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, 16 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

1.1 Ontology Reuse in the Semantic Web

The vast scale, diversity, and dynamic nature of Web data render it a vital resource for training advanced AI models, such as large language models. However, the metadata — the underlying semantic information that provides meaning to the data — has always been overlooked. *Ontologies* [24], which utilize expressive logic and a standardized vocabulary of terms (referred to as the *signature* of the ontology), enable Web data to be linked to specific, well-defined concepts through logical axioms. By establishing these linkages, data from different Web sources can be effectively aggregated and compared in meaningful ways, which is a critical step toward realizing *the Semantic Web* vision [62].

Meanwhile, developing high-quality, scalable Semantic Web ontologies 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.[1] In contrast, non-reusable ontologies are limited to traditional knowledge bases, limiting their potential to contribute to the realization of the Semantic Web vision.

Interest in *Ontology Reuse* originated from the seminal work of Neches and Fikes [44] and Gruber [23, 25] on knowledge sharing across heterogeneous ontologies. Over the past three decades, this topic has remained a key area of research in Knowledge Representation and the Semantic Web, resulting in a considerable body of foundational, practical, and empirical studies. However, the scope of the problem is very broad, and there exist many approaches to ontology reuse, formalized to various degrees, and yet there is no clear definition of what it means to develop an ontology via reuse, and precisely what this entails [5, 12, 13, 19, 20, 22, 50, 53, 57, 58, 61, 63, 64]

1.2 Modular Reuse of Ontologies

This paper focuses on the notion of *modular reuse* of ontologies [21, 22], 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. 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. This idea was inspired by that of modularity in software engineering, which refers to the practice of breaking up a software system into separate, interchangeable components (modules). These modules are designed to be relatively

[1]This work fits into the scope of "Scalable techniques for the creation, curation, publication, maintenance, and consumption of large, Web-based, structured, reusable, knowledge graphs and ontologies" as part of the Semantic and Knowledge Track at TheWebConf 2025.

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

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

independent of each other, allowing for easier maintenance, testing, and reusability of the code [52].

Modular reuse of ontologies provides a highly desirable strategy for Web-based ontological knowledge processing for several reasons. Firstly, it enables collaborative and sustainable design by facilitating the development of ontologies as loosely coupled, self-contained units, thereby simplifying maintenance through localized modifications. Secondly, independent modules can be reused across diverse Web contexts, enhancing flexibility and efficiency. Thirdly, modular reuse optimizes reasoning tasks by allowing only relevant modules to be utilized for specific deductions, while also enabling distributed reasoning processes across multiple machines. Finally, in scenarios where privacy and security are paramount, modular reuse allows ontology owners to restrict access to specific ontology components while providing controlled reasoning services.

There are multiple ways to define a module within an ontology. The most general form, termed *general module*, was first proposed in 2012 [45] and has since been explored as a task of semantics-preserving knowledge extraction [74]: 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* [37], ensures that the extracted module preserves the intended semantics of the selected terms while minimizing the inclusion of unrelated axioms.

A formal aspect of defining a general module for an ontology $\mathcal{O}$ w.r.t. a sub-signature Σ of $\mathcal{O}$ is establishing the correctness of its extraction. Building on Garson's validity criterion for a component of a logical theory (e.g., an ontology) [16], we adopt the following conditions for a component $\mathcal{M}$ to qualify as a general module:

- *Local soundness*, i.e., Every axiom α that is provable in $\mathcal{M}$ must also be provable in $\mathcal{O}$. Formally: $\mathcal{M} \models \alpha \Longrightarrow \mathcal{O} \models \alpha$ (equivalent to $\mathcal{O} \models \mathcal{M}$);
- *Local completeness*, i.e., Every axiom α that is formulated using terms from Σ and is provable in $\mathcal{O}$ must also be provable in $\mathcal{M}$. Formally: $\mathcal{O} \models \alpha \Longrightarrow \mathcal{M} \models \alpha$

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. The primary characteristic of syntax-restricted modules is the preservation of syntactic similarity between the resulting module and 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. The goal is to retain all logical entailments regarding the terms of interest while excluding any irrelevant information. 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* [35] is a non-standard reasoning procedure that, given an ontology $\mathcal{O}$ and a sub-signature Σ of $\mathcal{O}$, computes a new ontology $\mathcal{M}$, termed "*uniform interpolant*", which employs only the terms in Σ while preserving their original semantics in the absence of the discarded terms. Uniform interpolation 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 [4, 43, 68]. Despite its proven utility in numerous ontology-based knowledge management tasks, including debugging and repair [54, 67], merging and alignment [49, 70], versioning [27, 28, 60], semantic difference [30, 31, 40, 80], abduction and explanation generation [11, 34], and interactive ontology revision [47], uniform interpolation's full potential can only be realized through the development of a highly efficient algorithm and its corresponding implementation for computing such modules.

This paper challenges the conventional view that the inherent computational difficulty poses an insurmountable obstacle to practical uniform interpolation. Specifically, we present a novel practical approach for computing uniform interpolants of ontologies within the $\mathcal{ALCI}$ description logic framework. Our method leverages 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 exceptional 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 uniform interpolation can not only match but often surpass these techniques in key performance metrics. By offering a computationally feasible solution for large ontologies, our uniform interpolation 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.

A **long version** of this paper, which includes all omitted proofs, additional illustrative examples, comprehensive experimental results, and the source code for our prototype implementation along with the test datasets, is available for review at https://github.com/anonymous-ai-researcher/www2025.

2 Preliminaries

2.1 The Description Logic $\mathcal{ALCI}$

Web ontologies are often formulated in the web ontology language (OWL) based on Description Logics (DLs) [2, 3]. DLs constitute a prominent family of knowledge representation formalisms that are widely used in ontology modeling, with various variants that differ in expressivity depending on which logical connectives are set to be used to describe domain knowledge. A basic DL, called $\mathcal{ALC}$, utilizes concept names, role names, and the logical connectives of $\neg$, $\sqcap$, $\sqcup$, $\exists$, and $\forall$ to build complex concepts. The language considered in this paper is the DL $\mathcal{ALCI}$, which extends $\mathcal{ALC}$ with inverse roles ($\mathcal{I}$). While increased expressivity enhances the capacity for knowledge representation, it typically incurs additional computational complexity.

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. $\sqcap$ ($\sqcup$). This paper adopts the following notation conventions:

- Uppercase letter A denote concept names;
- Uppercase letters B and subsequent letters in the alphabet and the Greek letters ϕ and ψ denote general concepts;
- Uppercase letter R denote general roles;
- Lowercase letters r, s, and t denote role names.

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 in terms of an *interpretation* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ is a non-empty set, known as the *domain of the interpretation*, and $\cdot^{\mathcal{I}}$ is the *interpretation function* that maps 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:

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}} \qquad \bot^{\mathcal{I}} = \emptyset \qquad (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$$
$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \qquad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$
$$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y.(x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$$
$$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y.(x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$$
$$(R^-)^{\mathcal{I}} = \{(y,x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (x,y) \in R^{\mathcal{I}}\}$$

Let $\mathcal{I}$ be an interpretation. A GCI $C \sqsubseteq D$ is *true* in $\mathcal{I}$, 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 [66].*

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

2.2 Syntax-Restricted Module (Subset Module) & Signature-Restricted Module (Uniform Interpolant)

In this paper, we define modules based on the notion of *Inseparability* [4, 29]. 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*
- $\mathcal{M} \subseteq \mathcal{O}$.

Building on these definitions, we further introduce the notion of *minimal subset module*. This addition is necessary because, for any given ontology $\mathcal{O}$ and signature Σ, $\mathcal{O}$ itself always qualifies as a subset module for $\mathcal{O}$ w.r.t. Σ. However, this trivial case clearly does not align with our intended goal for modularization, especially when considering the application of modules for knowledge reuse. The goal is to ensure that the extracted subset module, while preserving all Σ-relevant information, contains as few axioms as possible.

