Practical Uniform Interpolation for Efficient Computation of Signature-Restricted Modules of Web Ontologies

Anonymous Author(s)

Abstract

Modular reuse of ontologies provides a highly desirable strategy for Web-based ontological knowledge processing, but also presents unique challenges, particularly in privacy-sensitive contexts. This paper presents a practical method for computing signature-restricted modules of Web ontologies using a uniform interpolation (UI) approach. While UI has proven beneficial for privacy-related tasks, its computational complexity has limited its practical utility compared to subset modularization approaches. To address this, we propose a highly efficient forgetting method for computing UI-based modules of $\mathcal{ALCI}$-ontologies, achieving performance comparable to subset modularization through advanced normalization and definer introduction strategies. Evaluations on benchmark datasets show superb success rates and significant efficiency gains over state-of-the-art UI and forgetting tools. Comparisons with subset modularization approaches reveal that our UI method often surpasses traditional syntax-restricted techniques in terms of module size, computation time, and memory usage. This provides the World Wide Web community with a robust framework and tooling support for knowledge reuse that adheres to signature constraints, thereby facilitating enhanced knowledge sharing across diverse Web applications.

CCS Concepts

• **Information systems** → **Web Ontology Language (OWL)**; • **Computing methodologies** → **Description logics**; **Ontology engineering**.

Keywords

Ontology, Strong forgetting, Knowledge Reuse, Semantic Web

ACM Reference Format:

Anonymous Author(s). 2018. Practical Uniform Interpolation for Efficient Computation of Signature-Restricted Modules of Web Ontologies. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Developing high-quality, scalable ontologies for the Semantic Web is a labor-intensive, time-consuming, and error-prone task. This constitutes a major barrier to the development of large-scale knowledge systems and seamless interactions of web-based agents. Since many conceptualizations are intended to be useful for a variety of tasks, an important means of removing this barrier is to encode ontologies in a "reusable" form so that large components of ontologies for a specific Web application can be created or assembled from existing ontologies. In contrast, non-reusable ontologies are limited to traditional knowledge bases, limiting their potential to contribute to the realization of the Semantic Web vision.

This paper focuses on the *Modular Reuse* of ontologies [14, 15], which refers to the practice of reusing individual, self-contained components (*modules*) of established ontologies when constructing new ontologies or extending existing ones. Rather than developing an ontology entirely from scratch, modular reuse enables the selective incorporation of modules that capture specific aspects of the domain of interest [4, 9, 10, 12, 13, 15, 36, 37, 40, 41, 43–45]. Imagine a medical ontology with a module dedicated to the domain of "Anatomy". If a new ontology is being developed for a healthcare application focused on diseases, it may reuse the anatomy module from the medical ontology, ensuring consistency in how anatomical concepts are represented without the need to redevelop that part.

There are multiple ways to define a module within an ontology. The most general form, termed *general module*, was first proposed in 2012 [32] and has since been explored as a task of semantics-preserving knowledge extraction [54]: given a set of terms in the ontology that we want to "borrow" for reuse, find the axioms in the ontology that are "relevant" to the meanings of these terms. This process, often known as *module extraction* [25], ensures that the extracted module preserves the intended semantics of the selected terms while minimizing the inclusion of unrelated axioms.

This definition is overly generic. While it ensures semantic integrity, it overlooks crucial quality criteria essential for modules to be useful in practice. Consequently, researchers have further refined this definition by introducing additional constraints on what constitutes a module. Two important refinements have emerged, namely "syntax-restricted module" and "signature-restricted module".

- *Syntax-Restricted Module*: This module type further requires the resulting ontology to be a syntactic subset of the original ontology. This ensures that the syntactic structure remains intuitive and accessible, which is particularly useful for knowledge extraction when ontologies are deployed in contexts where human experts need to interpret and work with the knowledge.
- *Signature-Restricted Module*: This module type imposes an additional restriction: the resulting ontology must contain only the "relevant terms", specifically those within the signature Σ, while excluding any terms outside of it. These modules allow for efficient knowledge extraction by condensing ontologies to task-specific essentials.

This paper focuses on the computation of signature-restricted modules in $\mathcal{ALCI}$-ontologies using uniform interpolation, also known as forgetting. *Uniform Interpolation* (UI) [24] is a non-standard

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

reasoning procedure that, given an ontology $\mathcal{O}$ and a sub-signature Σ of $\mathcal{O}$, computes a new ontology $\mathcal{M}$, termed "*uniform interpolant*", which employs only the terms in Σ while preserving their original semantics in the absence of the discarded terms. UI is said "non-standard" since it cannot be solved by reduction to the standard satisfiability testing. Its functionality extends beyond mere module extraction from the original ontology. To maintain the semantic integrity of Σ-names, new axioms must be derived from the original ontology. Thus, uniform interpolants may contain axioms that differ syntactically from those in their source ontologies, leading to their characterization as "rewritten modules".

Example 1. *Consider the ontology $\mathcal{O}$ and let $\Sigma = \{r, A_0, A_{100}\}$ be the signature containing the "relevant terms" of interest:*

$$\mathcal{O} = \{A_1 \sqsubseteq \exists r.A_1\} \cup \{A_i \sqsubseteq A_{i+1} \mid 0 \leq i \leq 99\}$$

While the uniform interpolant of $\mathcal{O}$ w.r.t. Σ (Σ-restricted module) is

$$\mathcal{M} = \{A_0 \sqsubseteq \exists r.A_{100}\},$$

the only subset module of $\mathcal{O}$ w.r.t. Σ is $\mathcal{O}$ itself.

This example nicely demonstrates that the size of uniform interpolants is significantly smaller than that of subset modules. However, achieving this compactness necessitates extensive inference, making the entire computation process highly challenging. Previous research has shown that computing uniform interpolants is at least one exponential harder than computing subset modules [3, 31, 48]. Despite its proven utility in numerous ontology-based knowledge management tasks, including debugging and repair [38, 47], merging and alignment [35, 50], versioning [17, 18, 42], semantic difference [20, 21, 28, 59], abduction and explanation generation [8, 23], and interactive ontology revision [34], UI's full potential can only be realized through the development of a highly efficient algorithm and its corresponding implementation for computing such modules.

In this paper, we present a novel practical method for computing uniform interpolants of ontologies within the $\mathcal{ALCI}$ description logic framework. Our method employs a highly optimized "forgetting" procedure that systematically eliminate non-Σ names from the original ontology to derive the uniform interpolants. Through effective normalization and inference strategies, uniform interpolants can be computed with efficiency comparable to subset modules. A comprehensive evaluation using our prototype implementation reveals superb performance in terms of success rates and efficiency across benchmark datasets from NCBO BioPortal and Oxford-ISG, highlighting a significant computational advantage over state-of-the-art forgetting tools. Moreover, we conducted an extensive comparison with prevalent subset modularization methods, focusing on module size, computation time, and memory usage. Our findings contest the assumption that subset modularization techniques are inherently more feasible for practical computation, showing that UI can not only match but often surpass these techniques in key performance metrics. By offering a computationally feasible solution for large ontologies, our UI approach supports ontology curators in facilitating controlled knowledge sharing across diverse Web applications, thereby ultimately enhancing the privacy and security of knowledge management on the Web.

2 Preliminaries

2.1 The Description Logic $\mathcal{ALCI}$

Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be pairwise disjoint, countably infinite sets of *concept* and *role* names, respectively. *Roles* in $\mathcal{ALCI}$ can be a role name $\mathsf{r} \in \mathsf{N_R}$ or the inverse r^- of r. *Concept descriptions* (or *concepts* for short) in $\mathcal{ALCI}$ have one of the following forms:

$$\top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C,$$

where $A \in \mathsf{N_C}$, C and D range over general concepts, and R over general roles. If R is a role, we define the inverse $\mathsf{Inv}(R)$ of R by $\mathsf{Inv}(r) = r^-$ and $\mathsf{Inv}(r^-) = r$, for all $r \in \mathsf{N_R}$. We assume w.l.o.g. that concepts are equivalent relative to associativity and commutativity of $\sqcap$ and $\sqcup$, $\neg$ and $^-$ are involutions, and $\top$ ($\bot$) is a unit w.r.t. $\sqcap$ ($\sqcup$).

