal{O})$ as the set derived from $\mathcal{O}$ by applying one of the rules NR1 – NR5 to $\mathcal{O}$, indicating one round of normalization for $\mathcal{O}$. Similarly, $\mathsf{NLZ}_k(\mathcal{O})$ is defined for any $k \geq 0$, with $\mathsf{NLZ}_0(\mathcal{O})$ representing the original $\mathcal{O}$. We further define $\mathsf{dp}(\mathsf{A}, \mathcal{O})$ as the sum of $\mathsf{dp}(\mathsf{A}, X \sqsubseteq Y)$, for every A-GCI $X \sqsubseteq Y$ in $\mathcal{O}$.

Proposition 4. $\mathsf{dp}(\mathsf{A}, \mathsf{NLZ}_{k-1}(\mathcal{O})) - \mathsf{dp}(\mathsf{A}, \mathsf{NLZ}_k(\mathcal{O})) = 1$, *where (i)* $k \geq 1$ *and (ii)* $\mathsf{NLZ}_{k-1}(\mathcal{O})$ *is not in* A-*normal form.*

Proof. Condition (ii) ensures that $\mathsf{NLZ}_{k-1}(\mathcal{O}) \neq \mathsf{NLZ}_k(\mathcal{O})$ and activates the applicability of NR2 – NR5. We prove that applying any of these rules decreases the depth of A in the resulting $\mathcal{O}$ by 1. We treat NR2 in detail. The rules NR3 – NR5 can be proved similarly. Consider $\mathsf{NLZ}_{k-1}(\mathcal{O})$ as a set with an A-GCI $X \sqsubseteq Y$ not in A-normal form, where $\mathsf{dp}(\mathsf{A}, X \sqsubseteq Y) = m$ and $\mathsf{dp}(\mathsf{A}, \mathsf{NLZ}_{k-1}(\mathcal{O})) = m + x$ for $x \geq 0$. Rule NR2 implies that $m \geq 1$. Assume $\mathsf{dp}(\mathsf{A}, C) = n$, and Rule NR2 indicates $n \geq 1$. Consequently, $\mathsf{dp}(\mathsf{A}, \exists r.C) = n + 1$ and $m = n + 1 + y$ ($y \geq 0$). Replacing C with a new definer Z eliminates $X \sqsubseteq Y$ from $\mathsf{NLZ}_k(\mathcal{O})$ and introduces two GCIs: $(X \sqsubseteq Y)^C_Z$ and $C \sqsubseteq Z$. Here, $(X \sqsubseteq Y)^C_Z$ denotes the GCI obtained from $X \sqsubseteq Y$ by replacing C with Z.

Since $\mathsf{dp}(\mathsf{A}, (X \sqsubseteq Y)^C_Z) = \mathsf{dp}(\mathsf{A}, (X \sqsubseteq Y)) - \mathsf{dp}(\mathsf{A}, \exists r.C) = n + 1 + y - (n + 1) = y$ and $\mathsf{dp}(\mathsf{A}, C \sqsubseteq Z) = \mathsf{dp}(\mathsf{A}, C) + \mathsf{dp}(\mathsf{A}, Z) = n$, $\mathsf{dp}(\mathsf{A}, \mathsf{NLZ}_k(\mathcal{O})) = \mathsf{dp}(\mathsf{A}, \mathsf{NLZ}_{k-1}(\mathcal{O})) - \mathsf{dp}(\mathsf{A}, X \sqsubseteq Y) + y + n = x + y + n$. Given that $\mathsf{dp}(\mathsf{A}, \mathsf{NLZ}_{k-1}(\mathcal{O})) = m + x = x + y + n + 1$, $\mathsf{dp}(\mathsf{A}, \mathsf{NLZ}_{k-1}(\mathcal{O})) - \mathsf{dp}(\mathsf{A}, \mathsf{NLZ}_k(\mathcal{O})) = 1$. □

Lemma 1. *Let* $\mathcal{O}$ *be an arbitrary* $\mathcal{ELIO}$*-ontology. Then* $\mathcal{O}$ *can be transformed into A-normal form* $\mathcal{O}'$ *by a linear number of applications of the above normalization rules NR1 – NR5. In addition, the size of the resulting ontology* $\mathcal{O}'$ *is linear in the size of* $\mathcal{O}$.