Definition 4 (Minimal 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* minimal subset module *for $\mathcal{O}$ and Σ iff the following conditions hold:*

- *$\mathcal{M}$ is a subset module of $\mathcal{O}$, and*
- *there does exist any proper subset $\mathcal{M}' \subset \mathcal{M}$ such that $\mathcal{M}'$ is also a subset module of $\mathcal{O}$ and Σ.*

Definition 5 (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 $\mathcal{ALCI}$-ontology $\mathcal{O}$ w.r.t. a signature $\Sigma \subseteq \mathrm{sig}(\mathcal{O})$ is equivalent to *forgetting* the complementary signature $\mathrm{sig}(\mathcal{O}) \setminus \Sigma$ from $\mathcal{O}$ [36, 43, 71, 80, 81]. Thus, forgetting and uniform interpolation constitute dual characterizations of the same computational task [43].

Definition 6 (Forgetting for $\mathcal{ALCI}$). *Let $\mathcal{O}$ be an $\mathcal{ALCI}$-ontology and $S \in sig(\mathcal{O})$ be a designated concept or role name. We say that an $\mathcal{ALCI}$-ontology $\mathcal{M}$ is a* result of forgetting S from $\mathcal{O}$ *if the following conditions hold:*

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

More generally, let $\mathcal{F} \subseteq sig(\mathcal{O})$ be a set of concept and role names, referred to as the forgetting signature. *An $\mathcal{ALCI}$-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 $\Sigma \subseteq \mathrm{sig}(\mathcal{O}) \setminus \mathcal{F}$. 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

3.1 Forgetting & Uniform Interpolation

Forgetting can be formalized in two ways that are closely related. Lin and Reiter [38] 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. They further showed that forgetting in finite theories equates to eliminating existential second-order quantifiers [15]. This implies that forgetting results may transcend first-order definability and can be computed using second-order quantifier elimination methods such as Scan[14], Dls[65], Sqema[8], and MsqEL[59].

Zhang and Zhou [75] later distinguished model-theoretic forgetting from a weaker notion, the latter defined "deductively" as only retaining first-order consequences irrelevant to P.[2] Weak forgetting typically produces results $\mathcal{O}1$ that is logically weaker than those of the stronger notion $\mathcal{O}_2$ (i.e., $\mathcal{O}_2 \models \mathcal{O}_1$). However, these notions coincide ($\mathcal{O}_2 \equiv \mathcal{O}_1$) when $\mathcal{O}_2$ is first-order definable. While $\mathcal{O}1$ is always first-order definable, it may feature an infinite set of first-order formulas.

In logic, weak forgetting has been explored as a dual problem of uniform interpolation [10, 26, 69]. Uniform interpolation, while closely related to Craig interpolation [9], imposes more restrictions. This paper employs uniform interpolation 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* [18, 21, 29, 42]. 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 $\mathcal{EL}$,

[2] A sentence is considered irrelevant to a predicate P if it is logically equivalent to another sentence not containing P.

a forgetting result within the expressivity of $\mathcal{EL}$ does not necessarily exist [42]. This applies to $\mathcal{ALC}$ as well [18];
- Determining whether a result exists for strong forgetting is undecidable on $\mathcal{EL}$ and $\mathcal{ALC}$ [29];
- Determining whether a result exists for weak forgetting is ExpTime-complete on $\mathcal{EL}$ [41, 46] and 2ExpTime-complete on $\mathcal{ALC}$ [43];
- 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 [41, 43, 46].

The primary practical methods for forgetting are:

- Lethe [33]: Uses classic resolution calculus [55] to compute uniform interpolants for $\mathcal{ALCI}$ and several its extensions.
- Fame [82]: employs generalized Ackermann's Lemma [1] for strong forgetting in $\mathcal{ALCOIH}$ ontologies.

Several approaches to computing uniform interpolants through forgetting have been proposed [31, 39, 72, 73, 77]. 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 uniform interpolation 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 7 (**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 $A \in N_C$, C is a concept, and R is a role. A* clause *in $\mathcal{ALCI}$ is a disjunction of finitely many literals.*[3] *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 8 (A-**Reduced Form**). *Let $A \in N_C$. 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 $r \in N_R$ 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.

[3] 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, $C^{\mathcal{I}} \cup (\geq mr.D)^{\mathcal{I}}$ being true indicates that $\top \sqsubseteq C \sqcup (\geq mr.D)$ is true in $\mathcal{I}$.

Definition 9 (r-**Reduced Form**). *Let $r \in N_R$. 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.*

Likewise, the r-reduced form restricts r to a single occurrence per r-clause. This form consolidates all possible positions where r can occur within an $\mathcal{ALCI}$ clause. Specifically, r, either in itself or as its inverse, can appear immediately below an $\exists$- or $\forall$-restriction.

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* [35].

- 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 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 $A \in \mathsf{sig}(C)$, perform the following transformation: replace C with a 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 $L_1 \sqcup \ldots \sqcup L_n$ where r occurs multiple times and a literal L_i $(1 \leq i \leq n)$ with $A \in \mathsf{sig}(L_i)$, perform the following transformation: replace L_i with a 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 $L_1 \sqcup \ldots \sqcup L_n$ where r occurs exactly once, if a 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 such that $r \in \mathsf{sig}(C)$), perform the following transformation: replace C with a definer $Z \in \mathsf{N_C} \setminus \mathsf{sig}(\mathcal{O})$, and add $\neg Z \sqcup L_i$ to $\mathcal{O}$.

In developing a normalization method, three important properties must be ensured to guarantee its effectiveness and efficiency:

- *Soundness*: The resulting ontology $\mathcal{O}'$ should have precisely the same logical entailments as the original $\mathcal{O}$ w.r.t. their shared signature — $\mathsf{sig}(\mathcal{O})$. This equivalence is essential for satisfying Condition (ii) of Definition 6. The specification of $\mathsf{sig}(\mathcal{O})$ as the shared signature accounts for the potential introduction of fresh definers during normalization.
- *Completeness*: The method should be able to transform any $\mathcal{ALCI}$-ontology into the reduced form.
- *Efficiency*: The normalization process must be both terminating and computationally efficient, preferably completed in polynomial time (i.e., the normalization is tractable).

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. This mechanism is complemented by novel normal forms (A-reduced and r-reduced) specifically designed to accommodate it. These normal forms, along with their associated inference rules (detailed in the subsequent section), 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. While this flexibility increases the complexity of subsequent inference rules, it ultimately enhances the overall efficiency of our method.

Let us define the following sets:

- $\mathrm{sig_D}(\mathcal{O})$: the set of definers introduced in $\mathcal{O}$;
- $\mathrm{sub}^{\forall}_{\exists}(\mathcal{O})$: 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;
- $\mathrm{sub}_X(\mathcal{O})$: the set of all subconcepts X in $\mathcal{O}$ with $\exists r^{(-)}.X \in \mathrm{sub}^{\forall}_{\exists}(\mathcal{O})$ or $\forall r^{(-)}.X \in \mathrm{sub}^{\forall}_{\exists}(\mathcal{O})$.

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 $\mathrm{sig_D}(\mathcal{O})$, specifically $f : \mathrm{sig_D}(\mathcal{O}) \to \mathrm{sub}_X(\mathcal{O})$. 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 $|\mathrm{sig_D}(\mathcal{O})|$, 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 [33]. 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.

In the Intra-resolution phase, LETHE applies Res rules to $\mathcal{O}$ until reaching saturation at $\mathrm{Res}(\mathcal{O})$. 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 $f' : \mathrm{sig_D}(\mathrm{Res}(\mathcal{O})) \to \mathrm{sub}_X(\mathrm{Res}(\mathcal{O}))$. This implies the size of the codomain $|\mathrm{sub}_X(\mathrm{Res}(\mathcal{O}))| = 2^{|\mathrm{sub_D}(\mathcal{O})|}$. The number of definers is bounded by $\mathrm{O}(2^n)$, 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 $\mathrm{O}(2^n) + |\mathcal{F}|$ 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 $\mathcal{F} \subseteq \mathrm{sig}(\mathcal{O})$ 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:

- A calculus for eliminating single concept names A
- A calculus for eliminating single role names r

Each calculus 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 (clauses below the line). The rules are sound, ensuring that the conclusion preserves the logical entailments of its premises within the remaining signature, thereby satisfying the conditions of Definition 6.