An $\mathcal{ALCI}$*-ontology* $\mathcal{O}$ is a finite set of *axioms* in the form $C \sqsubseteq D$, known as *general concept inclusions* (or GCIs for short), where C, D are concepts (not necessarily concept names). We use $C \equiv D$ as a shortcut to represent the pair of GCIs $C \sqsubseteq D$ and $D \sqsubseteq C$. Hence, in this paper, an $\mathcal{ALCI}$-ontology is assumed to contain only GCIs. The semantics of $\mathcal{ALCI}$ is defined as usual [2].

Let $\mathcal{I}$ be an interpretation. A GCI $C \sqsubseteq D$ is *true* in $\mathcal{I}$, written $\mathcal{I} \models C \sqsubseteq D$, 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 $C \sqsubseteq D$ is said 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 $\mathcal{O}$. An ontology $\mathcal{O}_1$ is entailed by an ontology $\mathcal{O}_2$, written $\mathcal{O}_2 \models \mathcal{O}_1$, iff every model of $\mathcal{O}_2$ is also a model of $\mathcal{O}_1$.

Theorem 1. *Standard reasoning (satisfiability testing) in $\mathcal{ALCI}$ is ExpTime-complete [46].*

Theorem 1 ensures the decidability of $\mathcal{ALCI}$ while underscoring the inherent computational difficulty within this logic.

2.2 Subset Module & Uniform Interpolant

In this paper, we define modules based on the notion of *Inseparability* [3, 19]. From an application perspective, two ontologies are deemed *inseparable* if one can be substituted for the other without altering their function in any given context. This concept is rooted in the logical principle of equivalence, according to which two ontologies are *inseparable* if they yield identical logical entailments. However, traditional logical equivalence lacks the necessary adaptability to underpin modularity effectively. For example, a module within an ontology generally does not exhibit logical equivalence to the original ontology, nor does an updated ontology maintain logical equivalence to its preceding version. By parameterizing logical equivalence with a signature Σ of relevant terms, we refine the definition of logical equivalence to achieve a notion of inseparability that has exactly the desired flexibility and properties.

A *signature* $\Sigma \subseteq \mathsf{N_C} \cup \mathsf{N_R}$ is a finite set of concept and role names. For any syntactic object X — which may range over concepts, roles, GCIs, clauses, or ontologies, we denote:

- $\mathsf{sig_C}(X)$: the set of concept names occurring in X
- $\mathsf{sig_R}(X)$: the set of role names occurring in X
- $\mathsf{sig}(X)$: the union of $\mathsf{sig_C}(X)$ and $\mathsf{sig_R}(X)$

For any signature Σ, if $\mathsf{sig}(X) \subseteq \Sigma$, we say that X is a Σ-object (e.g., Σ-concept, Σ-GCI, etc.).

Definition 1 (Inseparability for $\mathcal{ALCI}$). *Let $\mathcal{O}$ and $\mathcal{M}$ be $\mathcal{ALCI}$-ontologies and Σ a signature of concept and role names. We say that $\mathcal{O}$ and $\mathcal{M}$ are* inseparable *w.r.t. Σ (or Σ-inseparable), written $\mathcal{O} \equiv_\Sigma \mathcal{M}$, if for any $\mathcal{ALCI}$-GCI $C \sqsubseteq D$ with $sig(C \sqsubseteq \mathcal{D}) \subseteq \Sigma$:*

- $\mathcal{O} \models C \sqsubseteq D$ *iff* $\mathcal{M} \models C \sqsubseteq D$.

Definition 2 (General Module for $\mathcal{ALCI}$). *Let $\mathcal{O}$ and $\mathcal{M}$ be $\mathcal{ALCI}$-ontologies and $\Sigma \subseteq sig(\mathcal{O})$ a signature of concept and role names. We say that $\mathcal{M}$ is a* general module *for $\mathcal{O}$ w.r.t. Σ iff the following conditions hold:*

- $\mathcal{O} \equiv_\Sigma \mathcal{M}$, *and*
- $\mathcal{O} \models \mathcal{M}$.

Definition 3 (Subset Module for $\mathcal{ALCI}$). *Let $\mathcal{O}$ and $\mathcal{M}$ be $\mathcal{ALCI}$-ontologies and $\Sigma \subseteq sig(\mathcal{O})$ a signature of concept and role names. We say that $\mathcal{M}$ is a* syntax-restricted module *or* subset module *for $\mathcal{O}$ w.r.t. Σ iff the following conditions hold:*

- *$\mathcal{M}$ is a general module of $\mathcal{O}$, and*
- $\mathcal{M} \subseteq \mathcal{O}$.

Definition 4 (Uniform Interpolant for $\mathcal{ALCI}$). *Let $\mathcal{O}$ and $\mathcal{M}$ be $\mathcal{ALCI}$-ontologies and $\Sigma \subseteq sig(\mathcal{O})$ a signature of concept and role names, referred to as the* interpolation signature. *We say that $\mathcal{M}$ is a* signature-restricted module *or* uniform interpolant *of $\mathcal{O}$ w.r.t. Σ iff the following conditions hold:*

- *$\mathcal{M}$ is a general module of $\mathcal{O}$, and*
- $sig(\mathcal{M}) \subseteq \Sigma$.

Computing a uniform interpolant of an ontology $\mathcal{O}$ w.r.t. a signature $\Sigma \subseteq \text{sig}(\mathcal{O})$ is equivalent to *forgetting* the complementary signature $\text{sig}(\mathcal{O}) \setminus \Sigma$ from $\mathcal{O}$ [31, 51, 60]. Thus, forgetting and UI constitute dual characterizations of the same computational task [31].

Definition 5 (Forgetting for $\mathcal{ALCI}$). *Let $\mathcal{O}$ be an $\mathcal{ALCI}$-ontology and $\mathcal{F} \subseteq sig(\mathcal{O})$ a set of concept and role names, referred to as the* forgetting signature. *An $\mathcal{ALCI}$-ontology $\mathcal{M}$ is a* result of forgetting $\mathcal{F}$ from $\mathcal{O}$ *if the following conditions hold:*

- *$\mathcal{M}$ is a uniform interpolant of $\mathcal{O}$ w.r.t. $\Sigma = sig(\mathcal{O}) \setminus \mathcal{F}$.*

The forgetting process effectively distills an ontology $\mathcal{O}$ into a more focused perspective, $\mathcal{M}$, by concentrating on a sub-signature Σ of $\mathcal{O}$. Specifically, Σ is defined as a subset of $\mathcal{O}$'s signature excluding $\mathcal{F}$, denoted as $\Sigma \subseteq \text{sig}(\mathcal{O}) \setminus \mathcal{F}$. Within the $\mathcal{ALCI}$ framework, $\mathcal{M}$ preserves the semantic behavior of $\mathcal{O}$ w.r.t. Σ, meaning they generate identical $\mathcal{ALCI}$-entailments over Σ. This definition yields two important properties:

- The result of forgetting $\mathcal{F}$ from $\mathcal{O}$ can be computed by sequentially eliminating individual names in $\mathcal{F}$, independent of the elimination order.
- Forgetting results (uniform interpolants) are unique up to logical equivalence — any results obtained from the same forgetting procedure are logically equivalent, despite potential syntactic differences in their explicit representations.

3 Related Work

Forgetting can be formalized in two ways that are closely related. Lin and Reiter [26] initially introduced model-theoretic forgetting in first-order logic — forgetting a predicate P in a theory $\mathcal{O}$ yields a new theory $\mathcal{O}'$ with models agreeing on all aspects of those of $\mathcal{O}$ except for P's interpretation. Zhang and Zhou [55] later distinguished model-theoretic forgetting from a weaker notion, the latter defined "deductively" as only retaining first-order consequences irrelevant to P. In logic, weak forgetting has been explored as a dual problem of UI [7, 16, 49]. UI, while closely related to Craig interpolation [6], imposes more restrictions. This paper employs UI as a mechanism for computing signature-restricted modules. Consequently, the forgetting notion adopted in this paper aligns with the weaker one.

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* [11, 14, 19, 30]. Research in this domain has been focused on the following problems:

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

Key theoretical findings in this domain include:

- Very few logics are known to be closed under forgetting, whether it be under the strong or weak notion, though for very limited expressivity languages like $\mathcal{EL}$. 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 [30]. This applies to $\mathcal{ALC}$ as well [11];
- Determining whether a result exists for strong forgetting is undecidable on $\mathcal{EL}$ and $\mathcal{ALC}$ [19];
- Determining whether a result exists for weak forgetting is ExpTime-complete on $\mathcal{EL}$ [29, 33] and 2ExpTime-complete on $\mathcal{ALC}$ [31];
- 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 [29, 31, 33].