Proof. The A-normal form ensures that every A-GCI contains exactly one occurrence of A; it covers all elementary structures of an A-GCI and specifies where the A appears in these structures. Specifically, an A-concept is either the atomic concept A (the base case) or is constructed with $\exists$ and $\sqcap$ (the induction cases). In each case, A appears as a conjunct in an explicit or implicit conjunction. For example, the atomic concept A is a simpler representation of $\mathsf{A} \sqcap D$, with $D \equiv \top$. Since $\mathsf{A} \sqcap D$ appears under $\exists$-restrictions or without any, we identify two basic forms of an A-concept: $\mathsf{A} \sqcap D$ and $\exists \mathrm{r}.(\mathsf{A} \sqcap D) \sqcap F$. These can be on either side of a GCI. Form I and III generalize the scenarios where $\mathsf{A} \sqcap D$ is on the right and left sides of a GCI, respectively, while Form II (IV) covers cases with $\exists \mathrm{r}.(\mathsf{A} \sqcap D) \sqcap F$ on the right and left sides.

Proposition 4 states that any unnormalized A-GCI $X \sqsubseteq Y$ can be reduced to a set $\mathsf{NLZ}(X \sqsubseteq Y)$ of GCIs in finite steps with each GCI α in $\mathsf{NLZ}(X \sqsubseteq Y)$ having $\mathsf{dp}(\mathsf{A}, \alpha) \leq 1$. If $\mathsf{dp}(\mathsf{A}, \alpha) = 0$, α is either a non-A-GCI, or in A-normal form (as per Proposition 1). If $\mathsf{dp}(\mathsf{A}, \alpha) = 1$, α may not be in A-normal form because it might contain two occurrences of A (as per Proposition 3); for instance, $\mathsf{dp}(\mathsf{A}, \mathsf{A} \sqcap \exists r.\mathsf{A} \sqsubseteq Y) = 1$ but $\mathsf{fq}(\mathsf{A}, \mathsf{A} \sqcap \exists r.\mathsf{A} \sqsubseteq Y) = 2$. These cases, with an unguarded A and an A guarded by a single $\exists$-restriction, meet NR2 – NR3's syntactic requirements, and can be normalized in one step using NR2 – NR3. Since every definer replaces a subconcept immediately under an $\exists$-restriction, the number of the definers for $\mathcal{O}$'s normalization and the newly-added GCIs is bounded by $\mathrm{O}(n)$, where n is the count of $\exists$-restrictions in $\mathcal{O}$. This proves termination and completeness of the normalization procedure. □

Definition 5 (Conservative Extension). *Given two general $\mathcal{ELIO}$-ontologies $\mathcal{O}$ and $\mathcal{O}'$, we say that $\mathcal{O}$ is a* conservative extension *of $\mathcal{O}'$ if*

(1) $sig(\mathcal{O}') \subseteq sig(\mathcal{O})$,
(2) every model of $\mathcal{O}$ is a model of $\mathcal{O}'$, and
(3) for every model $\mathcal{I}'$ of $\mathcal{O}'$, there exists a model $\mathcal{I}$ of $\mathcal{O}$ such that the extensions of concept and role names from $sig(\mathcal{O}')$ coincide in $\mathcal{I}$ and $\mathcal{I}'$, i.e.,
  - *$A^{\mathcal{I}} = A^{\mathcal{I}'}$ for all concept names $A \in sig(\mathcal{O}')$, and*
  - *$r^{\mathcal{I}} = r^{\mathcal{I}'}$ for all role names $r \in sig(\mathcal{O}')$.*

It is easy to see that the notion of a conservative extension is *transitive*, i.e., if $\mathcal{O}$ is a conservative extension of $\mathcal{O}'$ and $\mathcal{O}'$ is a conservative extension of $\mathcal{O}''$, then $\mathcal{O}$ is a conservative extension of $\mathcal{O}''$. In addition, the notion of conservative extension preserves *subsumption* in the following sense: if $\mathcal{O}$ is a conservative extension of $\mathcal{O}'$, then subsumption w.r.t. $\mathcal{O}'$ coincides with subsumption w.r.t. $\mathcal{O}$ for all GCIs built using only names from $\mathrm{sig}(\mathcal{O}')$.