6.1 Calculus for Concept Name Elimination

The calculus for eliminating a concept name $A \in \mathrm{sig_C}(\mathcal{O})$ 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**
$C \sqcup A$	$\mathcal{P}^{+}_{\mho}(A)$		$C \sqcup \neg A$	$\mathcal{P}^{-}_{\mho}(A)$
$C \sqcup \exists r.A$	$\mathcal{P}^{+}_{\exists}(A)$		$C \sqcup \exists r.\neg A$	$\mathcal{P}^{-}_{\exists}(A)$
$C \sqcup \exists r^{-}.A$	$\mathcal{P}^{+}_{\exists,-}(A)$		$C \sqcup \exists r^{-}.\neg A$	$\mathcal{P}^{-}_{\exists,-}(A)$
$C \sqcup \forall r.A$	$\mathcal{P}^{+}_{\forall}(A)$		$C \sqcup \forall r.\neg A$	$\mathcal{P}^{-}_{\forall}(A)$

Figure 3: Distinct forms of A-reduced clauses

We denote premise sets of distinct A-reduced forms as shown in Figure 3.[4] We define:

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

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

[4] 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 [78]. Consequently, we exclude these two cases from the A-reduced form for concept forgetting.

$$\frac{\begin{array}{c}\mathcal{O}^{-\mathsf{A}}, \overbrace{B_1 \sqcup \mathsf{A}, \ldots, B_l \sqcup \mathsf{A}}^{\mathcal{P}^+_{\mho}(\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})} \\ \overbrace{E_1 \sqcup \neg\mathsf{A}, \ldots, E_{l'} \sqcup \neg\mathsf{A}}^{\mathcal{P}^-_{\mho}(\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})}\end{array}}{\begin{array}{c}\mathcal{O}^{-\mathsf{A}}, \mathsf{combine}(\mathcal{P}^+_{\mho}(\mathsf{A}), \mathcal{P}^-_{\mho}(\mathsf{A})), \mathsf{combine}(\mathcal{P}^+_{\mho}(\mathsf{A}), \mathcal{P}^-_{\exists}(\mathsf{A})), \mathsf{combine}(\mathcal{P}^+_{\mho}(\mathsf{A}), \mathcal{P}^-_{\exists,-}(\mathsf{A})), \mathsf{combine}(\mathcal{P}^+_{\mho}(\mathsf{A}), \mathcal{P}^-_{\forall}(\mathsf{A})), \\ \mathsf{combine}(\mathcal{P}^+_{\exists}(\mathsf{A}), \mathcal{P}^-_{\mho}(\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}^-_{\mho}(\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}^-_{\mho}(\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 ($1 \leq h \leq l$, $1 \leq i \leq m$, $1 \leq j \leq n$, $1 \leq k \leq o$, $1 \leq h' \leq l'$, $1 \leq i' \leq m'$, $1 \leq j' \leq n'$, $1 \leq k' \leq o'$):

$B_h, C_i, D_j, \phi_k, E_{h'}, F_{i'}, G_{j'}$ 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: $\mathsf{combine}(\mathcal{P}^+_{\mho}(\mathsf{A}), \mathcal{P}^-_{\mho}(\mathsf{A})) = \bigcup_{1\leq h\leq l} \bigcup_{1\leq h'\leq l'} \{B_h \sqcup E_{h'}\}$ **2:** $\mathsf{combine}(\mathcal{P}^+_{\mho}(\mathsf{A}), \mathcal{P}^-_{\exists}(\mathsf{A})) = \bigcup_{1\leq h\leq l} \bigcup_{1\leq i'\leq m'} \{F_{i'} \sqcup \exists u_{i'}.B_h\}$

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

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

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

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

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

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

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

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

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

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

Figure 1: The combination rule for eliminating a concept name A $\in \mathsf{sig_C}(\mathcal{O})$ 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: $\mathsf{combine}(\mathcal{P}^+_{\exists}(\mathsf{r}), \mathcal{P}^-_{\forall}(\mathsf{r})) = \bigcup_{1\leq i\leq m} \bigcup_{1\leq x\leq p} \{C_i \sqcup V_x\}$, for any D_i, W_x s.t. $D_i \sqcap W_x \sqsubseteq \bot$

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

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

4: $\mathsf{combine}(\mathcal{P}^+_{\exists,-}(\mathsf{r}), \mathcal{P}^-_{\forall,-}(\mathsf{r})) = \bigcup_{1\leq j\leq n} \bigcup_{1\leq y\leq q} \{E_j \sqcup X_y\}$, 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 r $\in \mathsf{sig_R}(\mathcal{O})$ from an r-reduced $\mathcal{ALCI}$-ontology

yields all logical entailments of $\mathcal{O}$ in the signature $\mathrm{sig}(\mathcal{O})\backslash A$. 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 combine(α,β). This set represents the strongest logical entailment of the premises in the signature $(\mathrm{sig}(\alpha)\cup\mathrm{sig}(\beta))\backslash A$. Since premise sets like $\mathcal{P}^+_{\mho}(A)$ contain A-clauses of the same form, we treat them collectively. Consequently, combine$(\mathcal{P}^+_{\mho}(A),\mathcal{P}^-_{\mho}(A))$ denotes the union of all ground combinations between these sets. More generally, the conclusion of the combination rule is expressed as combine$(\mathcal{P}^+(A),\mathcal{P}^-(A))$, featuring all such combinations. The rule also addresses cases where $\mathcal{P}^+(A)=\emptyset$ or $\mathcal{P}^-(A)=\emptyset$. In these instances, A is eliminated from $\mathcal{O}$ by substituting the top (or bottom) concept for each A occurrence, a process termed *purification*. 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 ∃- or ∀-restriction. The complementarity of these literals becomes apparent only through their first-order translations (see [79] for an example). Our inference rule directly resolves literals of the forms ∃ and ∀ 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 $\mathcal{O}\equiv_{sig(\mathcal{O})\backslash\{A\}}\mathcal{M}$.*

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. $\Sigma=\mathrm{sig}(\mathcal{O})\backslash\{A\}$. 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 $r\in\mathrm{sig_R}(\mathcal{O})$ 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 combine(α,β) of r-free clauses. The entire process involves:

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

The result is a refined ontology, $\mathcal{M}$, free of r occurrences. Lemma 5 establishes the partial soundness of the calculus.

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 $\mathcal{O}\equiv_{sig(\mathcal{O})\backslash\{r\}}\mathcal{M}$.*

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 6, 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 $\{\mathrm{D}_1\sqsubseteq\exists r^-.\mathrm{D}_1\}$, where D1 $\in\mathsf{N_D}$ is a new definer. Attempting to forget D_1 produces $\{\mathrm{D}_2\sqsubseteq\forall r^-.\mathrm{D}_2\}$, with D2 $\in\mathsf{N_D}$ as another definer, potentially leading to an infinite introduction.

While fixpoints could address cyclic situations [6], 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 $\Sigma=sig(\mathcal{O})\backslash\mathcal{F}$

7 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 interpolation/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 $\mathcal{ALC}$ with nominal and number restrictions. This extension is important for handling a broader range of knowledge bases, thereby enhancing the method's applicability in more complex Web scenarios.

References

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

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

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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).

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

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

[15] 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.

[16] James W. Garson. 1989. Modularity and Relevant Logic. *Notre Dame J. Formal Log.* 30, 2 (1989), 207–223.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

[23] Thomas R. Gruber. 1991. The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases. In *Proc. KR'91*. Morgan Kaufmann, 601–602.

[24] Thomas R. Gruber. 1993. A translation approach to portable ontology specifications. *Knowledge Acquisition* 5, 2 (1993), 199–220.