The primary practical methods for forgetting are:

- Lethe [22]: Uses classic resolution calculus [39] to compute uniform interpolants for $\mathcal{ALCI}$ and several its extensions.
- Fame [61]: employs generalized Ackermann's Lemma [1] for strong forgetting in $\mathcal{ALCOIH}$ ontologies.

Several approaches to computing uniform interpolants through forgetting have been proposed [21, 27, 52, 53, 56]. However, these methods face significant practical limitations: they either scale only to small ontologies, are no longer maintained, or support only ontologies that are less expressive than $\mathcal{ALCI}$. As a result, we adopt Lethe as the state-of-the-art baseline for UI and forgetting, using it as our primary comparison benchmark.

4 Normalization of $\mathcal{ALCI}$-Ontologies

Our forgetting method operates on $\mathcal{ALCI}$-ontologies in clausal normal form, defined as a finite set of clauses.

Definition 6 (**Literals & Clauses in $\mathcal{ALCI}$**). *A* literal *in $\mathcal{ALCI}$ is a concept of the form A, $\neg A$, $\exists R.C$ or $\forall R.C$, where $A \in \mathsf{N_C}$, C is a concept, and R is a role. A* clause *in $\mathcal{ALCI}$ is a disjunction of finitely many literals.*[1] *An $\mathcal{ALCI}$-ontology is in* clausal normal form *if all its GCIs are clauses.*

Clauses are derived from GCIs by incrementally applying standard, equivalence-preserving transformations, a process completed in polynomial time. Henceforth, unless explicitly stated otherwise, $\mathcal{ALCI}$-ontologies are assumed to be sets of clauses.

We further introduce two specialized normal forms for $\mathcal{ALCI}$-ontologies: *A-reduced form*, tailored for single concept name elimination, and *r-reduced form*, tailored for single role name elimination.

Definition 7 (*A*-**Reduced Form**). *Let $A \in \mathsf{N_C}$. A clause is in* A-reduced form *if it has one of the following forms: $C \sqcup A$, $C \sqcup \neg A$, $C \sqcup \exists r.A$, $C \sqcup \exists r.\neg A$, $C \sqcup \exists r^-.A$, $C \sqcup \exists r^-.\neg A$, $C \sqcup \forall r.A$, $C \sqcup \forall r.\neg A$, $C \sqcup \forall r^-.A$ or $C \sqcup \forall r^-.\neg A$, where $r \in \mathsf{N_R}$ can be any role name, and C is a clause with $A \notin sig(C)$. An $\mathcal{ALCI}$-ontology is in* A-reduced form *if all its A-clauses are in A-reduced form.*

The A-reduced form ensures that A occurs only once within each A-clause. It consolidates all possible positions where A can occur in $\mathcal{ALCI}$-clauses. Specifically, A, in either positive or negative form, may appear at the surface level of the clause as one of its disjuncts, or immediately below an $\exists$- or $\forall$-restriction.

Definition 8 (*r*-**Reduced Form**). *Let $r \in \mathsf{N_R}$. A clause is in* r-reduced form *if it is of the form $C \sqcup \exists r.D$, $C \sqcup \exists r^-.D$, $C \sqcup \forall r.D$, or $C \sqcup \forall r.D$, where C (D) is a clause (concept) that does not contain r. An $\mathcal{ALCI}$-ontology is in* r-reduced form *if all its r-clauses are in r-reduced form.*

Any clause not conforming to these syntactic structures can be transformed into reduced form in polynomial time by incrementally applying the following transformations. Unlike the standard CNF transformations mentioned above, the conversion of an $\mathcal{ALCI}$-clause to reduced form may introduce new concept names that are not present in the original ontology, referred to as *definers* [24].

- For each A-clause instance $L_1 \sqcup \ldots \sqcup L_n$, if A occurs multiple times in this clause and any literal L_i $(1 \leq i \leq n)$ has the form $\exists R.C$ or $\forall R.C$, where R is a general role and C is a concept with $A \in \mathsf{sig}(C)$, perform the following transformation: replace C with a fresh definer $Z \in \mathsf{N_C} \setminus \mathsf{sig}(\mathcal{O})$, and add the clause $\neg Z \sqcup C$ to $\mathcal{O}$;
- For each A-clause instance $L_1 \sqcup \ldots \sqcup L_n$, if A occurs exactly once in this clause and any literal L_i $(1 \leq i \leq n)$ has the form $\exists R.C$ or $\forall R.C$, where R is a general role and $C \neq A$ is a concept such that $A \in \mathsf{sig}(C)$, replace C with a fresh definer $Z \in \mathsf{N_C} \setminus \mathsf{sig}(\mathcal{O})$, and add the clause $\neg Z \sqcup C$ to $\mathcal{O}$;
- For each r-clause instance $L_1 \sqcup \ldots \sqcup L_n$ containing a literal L_i $(1 \leq i \leq n)$ where $A \in \mathsf{sig}(L_i)$: if A appears in other literals of the clause, replace L_i with a fresh definer $Z \in \mathsf{N_C} \setminus \mathsf{sig}(\mathcal{O})$ and add the clause $\neg Z \sqcup L_i$ to $\mathcal{O}$;
- For each r-clause instance $L_1 \sqcup \ldots \sqcup L_n$ containing a literal L_i $(1 \leq i \leq n)$ of the form $\exists R.C$ or $\forall R.C$ where $r \in \mathsf{sig}(C)$, replace C with a fresh definer $Z \in \mathsf{N_C} \setminus \mathsf{sig}(\mathcal{O})$ and add $\neg Z \sqcup C$ to $\mathcal{O}$.

Lemma 1. *For an $\mathcal{ALCI}$-ontology $\mathcal{O}$ and its reduced form $\mathcal{O}'$ obtained through the above transformations, $\mathcal{O} \equiv_{sig(\mathcal{O})} \mathcal{O}'$ holds.*

Lemma 2. *For any $\mathcal{ALCI}$-ontology $\mathcal{O}$, there exists a transformation to its A-reduced or r-reduced form $\mathcal{O}'$ through a linear number of applications of the corresponding normalization rules. Moreover, $|\mathcal{O}'|$ is linear in $|\mathcal{O}|$, where $|\mathcal{O}|$ denotes the number of clauses in $\mathcal{O}$.*

Lemma 1 proves soundness while Lemma 2 shows completeness, termination, and efficiency of the normalization method.

5 Efficient Definer Introduction Mechanism

A significant contribution of this paper is the definer introduction mechanism for ontology normalization, as described in the previous section, which facilitate the efficient elimination of single concept and role names. Next, we examine the definer introduction mechanism of Lethe, the current state-of-the-art forgetting method for $\mathcal{ALCI}$, with a focus on its computational complexity. Our analysis highlights the superior efficiency of the definer introduction mechanism implemented in our approach.

Lethe operates on clauses of the form $L_1 \sqcup \ldots \sqcup L_n$, where each L_i $(1 \leq i \leq n)$ is a literal:

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

where $r \in \mathsf{N_R}$ and $A, Z \in \mathsf{N_C}$. A key distinction is that Lethe mandates every subconcept Z immediately below an $\exists$- or $\forall$-restriction to be a definer throughout the forgetting process. In contrast, our method permits more flexible specifications of Z, allowing it to be any general $\mathcal{ALCI}$-concept. Let us define the following sets:

- $\mathsf{sig_D}(\mathcal{O})$: the set of definers introduced in $\mathcal{O}$;
- $\mathsf{sub}^{\forall}_{\exists}(\mathcal{O})$: the set of all subconcepts of the form $\exists r^{(-)}.X$ or $\forall r^{(-)}.X$ in $\mathcal{O}$, where $r \in \mathsf{N_R}$ and X is a general concept;
- $\mathsf{sub}_X(\mathcal{O})$: the set of all subconcepts X in $\mathcal{O}$ with $\exists r^{(-)}.X \in \mathsf{sub}^{\forall}_{\exists}(\mathcal{O})$ or $\forall r^{(-)}.X \in \mathsf{sub}^{\forall}_{\exists}(\mathcal{O})$.