Proposition 5. *Let $\mathcal{O}$ and $\mathcal{O}'$ be general $\mathcal{ELIO}$ ontologies such that $\mathcal{O}$ is a conservative extension of $\mathcal{O}'$, and C, D are $\mathcal{ELIO}$ concepts containing only concept and role names from $\mathcal{O}'$. Then we have*

$$\mathcal{O}' \models C \sqsubseteq D \text{ iff } \mathcal{O} \models C \sqsubseteq D.$$

Proof. We prove this by contraposition. First, assume that $\mathcal{O} \not\models C \sqsubseteq D$. Then there is a model $\mathcal{I}$ of $\mathcal{O}$ such that $C^{\mathcal{I}} \not\subseteq D^{\mathcal{I}}$. Since $\mathcal{I}$ is also a model of $\mathcal{O}'$, this implies $\mathcal{O}' \not\models C \sqsubseteq D$. Next, assume that $\mathcal{O}' \not\models C \sqsubseteq D$. Then there is a model $\mathcal{I}'$ of $\mathcal{O}'$ such that $C^{\mathcal{I}'} \not\subseteq D^{\mathcal{I}'}$. Let $\mathcal{I}$ be a model of $\mathcal{O}$ such that the extensions of concept and role names from $\mathrm{sig}(\mathcal{O}')$ coincide in $\mathcal{I}$ and $\mathcal{I}'$. Since C and D contain only concept and role names from $\mathrm{sig}(\mathcal{O}')$, we have $C^{\mathcal{I}} = C^{\mathcal{I}'} \not\subseteq D^{\mathcal{I}'} = D^{\mathcal{I}}$, and $\mathcal{O} \not\models C \sqsubseteq D$. □

Lemma 2. *Let $\mathcal{O}$ be a general $\mathcal{ELIO}$-ontology and $\mathcal{O}'$ the normalized one obtained from $\mathcal{O}$ using the normalization rules NR1 – NR5. Then we have*

$$\mathcal{O} \models C \sqsubseteq D \text{ iff } \mathcal{O}' \models C \sqsubseteq D,$$

for any $\mathcal{ELIO}$-GCI $C \sqsubseteq D$ with $sig(C \sqsubseteq D) \subseteq sig(\mathcal{O})$.

Proof. Our proof is based on the notion of conservative extension defined above [13, 33, 34]. Specifically, we show that the ontology $\mathcal{O}'$ obtained from $\mathcal{O}$ by applying one of the normalization rules is a conservative extension of $\mathcal{O}$. We treat NR2 in detail. The rules NR3, NR4 and NR5 can be proved similarly. This statement holds trivially for NR1 since in that case $\mathcal{O}$ and $\mathcal{O}'$ have the same signature and are obviously equivalent.

Regarding NR2, assume that $\mathcal{O}'$ is obtained from $\mathcal{O}$ by replacing the GCI $X \sqsubseteq Y$ (NR2 generalizes the cases where $X = \exists R.C \sqcap D$) with two GCIs $\exists R.Z \sqcap D \sqsubseteq Y$ and $C \sqsubseteq Z$, where $Z \in \mathsf{N_C}$ is a fresh definer, i.e., $Z \notin \mathrm{sig}(\mathcal{O})$. Obviously, $\mathrm{sig}(\mathcal{O}') = \mathrm{sig}(\mathcal{O}) \cup \{Z\}$, and therefore $\mathrm{sig}(\mathcal{O}) \subseteq \mathrm{sig}(\mathcal{O}')$, satisfying Condition (1) of Definition 5. Next, assume that $\mathcal{I}'$ is a model of $\mathcal{O}'$. Then we have $(\exists R.Z)^{\mathcal{I}'} \cap D^{\mathcal{I}'} \subseteq Y^{\mathcal{I}'}$ and $C^{\mathcal{I}'} \subseteq Z^{\mathcal{I}'}$. This implies $(\exists R.C)^{\mathcal{I}'} \cap D^{\mathcal{I}'} \subseteq (\exists R.Z)^{\mathcal{I}'} \cap D^{\mathcal{I}'} \subseteq Y^{\mathcal{I}'}$, and thus $\mathcal{I}'$ is a model of $\mathcal{O}$. Finally, assume that $\mathcal{I}$ is a model of $\mathcal{O}$. Let $\mathcal{I}'$ be the interpretation that coincides with $\mathcal{I}$ on all concept and role names with the exception of Z. For Z, we define the extension in $\mathcal{I}'$ as $Z^{\mathcal{I}'} = C^{\mathcal{I}}$. Since $\mathcal{I}$ is a model of $\mathcal{O}$, we have $(\exists R.C)^{\mathcal{I}} \cap D^{\mathcal{I}} \subseteq Y^{\mathcal{I}}$. In addition, since Z does not occur in $\exists R.C$, D, or Y, we have $C^{\mathcal{I}} = C^{\mathcal{I}'}$, $(\exists R.C)^{\mathcal{I}} = (\exists R.C)^{\mathcal{I}'}$, $D^{\mathcal{I}} = D^{\mathcal{I}'}$ and $Y^{\mathcal{I}} = Y^{\mathcal{I}'}$. This yields $C^{\mathcal{I}'} = C^{\mathcal{I}} = Z^{\mathcal{I}'}$ and $(\exists R.Z)^{\mathcal{I}'} \cap D^{\mathcal{I}'} = (\exists R.C)^{\mathcal{I}} \cap D^{\mathcal{I}} \subseteq Y^{\mathcal{I}} = Y^{\mathcal{I}'}$, showing that $\mathcal{I}'$ is a model of $\mathcal{O}'$. Because of transitivity, Lemma 2 is an immediate consequence of Proposition 5. □