[25] Thomas R. Gruber. 1995. Toward principles for the design of ontologies used for knowledge sharing? *Int. J. Hum. Comput. Stud.* 43, 5-6 (1995), 907–928.

[26] 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.

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

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

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

[34] 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.

[35] 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.

[36] 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)*.

[37] 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.

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

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] Robert Neches, Richard Fikes, Timothy W. Finin, Thomas R. Gruber, Ramesh S. Patil, Ted E. Senator, and William R. Swartout. 1991. Enabling Technology for Knowledge Sharing. *AI Mag.* 12, 3 (1991), 36–56.

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

[46] 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.

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

[48] Riku Nortje, Katarina Britz, and Thomas Meyer. 2013. Reachability Modules for the Description Logic $\mathcal{SRIQ}$. In *Proc. LPAR'19 (Lecture Notes in Computer Science, Vol. 8312)*. Springer, 636–652.

[49] 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.

[50] 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.

[51] Koopmann P. and Jieying Chen. 2020. Deductive Module Extraction for Expressive Description Logics. In *IJCAI'20*, Christian Bessiere (Ed.). ijcai.org, 1636–1643.

[52] David Lorge Parnas. 1972. On the Criteria To Be Used in Decomposing Systems into Modules. *Commun. ACM* 15, 12 (1972), 1053–1058.

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

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

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

[56] 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.

[57] 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.

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

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

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

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

[62] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. 2006. The Semantic Web Revisited. *IEEE Intell. Syst.* 21, 3 (2006), 96–101.

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

[64] 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.

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

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

[67] 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.

[68] 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.

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

[70] 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.

[71] 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.

[72] 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.

[73] 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.

[74] 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.

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

[76] Yizheng Zhao. 2018. *Automated Semantic Forgetting for Expressive Description Logics.* Ph. D. Dissertation. The University of Manchester, UK.

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

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

[79] 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.

[80] 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.

[81] 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.

[82] 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.

A Appendix

A.1 Missing Proofs

To prove Lemma 1, we first introduce the following definition:

Definition 10 (Conservative Extension). *Given two $\mathcal{ALCI}$-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 interpretations of the 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 $r \in sig(\mathcal{O})$.*

Conservative extension has two key properties:

- *Transitivity*: For ontologies $\mathcal{O}$, $\mathcal{O}'$, and $\mathcal{O}''$, if $\mathcal{O}''$ conservatively extends $\mathcal{O}'$ and $\mathcal{O}'$ conservatively extends $\mathcal{O}$, then $\mathcal{O}''$ conservatively extends $\mathcal{O}$; this is quite obvious.
- *Subsumption Preservation*: If $\mathcal{O}'$ conservatively extends $\mathcal{O}$, then for all clauses constructed using only the names from $\mathrm{sig}(\mathcal{O})$, subsumption w.r.t. $\mathcal{O}$ and $\mathcal{O}'$ coincide; see Prop. 1.

Proposition 1. *Let $\mathcal{O}'$ be a conservative extension of $\mathcal{ALCI}$-ontology $\mathcal{O}$, and $L_1 \sqcup \ldots \sqcup L_n$ a clause constructed from $sig(\mathcal{O})$. Then:*

$$\mathcal{O} \models (\top \sqsubseteq)\ L_1 \sqcup \ldots \sqcup L_n \text{ iff } \mathcal{O}' \models (\top \sqsubseteq)\ L_1 \sqcup \ldots \sqcup L_n,$$

In other words, $\mathcal{O}$ and $\mathcal{O}'$ are $sig(\mathcal{O})$-inseparable:

$$\mathcal{O} \equiv_{sig(\mathcal{O})} \mathcal{O}'$$

Proof. We prove both directions by contraposition:
($\Rightarrow$) Assume $\mathcal{O}' \not\models L_1 \sqcup \ldots \sqcup L_n$. Then there exists a model $\mathcal{I}'$ of $\mathcal{O}'$ such that $\top^{\mathcal{I}'} \not\subseteq (L_1 \sqcup \ldots \sqcup L_n)^{\mathcal{I}'}$. Since $\mathcal{I}'$ is also a model of $\mathcal{O}$ by conservative extension, we have $\mathcal{O} \not\models L_1 \sqcup \ldots \sqcup L_n$.
($\Leftarrow$) Assume $\mathcal{O} \not\models L_1 \sqcup \ldots \sqcup L_n$. Then there exists a model $\mathcal{I}$ of $\mathcal{O}$ such that $\top^{\mathcal{I}} \not\subseteq (L_1 \sqcup \ldots \sqcup L_n)^{\mathcal{I}}$. By conservative extension, there exists a model $\mathcal{I}'$ of $\mathcal{O}'$ where the interpretations of of the names from $\mathrm{sig}(\mathcal{O})$ coincide in $\mathcal{I}$ and $\mathcal{I}'$. Since $L_1 \sqcup \ldots \sqcup L_n$ contains only names from $\mathrm{sig}(\mathcal{O})$, we have: $\top^{\mathcal{I}} = \top^{\mathcal{I}'} \not\subseteq (L_1 \sqcup \ldots \sqcup L_n)^{\mathcal{I}} = (L_1 \sqcup \ldots \sqcup L_n)^{\mathcal{I}'}$. Therefore, $\mathcal{O}' \not\models L_1 \sqcup \ldots \sqcup L_n$. □

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.*

Proof. The proof relies on showing that applying a normalization rule to $\mathcal{O}$ yields a conservative extension $\mathcal{O}'$. We detail this for the first rule; similar proofs apply to the other rules.

Consider $\mathcal{O}'$ obtained by replacing the clause $L_1 \sqcup \ldots \sqcup L_n$ with:

- $(L_1 \sqcup \cdots \sqcup L_n)^C_Z$ (where C is replaced by fresh definer Z)
- $\neg Z \sqcup C$

We verify the conditions for conservative extension:

(1) $\mathrm{sig}(\mathcal{O}') = \mathrm{sig}(\mathcal{O}) \cup \{Z\}$, so $\mathrm{sig}(\mathcal{O}) \subseteq \mathrm{sig}(\mathcal{O}')$.
(2) Let $\mathcal{I}'$ be a model of $\mathcal{O}'$. Then:
   - $\top^{\mathcal{I}'} \subseteq ((L_1 \sqcup \cdots \sqcup L_n)^C_Z)^{\mathcal{I}'}$
   - $Z^{\mathcal{I}'} \subseteq C^{\mathcal{I}'}$

   Since Z occurs only positively in $(L_1 \sqcup \cdots \sqcup L_n)^C_Z$, by monotonicity [76], we have $\top^{\mathcal{I}'} \subseteq (L_1 \sqcup \cdots \sqcup L_n)^{\mathcal{I}'}$. Thus $\mathcal{I}'$ is a model of $\mathcal{O}$.
(3) Let $\mathcal{I}$ be a model of $\mathcal{O}$. Define $\mathcal{I}'$ to coincide with $\mathcal{I}$ on all names except for Z, where $Z^{\mathcal{I}'} = C^{\mathcal{I}}$. Since:
   - $\top^{\mathcal{I}} \subseteq (L_1 \sqcup \cdots \sqcup L_n)^{\mathcal{I}}$
   - Z does not occur in $L_1 \sqcup \cdots \sqcup L_n$ or C

   We have:
   - $\top^{\mathcal{I}'} \subseteq (L_1 \sqcup \cdots \sqcup L_n)^{\mathcal{I}'}$
   - $Z^{\mathcal{I}'} = C^{\mathcal{I}} = C^{\mathcal{I}'}$

Thus $\top^{\mathcal{I}'} \subseteq ((L_1 \sqcup \cdots \sqcup L_n)^C_Z)^{\mathcal{I}'}$, showing $\mathcal{I}'$ is a model of $\mathcal{O}'$.