In the Lethe framework, which employs a definer reuse strategy (i.e., Lethe consistently reuses a definer to refer to identical subconcepts), an injective function f can be defined over $\mathsf{sig_D}(\mathcal{O})$, specifically $f : \mathsf{sig_D}(\mathcal{O}) \rightarrow \mathsf{sub}_X(\mathcal{O})$. f is also surjective, given Lethe's exhaustive approach to introducing definers — Lethe requires every subconcept immediately below an $\exists$- or $\forall$-restriction to be a definer. Conversely, in our method, f is defined as non-surjective. However, for both methods, the number of definers introduced in $\mathcal{O}$ during the normalization process, denoted as $|\mathsf{sig_D}(\mathcal{O})|$, is bounded by $\mathrm{O}(n)$, where n denotes the number of $\exists$- and $\forall$-restrictions in $\mathcal{O}$. This implies a linear growth in definer introduction.

Lethe employs a saturation-based reasoning approach for single name elimination from ontology $\mathcal{O}$, using a generalized resolution calculus Res [22]. The process involves two phases:

- *Pre-resolution phase*: this phase witnesses Lethe's inaugural computation of $\mathcal{O}$'s normal form to fire up Res, where definers are linearly and statically introduced.
- *Intra-resolution phase*: Res is applied exhaustively until saturation, where new entailments are iteratively generated.

[1] We define a clause as a GCI of the form $\top \sqsubseteq L_1 \sqcup \ldots \sqcup L_n$, where each L_i $(1 \leq i \leq n)$ is a literal. We typically omit the prefix "$\top \sqsubseteq$" and treat clauses as sets, implying no duplicates and no significant order. Thus, $C^{\mathcal{I}} \cup (\geq m r.D)^{\mathcal{I}}$ being true indicates that $\top \sqsubseteq C \sqcup (\geq m r.D)$ is true in $\mathcal{I}$.

In the Intra-resolution phase, LETHE applies Res rules to $\mathcal{O}$ until reaching saturation at $\text{Res}(\mathcal{O})$. For instance, the $\forall\exists$-role propagation rule applied to $C_1 \sqcup \forall r.D_1$ and $C_2 \sqcup \exists r.D_2$ yields $C_1 \sqcup C_2 \sqcup \exists r.(D_1 \sqcap D_2)$. LETHE normalizes this new entailment by introducing definer $D_{12} \in \mathsf{N_C}$ for $D_1 \sqcap D_2$ and adding $\neg D_{12} \sqcup (D_1 \sqcap D_2)$ to $\mathcal{O}$. This dynamic definer introduction during Res iterations yields an injective, non-surjective function $f' : \text{sig}_\text{D}(\text{Res}(\mathcal{O})) \rightarrow \text{sub}_X(\text{Res}(\mathcal{O}))$. This implies the size of the codomain $|\text{sub}_X(\text{Res}(\mathcal{O}))| = 2^{|\text{sub}_\text{D}(\mathcal{O})|}$. The number of definers is bounded by $\text{O}(2^n)$, where n is the number of $\exists$- and $\forall$-restrictions in $\mathcal{O}$. In contrast, our method confines normalization to the pre-resolution stage, ensuring linear definer introduction throughout the forgetting process.

As definers are extraneous to the desired signature, LETHE may need to eliminate up to $(2^n) + |\mathcal{F}|$ names and thus execute Res for $\text{O}(2^n) + |\mathcal{F}|$ iterations in the worst case. Conversely, our method introduces at most n definers and requires only $n + |\mathcal{F}|$ activations of the forgetting calculus in the worst case.

6 The Forgetting Process

Let $\mathcal{O}$ be an $\mathcal{ALCI}$-ontology and $\mathcal{F} \subseteq \text{sig}(\mathcal{O})$ a forgetting signature. The result of forgetting $\mathcal{F}$ from $\mathcal{O}$ is computed by iteratively eliminating the names in $\mathcal{F}$. The forgetting process comprises two independent calculi. Each is based on a generalized inference rule. Both rules are *replacement rules*, substituting the premises of the rule (clauses above the line) with its conclusion (below the line).

6.1 Calculus for Concept Name Elimination

The calculus for eliminating a concept name $A \in \text{sig}_\text{C}(\mathcal{O})$ from $\mathcal{O}$ is based on the *combination rule* in Figure 1, applicable when $\mathcal{O}$ is in A-reduced form. A-reduced clauses containing exactly one occurrence of A have at most 10 distinct forms, categorized as *Positive Premises* (i.e., A-clauses where A occurs positively) and *Negative Premises* (i.e., A-clauses where A occurs negatively).

Positive Premises	Notation		Negative Premises	Notation
$C \sqcup A$	$\mathcal{P}^+_\mathfrak{U}(A)$		$C \sqcup \neg A$	$\mathcal{P}^-_\mathfrak{U}(A)$
$C \sqcup \exists r.A$	$\mathcal{P}^+_\exists(A)$		$C \sqcup \exists r.\neg A$	$\mathcal{P}^-_\exists(A)$
$C \sqcup \exists r^-.A$	$\mathcal{P}^+_{\exists,-}(A)$		$C \sqcup \exists r^-.\neg A$	$\mathcal{P}^-_{\exists,-}(A)$
$C \sqcup \forall r.A$	$\mathcal{P}^+_\forall(A)$		$C \sqcup \forall r.\neg A$	$\mathcal{P}^-_\forall(A)$

Figure 3: Distinct forms of A-reduced clauses

We denote premise sets of distinct A-reduced forms as shown in Figure 3.[2] We define:

- $\mathcal{P}^+(A) = \mathcal{P}^+_\mathfrak{U}(A) \cup \mathcal{P}^+_\exists(A) \cup \mathcal{P}^+_{\exists,-}(A) \cup \mathcal{P}^+_\forall(A)$
- $\mathcal{P}^-(A) = \mathcal{P}^-_\mathfrak{U}(A) \cup \mathcal{P}^-_\exists(A) \cup \mathcal{P}^-_{\exists,-}(A) \cup \mathcal{P}^-_\forall(A)$

We further define $\mathcal{O}^{-A}$ as the set of non A-clauses in $\mathcal{O}$.

The fundamental idea of the combination rule is to resolve each positive premise with each negative one on the name being eliminated (in this case, A), hence the term "*combination*". This process yields all logical entailments of $\mathcal{O}$ in the signature $\text{sig}(\mathcal{O})\backslash A$. Given four distinct forms of both positive and negative premises, there are 16 possible combination cases. Resolving the premises α and β on the target name A yields a finite set of A-free clauses, denoted as $\text{combine}(\alpha, \beta)$. This set represents the strongest logical entailment of the premises in the signature $(\text{sig}(\alpha) \cup \text{sig}(\beta))\backslash A$. Since premise sets like $\mathcal{P}^+_\mathfrak{U}(A)$ contain A-clauses of the same form, we treat them collectively. Consequently, $\text{combine}(\mathcal{P}^+_\mathfrak{U}(A), \mathcal{P}^-_\mathfrak{U}(A))$ denotes the union of all ground combinations between these sets. More generally, the conclusion of the combination rule is expressed as $\text{combine}(\mathcal{P}^+(A), \mathcal{P}^-(A))$, featuring all such combinations. For conciseness, we represent the combination rule "at the set level".

Unlike propositional resolution, which directly resolves propositional variables, resolution in $\mathcal{ALCI}$ targets a concept name A that may occur under an $\exists$- or $\forall$-restriction. The complementarity of these literals becomes apparent only through their first-order translations (see [58] for an example). Our inference rule directly resolves literals of the forms $\exists$ and $\forall$ on r, producing a forgetting result in DLs. The process involves:

- Exhaustively applying this inference rule until saturation
- Removing all A-clauses from the saturated ontology $\mathcal{O}$

The result is a refined ontology, $\mathcal{M}$, free of A occurrences.

LEMMA 3 (SOUNDNESS OF RULE). *Let $\mathcal{O}$ be an A-reduced $\mathcal{ALCI}$-ontology with $A \in sig_C(\mathcal{O})$. If $\mathcal{M}$ is the ontology derived by applying the combination rule in Figure 1, then $\mathcal{O} \equiv_{sig(O)\backslash\{A\}} \mathcal{M}$.*

Lemma 3 establishes the partial soundness of the calculus. Specifically, the derived ontology $\mathcal{M}$ satisfies the first condition required for being a uniform interpolant of $\mathcal{O}$ w.r.t. $\Sigma = \text{sig}(\mathcal{O})\backslash\{A\}$. However, $\mathcal{M}$ may contain definers that are outside Σ, potentially violating the second condition. This will be later discussed and addressed.