Lemma 3. *Let $\mathcal{O}$ be an $\mathcal{ELIO}$-ontology in A-NF, and $\mathcal{O}^{-A}$ an ontology obtained from forgetting $\{\mathsf{A}\}$ from $\mathcal{O}$ using the inference rules in Figure 2, then we have:*

$$\mathcal{O} \models C \sqsubseteq D \text{ iff } \mathcal{O}^{-A} \models C \sqsubseteq D,$$

for any $\mathcal{ELIO}$-GCI $C \sqsubseteq D$ with $sig(C \sqsubseteq D) \subseteq sig(\mathcal{O}) \backslash \{A\}$.

Proof. The rules $\mathsf{IR1}$ and $\mathsf{IR2}$ essentially reverse the normalization process, thereby preserving subsumption directly. We then focus on proving Rule $\mathsf{IR3}$. Rule $\mathsf{IR3}$ can be proved similarly. We show that the GCIs on the left side (the *premises*, denoted by $\mathcal{O}$) of the $\Longrightarrow$ symbol are a conservative extension of those on the right side (the *conclusion*, denoted by $\mathcal{O}'$). Obviously, $\mathrm{sig}(\mathcal{O}) = \mathrm{sig}(\mathcal{O}') \cup \{\mathsf{A}\}$, and therefore $\mathrm{sig}(\mathcal{O}') \subseteq \mathrm{sig}(\mathcal{O})$, satisfying Condition (1) of Definition 2.
Case I (i.e., provided that: $\mathcal{O} \models \mathsf{A} \sqcap D \sqsubseteq E_1 \sqcap \ldots \sqcap E_n$):
Assume that $\mathcal{I}$ is a model of $\mathcal{O}$. Then we have the following:

$$C^{\mathcal{I}} \subseteq (\exists R.(\mathsf{A} \sqcap D))^{\mathcal{I}} \tag{1}$$

$$(\mathsf{A} \sqcap E_1)^{\mathcal{I}} \subseteq {G_1}^{\mathcal{I}}, \ldots, (\mathsf{A} \sqcap E_n)^{\mathcal{I}} \subseteq {G_n}^{\mathcal{I}} \tag{2}$$

$$(\mathsf{A} \sqcap D)^{\mathcal{I}} \subseteq (E_1 \sqcap \ldots \sqcap E_n)^{\mathcal{I}} \tag{3}$$

As Inclusion (2) implies $(\mathsf{A} \sqcap E_1)^{\mathcal{I}} \cap \ldots \cap (\mathsf{A} \sqcap E_n)^{\mathcal{I}} \subseteq {G_1}^{\mathcal{I}} \cap \ldots \cap {G_n}^{\mathcal{I}}$, we have $(\mathsf{A} \sqcap E_1 \sqcap \ldots \sqcap E_n)^{\mathcal{I}} \subseteq (G_1 \sqcap \ldots \sqcap G_n)^{\mathcal{I}}$ and further $(\mathsf{A} \sqcap E_1 \sqcap \ldots \sqcap E_n)^{\mathcal{I}} \subseteq (E_1 \sqcap \ldots \sqcap E_n \sqcap G_1 \sqcap \ldots \sqcap G_n)^{\mathcal{I}}$. As Inclusion (3) implies $(\mathsf{A} \sqcap D)^{\mathcal{I}} \subseteq (\mathsf{A} \sqcap E_1 \sqcap \ldots \sqcap E_n)^{\mathcal{I}}$, we have $(\mathsf{A} \sqcap D)^{\mathcal{I}} \subseteq (E_1 \sqcap \ldots \sqcap E_n \sqcap G_1 \sqcap \ldots \sqcap G_n)^{\mathcal{I}}$ and further $(\mathsf{A} \sqcap D)^{\mathcal{I}} \subseteq (E_1 \sqcap \ldots \sqcap E_n \sqcap G_1 \sqcap \ldots \sqcap G_n \sqcap D)^{\mathcal{I}}$. Together with Inclusion (1), we have the following:

$$C^{\mathcal{I}} \subseteq (\exists R.(E_1 \sqcap \ldots \sqcap E_n \sqcap G_1 \sqcap \ldots \sqcap G_n \sqcap D))^{\mathcal{I}} \tag{4}$$

This satisfies Condition (2) of Definition 2.

Assume that $\mathcal{I}'$ is a model of $\mathcal{O}'$. Let $\mathcal{I}$ be the interpretation that coincides with $\mathcal{I}'$ on all concept and role names with the exception of A. For A, we define the extension in $\mathcal{I}$ as $\mathsf{A}^{\mathcal{I}} = (E_1 \sqcap \ldots \sqcap E_n \sqcap G_1 \sqcap \ldots \sqcap G_n)^{\mathcal{I}'}$. Since $\mathcal{I}'$ is a model of $\mathcal{O}'$, we have $C^{\mathcal{I}'} \subseteq (\exists R.(E_1 \sqcap \ldots \sqcap E_n \sqcap G_1 \sqcap \ldots \sqcap G_n \sqcap D))^{\mathcal{I}'}$. In addition, since A does not occur in C or $\exists r.(E_1 \sqcap \ldots \sqcap E_n \sqcap G_1 \sqcap \ldots \sqcap G_n \sqcap D)$, we have $C^{\mathcal{I}'} = C^{\mathcal{I}}$, $(E_1 \sqcap \ldots \sqcap E_n \sqcap G_1 \sqcap \ldots \sqcap G_n)^{\mathcal{I}'} = (E_1 \sqcap \ldots \sqcap E_n \sqcap G_1 \sqcap \ldots \sqcap G_n)^{\mathcal{I}} = \mathsf{A}^{\mathcal{I}}$, $(\exists R.(E_1 \sqcap \ldots \sqcap E_n \sqcap G_1 \sqcap \ldots \sqcap G_n \sqcap D))^{\mathcal{I}'} = (\exists R.(E_1 \sqcap \ldots \sqcap E_n \sqcap G_1 \sqcap \ldots \sqcap G_n \sqcap D))^{\mathcal{I}} = (\exists R.(\mathsf{A} \sqcap D))^{\mathcal{I}}$, and thus, we have

$$C^{\mathcal{I}} \subseteq (\exists R.(\mathsf{A} \sqcap D))^{\mathcal{I}} \tag{5}$$

Since $(E_1 \sqcap \ldots \sqcap E_n \sqcap G_1 \sqcap \ldots \sqcap G_n \sqcap E_1)^{\mathcal{I}} \subseteq {G_1}^{\mathcal{I}}$, we have

$$(\mathsf{A} \sqcap E_1)^{\mathcal{I}} \subseteq {G_1}^{\mathcal{I}} \tag{6}$$

In the same way, we can prove the other GCIs in (2). Since $(E_1 \sqcap \ldots \sqcap E_n \sqcap G_1 \sqcap \ldots \sqcap G_n \sqcap D)^{\mathcal{I}} \subseteq (E_1 \sqcap \ldots \sqcap E_n)^{\mathcal{I}}$, we have

$$(\mathsf{A} \sqcap D)^{\mathcal{I}} \subseteq (E_1 \sqcap \ldots \sqcap E_n)^{\mathcal{I}} \tag{7}$$

Inclusions (5), (6), and (7) collectively satisfy Condition (3) of Definition 2.