By transitivity, Lemma 1 follows from Proposition 1. □

To prove Lemma 2, we first introduce the following notions:

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

- $\mathsf{fq}(\mathsf{A}, X) = \begin{cases} 1, & \text{if } X = \mathsf{A} \\ 0, & \text{if } X \in \mathsf{N_C} \text{ and } X \neq \mathsf{A} \end{cases}$
- $\mathsf{fq}(\mathsf{A}, \neg E) = \mathsf{fq}(\mathsf{A}, E)$
- $\mathsf{fq}(\mathsf{A}, QR.E) = \mathsf{fq}(\mathsf{A}, E)$, for $Q \in \{\exists, \forall\}$
- $\mathsf{fq}(\mathsf{A}, E \star F) = \mathsf{fq}(\mathsf{A}, E) + \mathsf{fq}(\mathsf{A}, F)$, for $\star \in \{\sqcap, \sqcup\}$.

We define the *frequency* $\mathsf{fq}(\mathsf{A}, L_1 \sqcup \ldots \sqcup L_n)$ of A in a clause $L_1 \sqcup \ldots \sqcup L_n$ as follows, where L_i is a literal ($1 \leq i \leq n$):

- $\mathsf{fq}(\mathsf{A}, L_1 \sqcup \ldots \sqcup L_n) = \sum_{i=1}^{n} \mathsf{fq}(\mathsf{A}, L_i)$.

Let A^* be a designated occurrence of A in an A-concept 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 $\mathsf{A}^* \star D$ or $\neg\mathsf{A}^* \star D$, where $\star \in \{\sqcap, \sqcup\}$ and D is an arbitrary concept,
- $\mathsf{dp}(\mathsf{A}^*, C) = \mathsf{dp}(\mathsf{A}^*, E) + 1$, if C is of the form $D \star QR.E$, where $\star \in \{\sqcap, \sqcup\}$, $Q \in \{\exists, \forall\}$, E is a concept in which A^* occurs, and D is an arbitrary concept,

i.e., $\mathsf{dp}(\mathsf{A}^*, C)$ counts the number of $\exists$ and $\forall$-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* $\mathsf{dp}(\mathsf{A}, L_1 \sqcup \ldots \sqcup L_n)$ of A in a clause $L_1 \sqcup \ldots \sqcup L_n$ is defined as:

- $\mathsf{dp}(\mathsf{A}, L_1 \sqcup \ldots \sqcup L_n) = \sum_{i=1}^{n} \mathsf{dp}(\mathsf{A}, L_i)$.

Proposition 2. *Any* A*-clause* $L_1 \sqcup \ldots \sqcup L_n$ *with* $\mathsf{dp}(\mathsf{A}, L_1 \sqcup \ldots \sqcup L_n) = 0$ *is in* A*-reduced form.*

Proof. $\mathsf{dp}(\mathsf{A}, L_1 \sqcup \ldots \sqcup L_n) = 0$ indicates that all occurrences of A are unguarded by quantifiers ($\exists$ or $\forall$) and appear only at the surface level of each L_i. After eliminating redundant occurrences, the clause $L_1 \sqcup \ldots \sqcup L_n$ reduces to either $\mathsf{A} \sqcup C$ or $\neg\mathsf{A} \sqcup C$, where C is a clause with $\mathsf{A} \notin \mathsf{sig}(C)$, making it A-reduced. □

Proposition 3. *Any* A*-clause* $L_1 \sqcup \ldots \sqcup L_n$ *with* $\mathsf{fq}(\mathsf{A}, L_1 \sqcup \ldots \sqcup L_n) = 1$ *and* $\mathsf{dp}(\mathsf{A}, L_1 \sqcup \ldots \sqcup L_n) = 1$ *is in* A*-reduced form.*

Proof. When $\mathsf{fq}(\mathsf{A}, L_1 \sqcup \ldots \sqcup L_n) = 1$ and $\mathsf{dp}(\mathsf{A}, L_1 \sqcup \ldots \sqcup L_n) = 1$, the single occurrence of A is guarded by exactly one quantifier ($\exists$ or $\forall$). The clause $L_1 \sqcup \ldots \sqcup L_n$ is then in A-reduced form, regardless of which L_i contains A. □

Proposition 4. *For any* A*-clause* $L_1 \sqcup \ldots \sqcup L_n$, $\mathsf{fq}(\mathsf{A}, L_1 \sqcup \ldots \sqcup L_n) - \mathsf{dp}(\mathsf{A}, L_1 \sqcup \ldots \sqcup L_n) \leq 1$.

Proof. Let $m = \mathsf{dp}(\mathsf{A}, L_1 \sqcup \cdots \sqcup L_n)$. By Proposition 2, at most one unguarded occurrence of A is possible, and each role depth allows at most one guarded occurrence, yielding a total bound of $m + 1$ occurrences. □

We define $\mathsf{NLZ}_1(\mathcal{O})$ as the set derived from $\mathcal{O}$ by applying one of the normalization rules 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 $\mathsf{dp}(\mathsf{A}, L_1 \sqcup \cdots \sqcup L_n)$, for every A-clause $L_1 \sqcup \cdots \sqcup L_n$ in $\mathcal{O}$.

Proposition 5. *$dp(A, NLZ_{k-1}(\mathcal{O})) - dp(A, NLZ_k(\mathcal{O})) = 1$, where (i) $k \geq 1$ and (ii)* $\mathsf{NLZ}_{k-1}(\mathcal{O})$ *is not* A*-reduced.*

Proof. Condition (ii) ensures $\mathsf{NLZ}_{k-1}(\mathcal{O}) \neq \mathsf{NLZ}_k(\mathcal{O})$ and enables the first two normalization rules for finding the A-reduced form. We prove that applying either rule reduces the role depth of A in the resulting $\mathcal{O}$ by 1. We detail the proof for the first rule, as the second follows analogously.

Consider $\mathsf{NLZ}_{k-1}(\mathcal{O})$ containing an A-clause $L_1 \sqcup \cdots \sqcup L_n$ not in A-reduced form. Let:

- $\mathsf{dp}(\mathsf{A}, L_1 \sqcup \cdots \sqcup L_n) = m$, where $m \geq 1$
- $\mathsf{dp}(\mathsf{A}, \mathsf{NLZ}_{k-1}(\mathcal{O})) = m + x$, where $x \geq 0$
- $\mathsf{dp}(\mathsf{A}, C) = n \geq 1$ (by the Rule)

For $Q \in \{\exists, \forall\}$, we have $\mathsf{dp}(\mathsf{A}, QR.C) = n + 1$ and $m = n + 1 + y$ where $y \geq 0$. The normalization replaces C with a new definer Z, removing $L_1 \sqcup \cdots \sqcup L_n$ from $\mathsf{NLZ}_k(\mathcal{O})$ and introducing two new clauses: $(L_1 \sqcup \cdots \sqcup L_n)^C_Z$ (which denotes the clause obtained from $L_1 \sqcup \cdots \sqcup L_n$ by replacing C with Z) and $\neg Z \sqcup C$. Note that:

- $\mathsf{dp}(\mathsf{A}, (L_1 \sqcup \cdots \sqcup L_n)^C_Z) = (n + 1 + y) - (n + 1) = y$
- $\mathsf{dp}(\mathsf{A}, \neg Z \sqcup C) = \mathsf{dp}(\mathsf{A}, C) = n$

Therefore, $\mathsf{dp}(\mathsf{A}, \mathsf{NLZ}_k(\mathcal{O})) = \mathsf{dp}(\mathsf{A}, \mathsf{NLZ}_{k-1}(\mathcal{O})) - \mathsf{dp}(\mathsf{A}, L_1 \sqcup \cdots \sqcup L_n) + y + n = x + y + n$. Since $\mathsf{dp}(\mathsf{A}, \mathsf{NLZ}_{k-1}(\mathcal{O})) = m + x = x + y + n + 1$, we conclude: $\mathsf{dp}(\mathsf{A}, \mathsf{NLZ}_{k-1}(\mathcal{O})) - \mathsf{dp}(\mathsf{A}, \mathsf{NLZ}_k(\mathcal{O})) = 1$. □

We define the *size* of an ontology $\mathcal{O}$, denoted as $|\mathcal{O}|$, as the sum of the clauses in $\mathcal{O}$.