6.2 Calculus for Role Name Elimination

The calculus for eliminating a role name $r \in \text{sig}_\text{R}(\mathcal{O})$ from $\mathcal{O}$ is based on the *ombination rule* in Figure 2, applicable when $\mathcal{O}$ is in r-reduced form. r-reduced clauses containing exactly one occurrence of r have at most four distinct forms, categorized as *Positive Premises* (i.e., r-clauses where r occurs positively) and *Negative Premises* (i.e., r-clauses where r occurs negatively).

Positive Premises	Notation		Negative Premises	Notation
$C \sqcup \exists r.D$	$\mathcal{P}^+_\exists(r)$		$C \sqcup \forall r.D$	$\mathcal{P}^-_\forall(r)$
$C \sqcup \exists r^-.D$	$\mathcal{P}^+_{\exists,-}(r)$		$C \sqcup \forall r^-.D$	$\mathcal{P}^+_{\forall,-}(r)$

Figure 4: Distinct forms of r-reduced clauses

We denote premise sets of distinct r-reduced forms as shown in Figure 4. We further define $\mathcal{O}^{-r}$ as the set of non r-clauses in $\mathcal{O}$.

The rule combines every positive premise α with every negative premise β to derive all logical entailments not regarding r. Given the distinct forms of positive and negative premises, there are four (2×2) different combination cases. Each combination yields a finite set $\text{combine}(\alpha, \beta)$ of r-free clauses.

LEMMA 4 (SOUNDNESS OF RULE). *Let $\mathcal{O}$ be an r-reduced $\mathcal{ALCI}$-ontology with $r \in sig_R(\mathcal{O})$. If $\mathcal{M}$ is the ontology derived by applying the combination rule in Figure 2, then $\mathcal{O} \equiv_{sig(O)\backslash\{r\}} \mathcal{M}$.*

[2] A-clauses of the form $C \sqcup \forall r^-.A$ or $C \sqcup \forall r^-.\neg A$ can be equivalently transformed into $A \sqcup \forall r.C$ or $\neg A \sqcup \forall r.C$, respectively, using Galois connections [57]. Consequently, we exclude these two cases from the A-reduced form for concept forgetting.

$$\frac{\mathcal{O}^{-\mathsf{A}}, \overbrace{B_1 \sqcup \mathsf{A}, \ldots, B_l \sqcup \mathsf{A}}^{\mathcal{P}^+_{\mathfrak{U}}(\mathsf{A})}, \overbrace{C_1 \sqcup \exists r_1.\mathsf{A}, \ldots, C_m \sqcup \exists r_m.\mathsf{A}}^{\mathcal{P}^+_{\exists}(\mathsf{A})}, \overbrace{D_1 \sqcup \exists s_1^-.\mathsf{A}, \ldots, D_n \sqcup \exists s_n^-.\mathsf{A}}^{\mathcal{P}^+_{\exists,-}(\mathsf{A})}, \overbrace{\phi_1 \sqcup \forall t_1.\mathsf{A}, \ldots, \phi_o \sqcup \forall t_o.\mathsf{A}}^{\mathcal{P}^+_{\forall}(\mathsf{A})} \quad \overbrace{E_1 \sqcup \neg\mathsf{A}, \ldots, E_{l'} \sqcup \neg\mathsf{A}}^{\mathcal{P}^-_{\mathfrak{U}}(\mathsf{A})}, \overbrace{F_1 \sqcup \exists u_1.\neg\mathsf{A}, \ldots, F_{m'} \sqcup \exists u_{m'}.\neg\mathsf{A}}^{\mathcal{P}^-_{\exists}(\mathsf{A})}, \overbrace{G_1 \sqcup \exists v_1^-.\neg\mathsf{A}, \ldots, G_{n'} \sqcup \exists v_{n'}^-.\neg\mathsf{A}}^{\mathcal{P}^-_{\exists,-}(\mathsf{A})}, \overbrace{\psi_1 \sqcup \forall w_1.\neg\mathsf{A}, \ldots, \psi_{o'} \sqcup \forall w_{o'}.\neg\mathsf{A}}^{\mathcal{P}^-_{\forall}(\mathsf{A})}}{\begin{array}{c}\mathcal{O}^{-\mathsf{A}}, \mathsf{combine}(\mathcal{P}^+_{\mathfrak{U}}(\mathsf{A}), \mathcal{P}^-_{\mathfrak{U}}(\mathsf{A})), \mathsf{combine}(\mathcal{P}^+_{\mathfrak{U}}(\mathsf{A}), \mathcal{P}^-_{\exists}(\mathsf{A})), \mathsf{combine}(\mathcal{P}^+_{\mathfrak{U}}(\mathsf{A}), \mathcal{P}^-_{\exists,-}(\mathsf{A})), \mathsf{combine}(\mathcal{P}^+_{\mathfrak{U}}(\mathsf{A}), \mathcal{P}^-_{\forall}(\mathsf{A})),\\ \mathsf{combine}(\mathcal{P}^+_{\exists}(\mathsf{A}), \mathcal{P}^-_{\mathfrak{U}}(\mathsf{A})), \mathsf{combine}(\mathcal{P}^+_{\exists}(\mathsf{A}), \mathcal{P}^-_{\exists}(\mathsf{A})), \mathsf{combine}(\mathcal{P}^+_{\exists}(\mathsf{A}), \mathcal{P}^-_{\exists,-}(\mathsf{A})), \mathsf{combine}(\mathcal{P}^+_{\exists}(\mathsf{A}), \mathcal{P}^-_{\forall}(\mathsf{A})),\\ \mathsf{combine}(\mathcal{P}^+_{\exists,-}(\mathsf{A}), \mathcal{P}^-_{\mathfrak{U}}(\mathsf{A})), \mathsf{combine}(\mathcal{P}^+_{\exists,-}(\mathsf{A}), \mathcal{P}^-_{\exists}(\mathsf{A})), \mathsf{combine}(\mathcal{P}^+_{\exists,-}(\mathsf{A}), \mathcal{P}^-_{\exists,-}(\mathsf{A})), \mathsf{combine}(\mathcal{P}^+_{\exists,-}(\mathsf{A}), \mathcal{P}^-_{\forall}(\mathsf{A})),\\ \mathsf{combine}(\mathcal{P}^+_{\forall}(\mathsf{A}), \mathcal{P}^-_{\mathfrak{U}}(\mathsf{A})), \mathsf{combine}(\mathcal{P}^+_{\forall}(\mathsf{A}), \mathcal{P}^-_{\exists}(\mathsf{A})), \mathsf{combine}(\mathcal{P}^+_{\forall}(\mathsf{A}), \mathcal{P}^-_{\exists,-}(\mathsf{A})), \mathsf{combine}(\mathcal{P}^+_{\forall}(\mathsf{A}), \mathcal{P}^-_{\forall}(\mathsf{A}))\end{array}}$$

Notation in the combination rule ($1 \le h \le l, 1 \le i \le m, 1 \le j \le n, 1 \le k \le o, 1 \le h' \le l', 1 \le i' \le m', 1 \le j' \le n', 1 \le k' \le o'$):
$B_h, C_i, D_j, \phi_k, E_{h'}, F_{i'}, G_{j'}$ and $\psi_{k'}$ are any concepts that do not contain A; $r_i, s_j, t_k, u_{i'}, v_{j'}$ and $w_{k'}$ are any role names.