Case II (i.e., provided that: $\mathcal{O} \not\models \mathsf{A} \sqcap D \sqsubseteq E$):

Assume that $\mathcal{I}$ is a model of $\mathcal{O}$. Then we have the following:

$$C^{\mathcal{I}} \subseteq (\exists R.(\mathsf{A} \sqcap D))^{\mathcal{I}} \tag{8}$$

$$(\mathsf{A} \sqcap E)^{\mathcal{I}} \subseteq G^{\mathcal{I}} \tag{9}$$

Directly, we have

$$C^{\mathcal{I}} \subseteq (\exists R.D)^{\mathcal{I}} \tag{10}$$

Assume that $\mathcal{I}'$ is a model of $\mathcal{O}'$. Let $\mathcal{I}$ be the interpretation that coincides with $\mathcal{I}'$ on all concept and role names with the exception of A, E and G. For A, we define the extension in $\mathcal{I}$ as $\mathsf{A}^{\mathcal{I}} = \top^{\mathcal{I}'}$. Since $\mathcal{I}'$ is a model of $\mathcal{O}'$, we have $C^{\mathcal{I}'} \subseteq (\exists R.D)^{\mathcal{I}'}$. In addition, since A does not occur in C or $\exists R.D$, we have $C^{\mathcal{I}'} = C^{\mathcal{I}}$, $\top^{\mathcal{I}'} = \top^{\mathcal{I}} = \mathsf{A}^{\mathcal{I}}$, $(\exists r.D)^{\mathcal{I}'} = (\exists R.(\top \sqcap D))^{\mathcal{I}'} = (\exists R.(\top \sqcap D))^{\mathcal{I}} = (\exists R.(\mathsf{A} \sqcap D))^{\mathcal{I}}$, and thus, we have

$$C^{\mathcal{I}} \subseteq (\mathsf{A} \sqcap \exists R.D)^{\mathcal{I}} \tag{11}$$

For E and G, we define their extensions in $\mathcal{I}$ as $E^{\mathcal{I}} = G^{\mathcal{I}} = \top^{\mathcal{I}'}$. Then we have

$$(\mathsf{A} \sqcap E)^{\mathcal{I}} \subseteq G^{\mathcal{I}} \tag{12}$$

Thus, $\mathcal{O}$ is a conservative extension of $\mathcal{O}'$. Because of the transitivity of conservative extension, $\mathcal{O}$ in Lemma 3 is a conservative extension of $\mathcal{O}^{-\mathsf{A}}$. According to Proposition 5, Lemma 3 holds. □

Likewise, Lemma 4 establishes the partial soundness of the calculus. Specifically, the derived ontology $\mathcal{O}^{-\mathsf{r}}$ fulfills the second condition necessary for it to be the result of forgetting $\{\mathsf{r}\}$ from $\mathcal{O}$. However, $\mathcal{O}^{-\mathsf{r}}$ may include definers which fall outside the scope of $\mathrm{sig}(\mathcal{O}) \backslash \{\mathsf{r}\}$, potentially failing to fulfill the first condition.

Lemma 4. *Let $\mathcal{O}$ be an $\mathcal{ELIO}$-ontology in r-NF, and $\mathcal{O}^{-r}$ an ontology obtained from forgetting $\{\mathsf{r}\}$ from $\mathcal{O}$ using the inference rules in Figure 3, then we have:*

$$\mathcal{O} \models C \sqsubseteq D \textit{ iff } \mathcal{O}^{-r} \models C \sqsubseteq D,$$

for any $\mathcal{ELIO}$-GCI $C \sqsubseteq D$ with $sig(C \sqsubseteq D) \subseteq sig(\mathcal{O}) \backslash \{r\}$.

Proof. We prove Rules IR5 – IR8, with a detailed examination of Rule IR5. The other rules can be proved similarly. Regarding IR5, we show that the GCIs on the left side (the *premises*, denoted by $\mathcal{O}$) of the $\Longrightarrow$ symbol are a conservative extension of those on the right side (the *conclusion*, denoted by $\mathcal{O}'$). Obviously, $\mathrm{sig}(\mathcal{O}) = \mathrm{sig}(\mathcal{O}') \cup \{\mathsf{r}\}$, and therefore $\mathrm{sig}(\mathcal{O}') \subseteq \mathrm{sig}(\mathcal{O})$, satisfying Condition (1) of Definition 2.