Proposition 6. *Let $NLZ_k(\mathcal{O})$ be the set of clauses obtained after applying the kth round of the last two normalization rules to the ontology $\mathcal{O}$ for finding its r-reduced form. Then, for $k \geq 1$ and non-A-reduced $NLZ_{k-1}(\mathcal{O})$, either: $|NLZ_{k-1}(\mathcal{O})| - |NLZ_k(\mathcal{O})| = 1$ or $dp(r, NLZ_{k-1}(\mathcal{O})) - dp(r, NLZ_k(\mathcal{O})) = 1$.*

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 $\mathcal{O}$.*

Proof. Proposition 4 states that any unreduced A-clause $L_1 \sqcup \cdots \sqcup L_n$ can be reduced to a set $\mathsf{NLZ}_k(L_1 \sqcup \cdots \sqcup L_n)$ of clauses in k steps with each clause α in $\mathsf{NLZ}(L_1 \sqcup \cdots \sqcup L_n)$ having $\mathsf{dp}(\mathsf{A}, \alpha) \leq 1$. If $\mathsf{dp}(\mathsf{A}, \alpha) = 0$, α is either a non-A-clause, or in A-reduced form (as per Proposition 1). If $\mathsf{dp}(\mathsf{A}, \alpha) = 1$, α may not be in A-reduced 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 $\mathsf{O}(n)$, where n is the count of $\exists$-restrictions in $\mathcal{O}$. This proves termination and completeness of the normalization procedure. □

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 $\mathcal{O} \equiv_{sig(\mathcal{O}) \setminus \{A\}} \mathcal{M}$.*

Proof. The combination cases 1, 2, 3, 4, 5, 9, and 13 essentially reverse the normalization process, thereby preserving inseparability directly. We then focus on proving Rule IR3. Rule 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 6.

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 6.

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 6.

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 **??** is a conservative extension of $\mathcal{O}^{-\mathsf{A}}$. According to Proposition 1, Lemma **??** 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 $\mathcal{ALCI}$-ontology in r-NF, and $\mathcal{O}^{-r}$ an ontology obtained from forgetting $\{\mathsf{r}\}$ from $\mathcal{O}$ using the inference rules in Figure 2, then we have:*

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

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

Lemma 5 (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 $\mathcal{O} \equiv_{sig(\mathcal{O})\backslash\{r\}} \mathcal{M}$.*

Proof. We treat the first combination case in detail. The others can be proved similarly. Regarding the first one, we prove that the premises $\mathcal{O}$ are a conservative extension of the *conclusion* $\mathcal{O}^{-\mathsf{r}}$. Obviously, $\mathrm{sig}(\mathcal{O}) = \mathrm{sig}(\mathcal{O}^{-\mathsf{r}}) \cup \{\mathsf{r}\}$, and hence $\mathrm{sig}(\mathcal{O}^{-\mathsf{r}}) \subseteq \mathrm{sig}(\mathcal{O})$, satisfying Condition (1) of Definition 6.

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

$$\Delta^{\mathcal{I}} \subseteq {C_1}^{\mathcal{I}} \cup (\exists r.D_1)^{\mathcal{I}} \tag{13}$$

$$\Delta^{\mathcal{I}} \subseteq {V_1}^{\mathcal{I}} \cup (\forall r.W_1)^{\mathcal{I}} \tag{14}$$

$${D_1}^{\mathcal{I}} \cap {W_1}^{\mathcal{I}} \subseteq \emptyset \tag{15}$$

Inclusion (13) implies $(\neg C_1)^{\mathcal{I}} \subseteq (\exists \mathsf{r}.V_1)^{\mathcal{I}}$, and similarly Inclusion (14) implies $(\neg V_1)^{\mathcal{I}} \subseteq (\exists \mathsf{r}.W_1)^{\mathcal{I}}$. When combined, they further imply $(\neg C_1)^{\mathcal{I}} \cap (\neg V_1)^{\mathcal{I}} \subseteq (\exists \mathsf{r}.D_1)^{\mathcal{I}} \cap (\forall \mathsf{r}.W_1)^{\mathcal{I}}$. Assume $(\neg C_1)^{\mathcal{I}} \cap (\neg V_1)^{\mathcal{I}} \neq \emptyset$, then there exists a domain element $x \in \Delta^{\mathcal{I}}$ such that $x \in (\neg C_1)^{\mathcal{I}} \cap (\neg V_1)^{\mathcal{I}}$ and then $x \in (\exists \mathsf{r}.D_1)^{\mathcal{I}} \cap (\forall \mathsf{r}.W_1)^{\mathcal{I}}$. This means that there exists an element $y \in \Delta^{\mathcal{I}}$ such that $(x, y) \in \mathsf{r}^{\mathcal{I}}$ and $y \in {V_1}^{\mathcal{I}}$ and $y \in {W_1}^{\mathcal{I}}$. This contradicts with Inclusion (15). Thus, such an element t does not exist, and neither does x. Therefore, we have $(\neg C_1)^{\mathcal{I}} \cap (\neg V_1)^{\mathcal{I}} \subseteq (\exists \mathsf{r}.D_1)^{\mathcal{I}} \cap (\forall \mathsf{r}.W_1)^{\mathcal{I}} = \emptyset$, and then $\mathcal{I} \models C_1 \sqcup V_1$.

Assume that $\mathcal{I}'$ is a model of $\mathcal{O}^{-\mathsf{r}}$. Let $\mathcal{I}$ be the interpretation that coincides with $\mathcal{I}'$ on all concept and role names with the exception of r, D_1, and W_1. For the exceptions, we define their extensions in $\mathcal{I}$ as $(\exists \mathsf{r}.D_1)^{\mathcal{I}} = (\exists \mathsf{r}.E)^{\mathcal{I}} = C^{\mathcal{I}'}$. 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 $C^{\mathcal{I}'} = C^{\mathcal{I}} \subseteq (\exists r.D)^{\mathcal{I}}$ and $(F^{\mathcal{I}} \sqcap \exists r.E)^{\mathcal{I}} \subseteq G^{\mathcal{I}}$

Thus, O is a conservative extension of O'. Because of the transitivity of conservative extension, O in Lemma 4 is a conservative extension of O^{-r}. According to Proposition 1, Lemma 4 holds. □

Theorem 1. *Given any $\mathcal{ALCI}$-ontology O and any forgetting signature $\mathcal{F} \subseteq sig(O)$ as input, our forgetting method always terminates and returns an $\mathcal{ALCI}$-ontology $\mathcal{V}$. If $\mathcal{V}$ does not contain any definers, then it is a result of forgetting $\mathcal{F}$ from 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 **??** and the termination of the forgetting calculi. The method's soundness is ensured by Lemma **??** and its counterpart lemma for r-normalization, along with Lemmas **??** and 4. □

A.2 Related Work — Subset Module Extraction

Similar to forgetting, subset modules can be categorized into *model subset modules* (model-theoretic notion) and *subsumption subset modules* (deductive notion). These can be further refined into three variants according to specific characteristics: *plain*, *self-contained*, and *depleting* modules. Plain modules represent the basic form of subset modules as defined above, while self-contained and depleting modules impose additional constraints on this basic form.

Definition 11 (Self-Contained Module for $\mathcal{ALCI}$). *Let O and $\mathcal{M}$ be $\mathcal{ALCI}$-ontologies and $\Sigma \subseteq sig(O)$ a signature of concept and role names. We say that $\mathcal{M}$ is a* self-contained module *of O w.r.t. Σ iff the following conditions hold:*

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

As shown by the second condition above, a self-contained module $\mathcal{M}$ must not only preserve logical entailments with the original ontology O over Σ, but must also maintain the logical entailments over all names contained within $\mathcal{M}$ itself. This is indeed a stringent requirement, particularly since $\mathcal{M}$ often needs to incorporate names from outside Σ to preserve the semantic equivalence for the names within Σ. A depleting module $\mathcal{M}$ imposes an even more stringent restriction: the remaining axioms in O (i.e., $O \setminus \mathcal{M}$) must be completely independent of $\mathcal{M}$, containing no logical entailments involving any names in $\mathrm{sig}(\mathcal{M})$. These requirements arise from the practical demands of effective knowledge reuse.