1: $\mathsf{combine}(\mathcal{P}^+_{\mathfrak{U}}(\mathsf{A}), \mathcal{P}^-_{\mathfrak{U}}(\mathsf{A})) = \bigcup_{1\le h\le l}\bigcup_{1\le h'\le l'} \{B_h \sqcup E_{h'}\}$ **2:** $\mathsf{combine}(\mathcal{P}^+_{\mathfrak{U}}(\mathsf{A}), \mathcal{P}^-_{\exists}(\mathsf{A})) = \bigcup_{1\le h\le l}\bigcup_{1\le i'\le m'} \{F_{i'} \sqcup \exists u_{i'}.B_h\}$

3: $\mathsf{combine}(\mathcal{P}^+_{\mathfrak{U}}(\mathsf{A}), \mathcal{P}^-_{\exists,-}(\mathsf{A})) = \bigcup_{1\le h\le l}\bigcup_{1\le j'\le n'} \{G_{j'} \sqcup \exists v_{j'}^-.B_h\}$ **4:** $\mathsf{combine}(\mathcal{P}^+_{\mathfrak{U}}(\mathsf{A}), \mathcal{P}^-_{\forall}(\mathsf{A})) = \bigcup_{1\le h\le l}\bigcup_{1\le k'\le o'} \{\psi_{k'} \sqcup \forall w_{k'}.B_h\}$

5: $\mathsf{combine}(\mathcal{P}^+_{\exists}(\mathsf{A}), \mathcal{P}^-_{\mathfrak{U}}(\mathsf{A})) = \bigcup_{1\le i\le m}\bigcup_{1\le h'\le l'} \{C_i \sqcup \exists r_i.E_{h'}\}$ **6:** $\mathsf{combine}(\mathcal{P}^+_{\exists}(\mathsf{A}), \mathcal{P}^-_{\exists}(\mathsf{A})) = \bigcup_{1\le i\le m}\bigcup_{1\le i'\le m'} \{C_i \sqcup \exists r_i.\top, F_{i'} \sqcup \exists u_{i'}.\top\}$

7: $\mathsf{combine}(\mathcal{P}^+_{\exists}(\mathsf{A}), \mathcal{P}^-_{\exists,-}(\mathsf{A})) = \bigcup_{1\le i\le m}\bigcup_{1\le h'\le l'} \{C_i \sqcup \exists r_i.\top, G_{j'} \sqcup \exists v_{j'}^-.\top\}$

8: $\mathsf{combine}(\mathcal{P}^+_{\exists}(\mathsf{A}), \mathcal{P}^-_{\forall}(\mathsf{A})) = \bigcup_{1\le i\le m}\bigcup_{1\le k'\le o'} \{C_i \sqcup \exists r_i.\top, C_i \sqcup \psi_{k'}\}$, for any r_i and $w_{k'}$ such that $r_i = w_{k'}$

9: $\mathsf{combine}(\mathcal{P}^+_{\exists,-}(\mathsf{A}), \mathcal{P}^-_{\mathfrak{U}}(\mathsf{A})) = \bigcup_{1\le j\le n}\bigcup_{1\le h'\le l'} \{D_j \sqcup \exists s_j^-.E_{h'}\}$ **10:** $\mathsf{combine}(\mathcal{P}^+_{\exists,-}(\mathsf{A}), \mathcal{P}^-_{\exists}(\mathsf{A})) = \bigcup_{1\le j\le n}\bigcup_{1\le i'\le m'} \{D_j \sqcup \exists s_j^-.\top, F_{i'} \sqcup \exists u_{i'}.\top\}$

11: $\mathsf{combine}(\mathcal{P}^+_{\exists,-}(\mathsf{A}), \mathcal{P}^-_{\exists,-}(\mathsf{A})) = \bigcup_{1\le j\le n}\bigcup_{1\le j'\le n'} \{D_j \sqcup \exists s_j^-.\top, G_{j'} \sqcup \exists v_{j'}^-.\top\}$ **12:** $\mathsf{combine}(\mathcal{P}^+_{\exists,-}(\mathsf{A}), \mathcal{P}^-_{\forall}(\mathsf{A})) = \bigcup_{1\le j\le n}\bigcup_{1\le k'\le o'} \{D_j \sqcup \exists s_j^-.\top\}$

13: $\mathsf{combine}(\mathcal{P}^+_{\forall}(\mathsf{A}), \mathcal{P}^-_{\mathfrak{U}}(\mathsf{A})) = \bigcup_{1\le k\le o}\bigcup_{1\le h'\le l'} \{\phi_k \sqcup \forall t_k.E_{h'}\}$

14: $\mathsf{combine}(\mathcal{P}^+_{\forall}(\mathsf{A}), \mathcal{P}^-_{\exists}(\mathsf{A})) = \bigcup_{1\le k\le o}\bigcup_{1\le i'\le m'} \{F_{i'} \sqcup \phi_k, F_{i'} \sqcup \exists u_{i'}.\top\}$, for any t_k and $u_{i'}$ such that $t_k = u_{i'}$

15: $\mathsf{combine}(\mathcal{P}^+_{\forall}(\mathsf{A}), \mathcal{P}^-_{\exists,-}(\mathsf{A})) = \bigcup_{1\le k\le o}\bigcup_{1\le j'\le n'} \{G_{j'} \sqcup \exists v_{j'}^-.\top\}$

16: $\mathsf{combine}(\mathcal{P}^+_{\forall}(\mathsf{A}), \mathcal{P}^-_{\forall}(\mathsf{A})) = \bigcup_{1\le k\le o}\bigcup_{1\le k'\le o'} \{\phi_k \sqcup \psi_{k'} \sqcup \forall t_k.\bot\}$, for any t_k and $w_{o'}$ such that $t_k = w_{o'}$

Figure 1: The combination rule for eliminating a concept name $\mathsf{A} \in \mathsf{sig}_{\mathbf{C}}(\mathcal{O})$ from an A-reduced $\mathcal{ALCI}$-ontology

$$\frac{\mathcal{O}^{-\mathsf{r}}, \overbrace{C_1 \sqcup \exists \mathsf{r}.D_1, \ldots, C_m \sqcup \exists \mathsf{r}.D_m}^{\mathcal{P}^+_{\exists}(\mathsf{r})}, \overbrace{E_1 \sqcup \exists \mathsf{r}^-.F_1, \ldots, E_n \sqcup \exists \mathsf{r}^-.F_n}^{\mathcal{P}^+_{\exists,-}(\mathsf{r})}, \overbrace{V_1 \sqcup \forall \mathsf{r}.W_1, \ldots, V_p \sqcup \forall \mathsf{r}.W_p}^{\mathcal{P}^-_{\forall}(\mathsf{r})}, \overbrace{X_1 \sqcup \forall \mathsf{r}^-.Y_1, \ldots, X_q \sqcup \forall \mathsf{r}^-.Y_q}^{\mathcal{P}^-_{\forall,-}(\mathsf{r}^-)}}{\mathcal{O}^{-\mathsf{r}}, \mathsf{combine}(\mathcal{P}^+_{\exists}(\mathsf{r}), \mathcal{P}^-_{\forall}(\mathsf{r})), \mathsf{combine}(\mathcal{P}^+_{\exists}(\mathsf{r}), \mathcal{P}^-_{\forall,-}(\mathsf{r})), \mathsf{combine}(\mathcal{P}^+_{\exists,-}(\mathsf{r}), \mathcal{P}^-_{\forall}(\mathsf{r})), \mathsf{combine}(\mathcal{P}^+_{\exists,-}(\mathsf{r}), \mathcal{P}^-_{\forall,-}(\mathsf{r}))}$$

1: $\mathsf{combine}(\mathcal{P}^+_{\exists}(\mathsf{r}), \mathcal{P}^-_{\forall}(\mathsf{r})) = \bigcup_{1\le i\le m}\bigcup_{1\le x\le p} \{C_i \sqcup V_x\}$, for any D_i, W_x s.t. $D_i \sqcap W_x \sqsubseteq \bot$

2: $\mathsf{combine}(\mathcal{P}^+_{\exists}(\mathsf{r}), \mathcal{P}^-_{\forall,-}(\mathsf{r})) = \bigcup_{1\le i\le m}\bigcup_{1\le y\le q} \{C_i \sqcup Y_y\}$, for any D_i, X_y s.t. $D_i \sqcap X_y \sqsubseteq \bot$

3: $\mathsf{combine}(\mathcal{P}^+_{\exists,-}(\mathsf{r}), \mathcal{P}^-_{\forall}(\mathsf{r})) = \bigcup_{1\le j\le n}\bigcup_{1\le x\le p} \{E_j \sqcup W_x\}$, for any F_j, V_x s.t. $F_j \sqcap V_x \sqsubseteq \bot$

4: $\mathsf{combine}(\mathcal{P}^+_{\exists,-}(\mathsf{r}), \mathcal{P}^-_{\forall,-}(\mathsf{r})) = \bigcup_{1\le j\le n}\bigcup_{1\le y\le q} \{E_j \sqcup X_y\}$, for any F_j, Y_y s.t. $F_j \sqcap Y_y \sqsubseteq \bot$

Figure 2: The combination rule for eliminating a role name $\mathsf{r} \in \mathsf{sig}_{\mathbf{R}}(\mathcal{O})$ from an r-reduced $\mathcal{ALCI}$-ontology

Note that an external DL reasoner is utilized to check the side conditions of the inference rules for role name elimination. Theorem 1 establishes that subsumption checking (reducible to satisfiability checking in polynomial time) in $\mathcal{ALCI}$ is ExpTime-complete.