Assume that $\mathcal{I}$ is a model of $\mathcal{O}$. Then we have the following:

$$C^{\mathcal{I}} \subseteq (\exists \mathsf{r}.D)^{\mathcal{I}} \tag{13}$$

$$(F \sqcap \exists \mathsf{r}.E)^{\mathcal{I}} \subseteq G^{\mathcal{I}} \tag{14}$$

$$(\exists \mathsf{r}.D)^{\mathcal{I}} \subseteq (\exists \mathsf{r}.E)^{\mathcal{I}} \tag{15}$$

Inclusions (13) and (15) jointly imply $C^{\mathcal{I}} \subseteq (\exists \mathsf{r}.E)^{\mathcal{I}}$, and when combined with (14), they further imply $(F \sqcap C)^{\mathcal{I}} \subseteq G^{\mathcal{I}}$.

Assume that $\mathcal{I}'$ is a model of $\mathcal{O}'$. Let $\mathcal{I}$ be the interpretation that coincides with $\mathcal{I}'$ on all concept and role names with the exception of r, D, and E. For the exceptions, we define their extensions in $\mathcal{I}$ as $(\exists \mathsf{r}.D)^{\mathcal{I}} = (\exists \mathsf{r}.E)^{\mathcal{I}} = C^{\mathcal{I}'}$. Then we have $(\exists \mathsf{r}.D)^{\mathcal{I}} \subseteq (\exists \mathsf{r}.E)^{\mathcal{I}}$ Since $\mathcal{I}'$ is a model of $\mathcal{O}'$, we have $(F \sqcap C)^{\mathcal{I}'} \subseteq G^{\mathcal{I}'}$. In addition, since r does not occur in C, D, E, F, and G, we have $C^{\mathcal{I}'} = C^{\mathcal{I}}$, $F^{\mathcal{I}'} = F^{\mathcal{I}}$ and $G^{\mathcal{I}'} = G^{\mathcal{I}}$. Thus we have $C^{\mathcal{I}'} = C^{\mathcal{I}} \subseteq (\exists \mathsf{r}.D)^{\mathcal{I}}$ and $(F^{\mathcal{I}} \sqcap \exists \mathsf{r}.E)^{\mathcal{I}} \subseteq G^{\mathcal{I}}$

Thus, $\mathcal{O}$ is a conservative extension of $\mathcal{O}'$. Because of the transitivity of conservative extension, $\mathcal{O}$ in Lemma 4 is a conservative extension of $\mathcal{O}^{-\mathsf{r}}$. According to Proposition 5, Lemma 4 holds. □

Theorem 1. *Given any $\mathcal{ELIO}$-ontology $\mathcal{O}$ and any forgetting signature $\mathcal{F} \subseteq sig(\mathcal{O})$ as input, our forgetting method always terminates and returns an $\mathcal{ELIO}$-ontology $\mathcal{V}$. If $\mathcal{V}$ does not contain any definers, then it is a result of forgetting $\mathcal{F}$ from $\mathcal{O}$.*

Proof. Note that the normalization and inference rules do not introduce new cycles. For cases where cyclic behavior originally exhibits over the names in $\mathcal{F}$, the method terminates upon detecting a cycle. In acyclic cases, termination of the method follows from Lemma 1 and the termination of the forgetting calculi. The method's soundness is ensured by Lemma 2 and its counterpart lemma for r-normalization, along with Lemmas 3 and 4. □

A.2 Statistics of Oxford-ISG and BioPortal

Table 2: Statistics of Oxford-ISG and BioPortal

Oxford		min	max	med	mean	90 percentile
I	$\lvert N_C \rvert$	0	1582	86	191	545
	$\lvert N_R \rvert$	0	332	10	29	80
	|Onto|	10	990	162	262	658
II	$\lvert N_C \rvert$	200	5877	1665	1769	2801
	$\lvert N_R \rvert$	0	887	11	34	61
	|Onto|	1008	4976	2282	2416	3937
III	$\lvert N_C \rvert$	1162	9809	4042	5067	8758
	$\lvert N_R \rvert$	1	158	4	23	158
	|Onto|	5112	9783	7277	7195	9179
BioPortal		**min**	**max**	**med**	**mean**	**90 percentile**
I	$\lvert N_C \rvert$	0	784	127	192	214
	$\lvert N_R \rvert$	0	122	5	15	17
	|Onto|	10	794	283	312	346
II	$\lvert N_C \rvert$	5	4530	1185	1459	1591
	$\lvert N_R \rvert$	0	131	12	30	33
	|Onto|	1023	4880	2401	2619	2782
III	$\lvert N_C \rvert$	432	8340	4363	4387	4806
	$\lvert N_R \rvert$	0	135	17	30	34
	|Onto|	5457	8339	6934	6912	7109