Definition 12 (Depleting Module for $\mathcal{ALCI}$). *Let O and $\mathcal{M}$ be $\mathcal{ALCI}$-ontologies and $\Sigma \subseteq sig(O)$ a signature of concept and role names. We say that $\mathcal{M}$ is a* depleting module *of O w.r.t. Σ iff the following conditions hold:*

- *$\mathcal{M}$ is a subset module of O, and*
- *$O \setminus \mathcal{M} \equiv_{\Sigma \cup sig(\mathcal{M})} \emptyset$.*

The minimality criterion for both self-contained and depleting modules follows the original definition for plain subset modules — no proper subset $\mathcal{M}' \subset \mathcal{M}$ should itself constitute a module of the corresponding type.

Specifically, MEX[5] implements a specialized algorithm for extracting minimal, self-contained, and depleting modules from acyclic $\mathcal{ELI}$-ontologies in polynomial time [30], which was later extended to DL-Lite ontologies [32]. AMEX[6] further extends this approach to acyclic $\mathcal{ALCIQ}$-ontologies, preserving self-containment and depletingness at the cost of computational efficiency [17].

Locality-based methods (LBMs), which have been integrated into the OWL API,[7] extract self-contained and depleting modules from $\mathcal{SROIQ}$-ontologies in polynomial time [21]. LBMs come in three variants based on their GCI locality testing: syntactic $\bot$ locality (BOT), syntactic $\top$ locality (TOP), and their nested version, syntactic $\bot\top^*$ locality (STAR). LBMs have spawned two major extensions: reachability-based methods (RBMs) and Datalog-based methods (DBMs). RBMs, implemented in CEL for $\mathcal{SRIQ}$-ontologies [48], come in $\bot$, $\top$ and $\bot\top^*$ variants. While $\bot$-reachability matches $\bot$-locality for ontologies in certain normal forms, other reachability variants typically yield smaller modules than their LBM counterparts but sacrifice self-containment and depletingness, thus failing to capture all Σ-related information. DBMs, implemented in PrisM[8] [56], maintain self-containment and depletingness while extracting modules from $\mathcal{SROIQ}$-ontologies in polynomial time.

While MEX provides exact solutions, all other approaches focus on computing approximations of model subset modules. Subsumption modules, despite their theoretical importance, remain understudied primarily because their deductive nature fails to guarantee *robustness under replacement* — a key property for safely reusing modules in different contexts [29]. To the best of our knowledge, the methods proposed by [7] and [51] are the only approaches for extracting subsumption modules, but their implementations are not publicly available. Therefore, we compare our approach against the following baselines: TOP, BOT, STAR, AMEX, and PrisM, excluding MEX as it supports only $\mathcal{ELI}$ but not $\mathcal{ALCI}$.

A.3 Experimental Results

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 UI method Lethe[10] and strong forgetting method Fame[11], 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[12], included a wide range of ontologies from multiple sources. The second corpus, a March 2017 snapshot from NCBO BioPortal[13], 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

[5] https://cgi.csc.liv.ac.uk/ konev/software/
[6] https://github.com/sadger/module-extraction
[7] http://owlapi.sourceforge.net
[8] https://github.com/anaphylactic/PrisM
[9] http://owlcs.github.io/owlapi/
[10] https://lat.inf.tu-dresden.de/ koopmann/LETHE/
[11] http://www.cs.man.ac.uk/ schmidt/sf-fame/
[12] http://krr-nas.cs.ox.ac.uk/ontologies/lib/
[13] https://zenodo.org/records/439510

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 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 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.

Table 1: Experiment results over Oxford-ISG and BioPortal (Time: Time Consumption, Mem: Memory Consumption, SR: Success Rate, TR: Timeout Rate, RER: Runtime Error Rate)

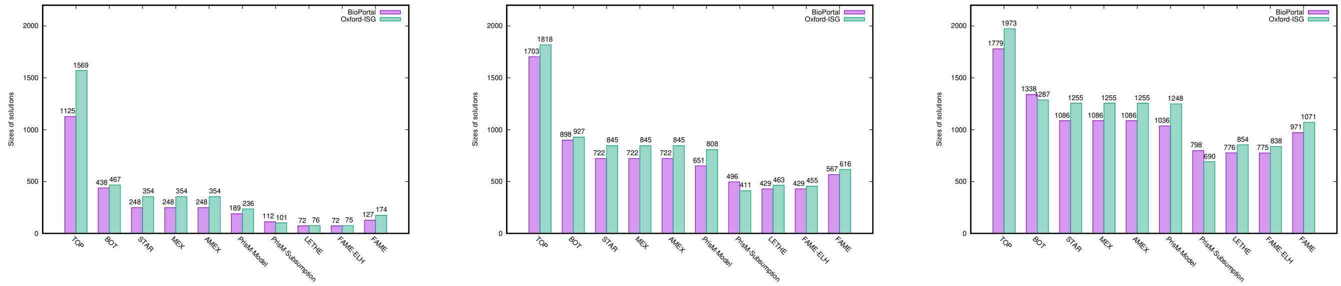
Oxford	%	PART	Time (sec.)	Mem (MB)	SR	TR	RER
Lethe	10%	I	4.23	37.06	92.42	4.10	3.48
		II	9.72	58.76	86.68	11.07	2.25
		III	14.58	88.54	77.86	22.14	0.00
	30%	I	12.46	52.24	87.09	9.45	3.48
		II	29.23	75.16	75.41	22.35	2.24
		III	41.36	123.11	67.82	32.18	0.00
	50%	I	14.63	71.36	79.91	16.61	3.48
		II	43.16	134.65	71.31	26.45	2.24
		III	74.02	189.13	64.55	35.45	0.00
Proto	10%	I	0.17	24.33	100	0.00	0.00
		II	0.46	38.54	100	0.00	0.00
		III	0.85	59.21	100	0.00	0.00
	30%	I	0.27	35.66	100	0.00	0.00
		II	0.74	51.02	100	0.00	0.00
		III	1.05	86.77	100	0.00	0.00
	50%	I	0.80	48.13	100	0.00	0.00
		II	1.33	91.21	100	0.00	0.00
		III	1.54	130.32	100	0.00	0.00
BioPortal	**%**	**PART**	**Time (sec.)**	**Mem (MB)**	**SR**	**TR**	**RER**
Lethe	10%	I	5.03	39.96	92.33	5.04	2.63
		II	11.16	59.04	85.58	10.58	3.57
		III	14.89	95.83	75.46	24.54	0.00
	30%	I	14.07	53.26	83.29	14.08	2.63
		II	32.11	88.33	73.01	23.42	3.57
		III	46.06	133.20	65.50	34.50	0.00
	50%	I	14.23	76.48	77.24	20.13	2.63
		II	45.73	140.11	69.00	27.43	3.57
		III	81.53	187.93	60.60	39.40	0.00
Proto	10%	I	0.16	21.34	100	0.00	0.00
		II	0.43	34.01	100	0.00	0.00
		III	0.83	52.03	100	0.00	0.00
	30%	I	0.31	31.23	100	0.00	0.00
		II	0.67	47.54	100	0.00	0.00
		III	1.04	78.27	100	0.00	0.00
	50%	I	0.76	44.09	100	0.00	0.00
		II	1.34	88.55	100	0.00	0.00
		III	1.53	120.81	100	0.00	0.00

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:

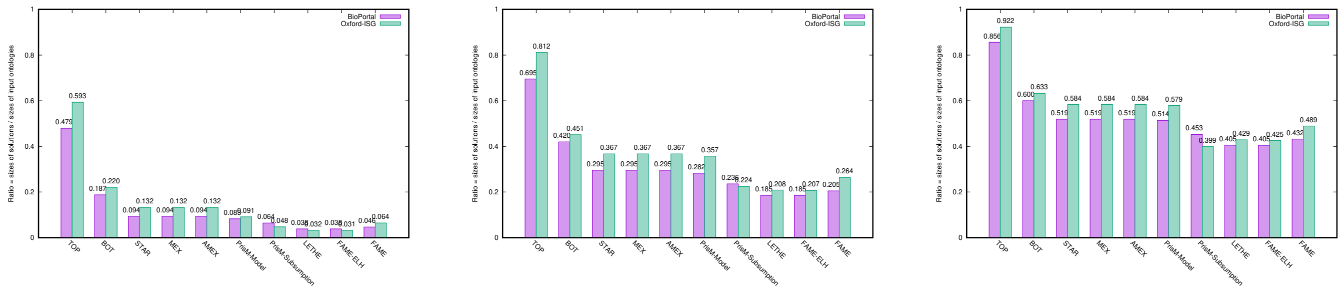
- First, the output size (Figure 5) contradicts the theoretical triple exponential growth projection. Instead, the forgetting results exhibit remarkable compactness, surpassing even modularization techniques that produce syntactic subsets of input ontologies. This contrasts sharply with previous works [43, 46] that deemed forgetting impractical due to exponential space complexity. Figure 6 quantifies this efficiency through output-to-input size ratios.
- Second, our method's runtime performance (Figure 7) significantly exceeds Lethe and even matches the fastest modularization methods, despite forgetting's inherently higher computational complexity [4, 68].
- 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 8), primarily through optimized definer introduction.

Table 2 demonstrates our method's efficiency in definer introduction compared to Lethe. 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 requirements 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 introduction — which necessitates additional computational steps for their subsequent elimination — explains our method's superior runtime performance and memory efficiency, matching even modularization methods.



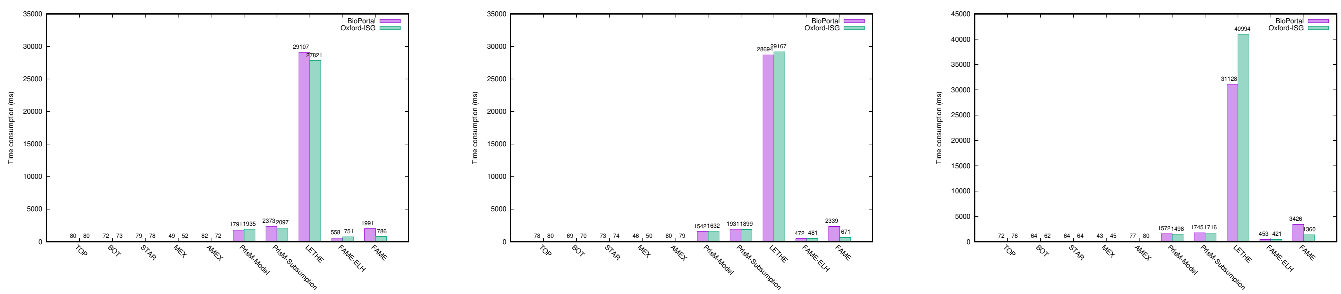
(a) Forgetting 10% of names in sig($\mathcal{O}$) **(b) Forgetting 30% of names in sig($\mathcal{O}$)** **(c) Forgetting 50% of names in sig($\mathcal{O}$)**

Figure 5: Average |Onto| in output ontologies



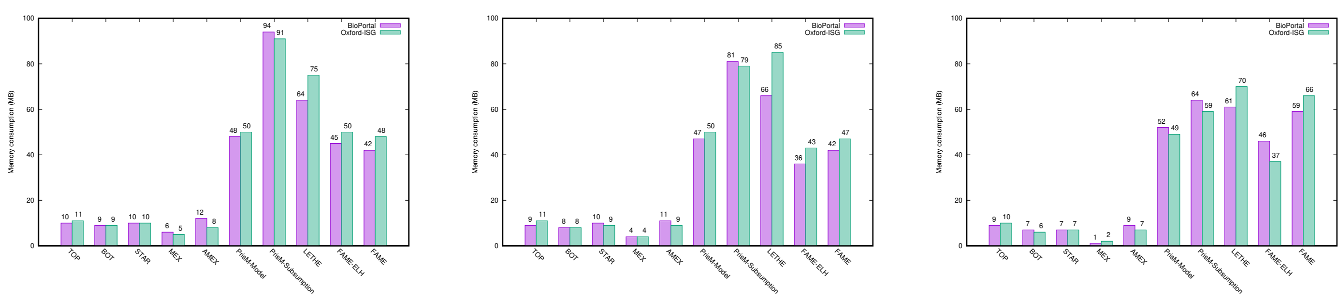
(a) Forgetting 10% of names in sig($\mathcal{O}$) **(b) Forgetting 30% of names in sig($\mathcal{O}$)** **(c) Forgetting 50% of names in sig($\mathcal{O}$)**

Figure 6: Average ratio of |Onto|: output vs. input



(a) Forgetting 10% of names in sig($\mathcal{O}$) **(b) Forgetting 30% of names in sig($\mathcal{O}$)** **(c) Forgetting 50% of names in sig($\mathcal{O}$)**

Figure 7: 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 8: Average memory consumption

Table 2: Definers introduced during forgetting (Oxford)

Onto Code	LETHE (0.1)	Proto (0.1)	LETHE (0.3)	Proto (0.3)	LETHE (0.5)	Proto (0.5)
00356	1719	111	5260	340	8760	566
00357	1410	0	4315	0	7185	0
00358	101	0	309	0	515	0
00359	515	24	1575	73	2625	122
00366	33	2	101	6	168	10
00367	40	14	122	43	204	71
00402	1614	118	4937	361	8225	602
00403	1907	0	5833	0	9715	0
00411	250	38	765	116	1275	194
00412	679	0	2077	0	3460	0
00413	1085	34	3320	104	5530	173
00423	619	68	1893	208	3155	347
00433	198	16	606	49	1009	82
00435	1	0	3	0	5	0
00445	120	3	368	9	612	15
00451	1749	0	5390	0	8920	0
00452	3163	73	9680	223	16120	372
00457	60	0	184	0	306	0
00458	68	0	208	0	347	0
00464	172	0	526	0	877	0
00468	5	0	15	0	25	0
00469	53	0	162	0	270	0
00494	1699	0	5195	0	8650	0
00495	2335	127	7150	389	11890	648
00497	8316	619	25450	1893	42350	3155
00498	6220	0	19030	0	31700	0
00505	4	0	12	0	20	0
00512	40	0	122	0	204	0
00513	38	5	116	15	194	25
00514	29	2	89	6	148	10
00515	729	0	2231	0	3715	0
00519	59	0	182	0	298	0
00520	76	4	232	12	387	20
00522	3144	0	9620	0	16020	0
00523	4224	213	12920	652	21520	1086
00527	502	17	1535	52	2550	87
00544	4558	6	13940	18	23210	30
00545	4714	16	14420	49	24015	82
00546	1952	70	5970	214	9945	357
00547	1950	119	5965	364	9935	607
00548	84	0	257	0	428	0
00550	2	0	6	0	10	0
00562	57	0	174	0	291	0
00563	60	0	184	0	306	0
00570	38	3	116	9	194	15
00571	39	2	119	6	199	10
00578	604	80	1848	245	3078	408
00580	7	0	21	0	35	0
00581	5	0	15	0	25	0
00589	81	0	248	0	413	0
00591	47	0	144	0	240	0
00592	90	0	275	0	459	0
00593	192	4	587	12	978	20
00594	174	0	532	0	887	0
00596	175	0	535	0	892	0
00600	173	0	529	0	882	0
00605	82	5	251	15	418	25
00606	62	0	190	0	316	0
00627	498	4	1524	12	2538	20
00629	524	0	1603	0	2670	0
00639	353	22	1080	67	1799	112
00640	408	49	1248	150	2082	250
00645	1990	111	6085	340	10150	567
00646	2382	95	7285	292	11950	486