6.3 Completeness and Termination

Our forgetting method takes as input an $\mathcal{ALCI}$-ontology $\mathcal{O}$ and a forgetting signature $\mathcal{F}$, which comprises concept and role names to be eliminated from $\mathcal{O}$. The elimination sequence is flexible, allowing user-specified ordering. We iteratively apply the single-name elimination procedure to each name in $\mathcal{F}$, producing intermediate results and potentially introducing definers. According to Definition 5, the final result must be definer-free. Therefore, after processing $\mathcal{F}$, we attempt to eliminate the introduced definers. However, complete definer elimination is not always possible. Consider forgetting A from the $\mathcal{ALCI}$-ontology $\{A \sqsubseteq \forall r^-.A\}$, which exhibits cyclic behavior. This yields $\{\mathsf{D}_1 \sqsubseteq \exists r^-.\mathsf{D}_1\}$, where $\mathsf{D1} \in \mathsf{N_D}$ is a new definer. Attempting to forget D_1 produces $\{\mathsf{D}_2 \sqsubseteq \forall r^-.\mathsf{D}_2\}$, with $\mathsf{D2} \in \mathsf{N_D}$ as another definer, potentially leading to an infinite introduction.

While fixpoints could address cyclic situations [5], as demonstrated by Lethe, mainstream reasoning tools and the OWL API lack fixpoint support. Our method, instead of using fixpoints, ensures forgetting termination by ceasing attempts to forget D_1, retaining it in the result, and declaring an unsuccessful forgetting attempt. This approach underscores the inherent unsolvability of certain forgetting problems.

Theorem 2. *For any $\mathcal{ALCI}$-ontology $\mathcal{O}$ and forgetting signature $\mathcal{F} \subseteq sig(\mathcal{O})$, our forgetting method always terminates and produces an $\mathcal{ALCI}$-ontology $\mathcal{M}$. If $\mathcal{M}$ is definer-free, then:*

(1) $\mathcal{M}$ is the result of forgetting $\mathcal{F}$ from $\mathcal{O}$, and
(2) $\mathcal{M}$ is a uniform interpolant of $\mathcal{O}$ w.r.t. Σ, where $\Sigma = sig(\mathcal{O}) \setminus \mathcal{F}$

7 Experimental Results

We have developed a prototype (Proto) of our forgetting method in Java using OWL API Version 5.1.7[3]. To evaluate its performance and practicality, we compared this prototype against the SOTA UI method Lethe[4] and strong forgetting method Fame[5], 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[6], included a wide range of ontologies from multiple sources. The second corpus, a March 2017 snapshot from NCBO BioPortal[7], 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 role restrictions or inverse roles to remove trivial cases; this left us with 177 ontologies. We then distilled these ontologies to their $\mathcal{ALCI}$-fragments by discarding GCIs not expressible in $\mathcal{ALCI}$. Applying the same strategy to BioPortal, we obtained a collection of 76 ontologies. A detailed statistical analysis of the refined ontologies can be found in the long version of this paper accessible at https://github.com/anonymous-ai-researcher/www2025.

[3]http://owlcs.github.io/owlapi/
[4]https://lat.inf.tu-dresden.de/ koopmann/LETHE/
[5]http://www.cs.man.ac.uk/ schmidt/sf-fame/
[6]http://krr-nas.cs.ox.ac.uk/ontologies/lib/
[7]https://zenodo.org/records/439510

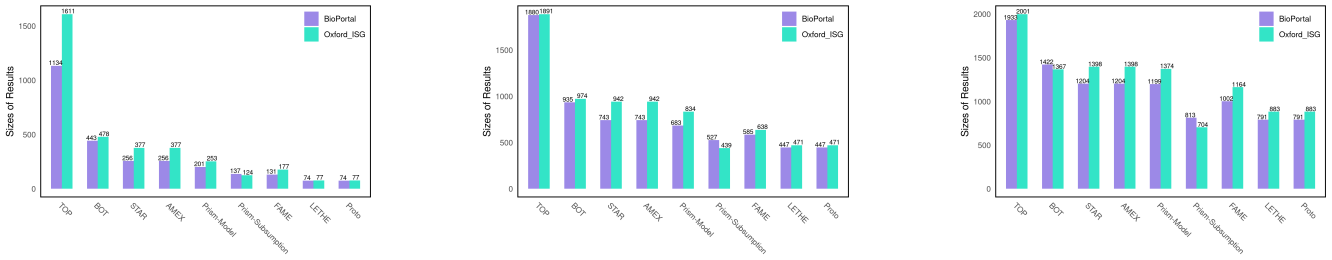
The creation of the forgetting signature $\mathcal{F}$ varies according to the specific requirements of different tasks. To address this variability, we designed three evaluation configurations to forget 10%, 30%, and 70% of the concept and role names in the signature of each ontology. We utilized a 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 evaluation, we set a maximum runtime of 300 seconds and a heap space limit of 9GB. A forgetting attempt was said *successful* if it met the following criteria: (i) all names 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.

Our method demonstrated complete effectiveness with a 100% success rate across all evaluation tracks, significantly outperforming Lethe's performance metrics. Specifically, on Oxford-ISG ontologies, Lethe achieved success rates of 84.74%, 74.34%, and 69.79% when forgetting 10%, 30%, and 50% of signature names, respectively. Similarly, on the BioPortal case, Lethe's success rates were 83.48%, 73.11%, and 69.04% for the corresponding forgetting percentages. The primary failure modes of Lethe were timeouts and memory overflows, underscoring the importance of computational efficiency 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 against various prevalent modularization methods, as well as two forgetting methods, focusing on metrics such as result size, computation time, and memory consumption. Our findings contradict traditional theoretical expectations and challenge prevailing stereotypes about UI and forgetting:

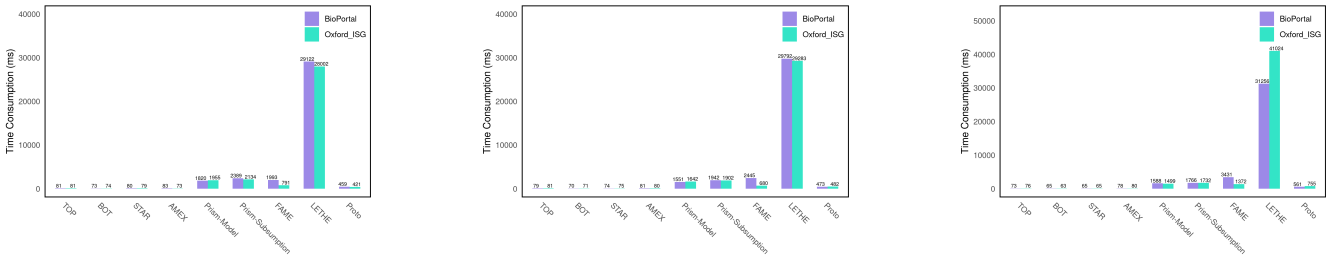
- First, the output size (Figure 5) contradicts the theoretical triple exponential growth projection. Instead, the forgetting results exhibit remarkable compactness, surpassing even modularization techniques that produce syntactic subsets of input ontologies. This contrasts sharply with previous works [31, 33] that deemed forgetting impractical due to exponential space complexity.
- Second, our method's runtime performance (Figure 6) significantly exceeds Lethe and even matches the fastest modularization methods, despite forgetting's inherently higher computational complexity [3, 48].
- Third, while forgetting generally demands higher memory usage than modularization due to entailment computation, our method achieves 20 — 40% memory reduction compared to existing forgetting methods (Figure 7), primarily through optimized definer introduction.

On Oxford-ISG ontologies, when forgetting 10%, 30%, and 50% of signature names, Lethe required definers in 67.3%, 65.7%, and 62.1% of tasks respectively, while our method reduced these demands to 24.2%, 23.1%, and 14.7%. Similarly, for BioPortal cases, while Lethe introduced definers in 28.2% of tasks across all three settings, our method dramatically reduced this requirement to 9.7%, 4.2%, and 2.9%, respectively. This significant reduction in definer



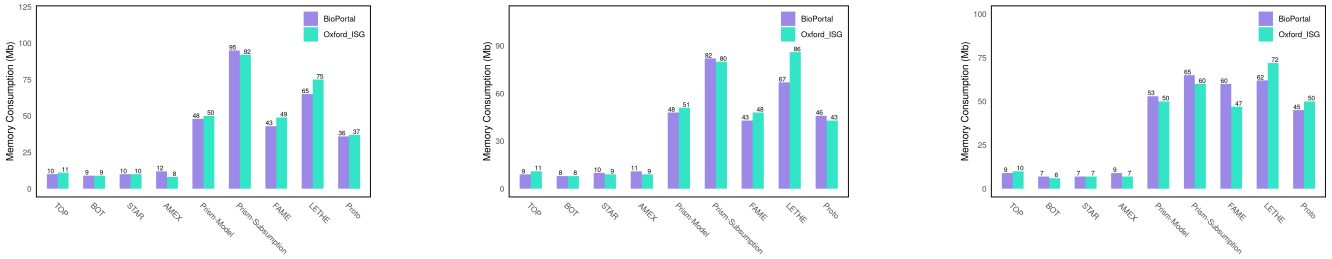
(a) Forgetting 10% of names in sig(O) (b) Forgetting 30% of names in sig(O) (c) Forgetting 50% of names in sig(O)

Figure 5: Average |Onto| in output ontologies



(a) Forgetting 10% of names in sig(O) (b) Forgetting 30% of names in sig(O) (c) Forgetting 50% of names in sig(O)

Figure 6: Average time consumption



(a) Forgetting 10% of names in sig(O) (b) Forgetting 30% of names in sig(O) (c) Forgetting 50% of names in sig(O)

Figure 7: Average memory consumption

introduction — which necessitates additional computational steps for their subsequent elimination — explains our method's superior runtime performance and memory efficiency, rivaling even subset modularization methods.

8 Conclusions and Future Work

Uniform interpolation is a non-standard reasoning procedure designed to create signature-restricted modules, conventionally considered less practical than subset modularization due to its inherent computational difficulty. In this paper, we have introduced a highly efficient uniform interpoaltion/forgetting method for computing signature-restricted modules of $\mathcal{ALCI}$-ontologies. Through careful and advanced normalization and inference strategies, we demonstrate that these modules can be computed with efficiency rivaling that of subset modules, providing the community with a powerful tool for knowledge reuse while adhering to signature constraints.

Future work will focus on extending our forgetting method to accommodate more expressive DLs, such as $\mathcal{ALCI}$ with number restrictions and several its major decidable extensions. Additionally, we aim to develop incremental forgetting techniques for dynamic ontology updates and integrate privacy-preserving mechanisms with formal guarantees, addressing both practical efficiency and security concerns in the Semantic Web environments.

References

[1] Wilhelm Ackermann. 1935. Untersuchungen über das Eliminationsproblem der mathematischen Logik. *Math. Ann.* 110, 1 (1935), 390–413.

[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. RW'16 (Lecture Notes in Computer Science, Vol. 9885).* Springer, 27–89.

[4] Enrico Giacinto Caldarola and Antonio Maria Rinaldi. 2016. An Approach to Ontology Integration for Ontology Reuse. In *Proc. IRI'16.* IEEE Computer Society, 384–393.

[5] 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.

[6] 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.

[7] 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.

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

[9] Paul Doran, Valentina A. M. Tamma, and Luigi Iannone. 2007. Ontology module extraction for ontology reuse: an ontology engineering perspective. In *Proc. CIKM'07.* ACM, 61–70.

[10] Mariano Fernández-López, Mari Carmen Suárez-Figueroa, and Asunción Gómez-Pérez. 2012. Ontology Development by Reuse. In *Ontology Engineering in a Networked World.* Springer, 147–170.

[11] 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.

[12] Asunción Gómez-Pérez and Dolores Rojas-Amaya. 1999. Ontological Reengineering for Reuse. In *Proc. EKAW'99 (Lecture Notes in Computer Science, Vol. 1621).* Springer, 139–156.

[13] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. 2007. Just the right amount: extracting modules from ontologies. In *Proc. WWW'07.* ACM, 717–726.

[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, Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. 2006. Modularity and Web Ontologies. In *Proc. KR'06.* AAAI Press, 198–209.

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

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

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

[19] 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.

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

[21] 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.

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

[23] 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.

[24] Patrick Koopmann and Renate A. Schmidt. 2013. Uniform Interpolation of $\mathcal{ALC}$-Ontologies Using Fixpoints. In *Proc. FroCoS'13 (Lecture Notes in Computer Science, Vol. 8152).* Springer, 87–102.

[25] Andrew LeClair, Alicia Marinache, Haya El Ghalayini, Wendy MacCaull, and Ridha Khédri. 2023. A Review on Ontology Modularization Techniques - A Multi-Dimensional Perspective. *IEEE Trans. Knowl. Data Eng.* 35, 5 (2023), 4376–4394.

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

[27] Zhao Liu, Chang Lu, Ghadah Alghamdi, Renate A. Schmidt, and Yizheng Zhao. 2021. Tracking Semantic Evolutionary Changes in Large-Scale Ontological Knowledge Bases. In *Proc. CIKM'21.* ACM, 1130–1139.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] Nadeschda Nikitina and Birte Glimm. 2012. Hitting the Sweetspot: Economic Rewriting of Knowledge Bases. In *Proc. ISWC'12,* Vol. 7649. Springer, 394–409.

[33] 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.

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

[35] Natalya F. Noy and Mark A. Musen. 2002. PROMPTDIFF: A Fixed-Point Algorithm for Comparing Ontology Versions. In *Proc. AAAI/IAAI'02.* AAAI/MIT Press, 744–750.

[36] Natalya Fridman Noy and Mark A. Musen. 2003. The PROMPT suite: interactive tools for ontology merging and mapping. *Int. J. Hum. Comput. Stud.* 59, 6 (2003), 983–1024.

[37] Helena Sofia Andrade N. P. Pinto and João Pavão Martins. 2001. A methodology for ontology integration. In *Proc. K-CAP'01.* ACM, 131–138.

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

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

[40] Fabiano Borges Ruy, Giancarlo Guizzardi, Ricardo de Almeida Falbo, Cássio Chaves Reginato, and Victor Amorim dos Santos. 2017. From reference ontologies to ontology patterns and back. *Data Knowl. Eng.* 109 (2017), 41–69.

[41] Sirko Schindler and Jan Martin Keil. 2019. Building Ontologies for Reuse. In *Proc. Jowo'19 (CEUR Workshop Proceedings, Vol. 2518).* CEUR-WS.org.

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

[43] Julian Seidenberg and Alan L. Rector. 2006. Web ontology segmentation: analysis, classification and use. In *Proc. WWW'06.* ACM, 13–22.

[44] Elena Simperl. 2009. Reusing ontologies on the Semantic Web: A feasibility study. *Data Knowl. Eng.* 68, 10 (2009), 905–925.

[45] Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, and Mariano Fernández-López. 2015. The NeOn Methodology framework: A scenario-based methodology for ontology development. *Appl. Ontology* 10, 2 (2015), 107–145.

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

[47] 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.

[48] 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.

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

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

[51] 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.

[52] Zhe Wang, Kewen Wang, Rodney W. Topor, and Xiaowang Zhang. 2010. Tableau-based Forgetting in $\mathcal{ALC}$ Ontologies. In *Proc. ECAI'10 (Frontiers in Artificial Intelligence and Applications, Vol. 215).* IOS Press, 47–52.

[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] Hui Yang, Patrick Koopmann, Yue Ma, and Nicole Bidoit. 2023. Efficient Computation of General Modules for $\mathcal{ALC}$ Ontologies. In *Proc. IJCAI'23.* ijcai.org, 3356–3364.

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

[56] Yizheng Zhao. 2023. Highly-Optimized Forgetting for Creating Signature-Based Views of Ontologies. In *Proc. CIKM'23.* ACM, 3444–3452.

[57] Yizheng Zhao and Schmidt Renate A. 2017. Role Forgetting for $\mathcal{ALCOQH}(\forall)$-Ontologies Using An Ackermann-Based Approach. In *Proc. IJCAI'17.* IJCAI/AAAI Press, 1354–1361.

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

[59] Yizheng Zhao, Ghadah Alghamdi, Schmidt Renate A., 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.

[60] Yizheng Zhao and Renate A. Schmidt. 2016. Forgetting Concept and Role Symbols in $\mathcal{ALCOIH}\mu^+(\triangledown, \sqcap)$-Ontologies. In *Proc. IJCAI'16*. IJCAI/AAAI Press, 1345–1352.

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