Breaking the Complexity Barrier — Practical Signature-Restricted Module Computation for Web Ontologies

No Author Given

No Institute Given

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. Signature-restricted modules, which preserve all logical entailments over a designated vocabulary, offer strong guarantees of semantic fidelity and vocabulary control. However, computing such modules remains challenging due to the tension between semantic preservation and symbol elimination.
To address this challenge, this paper presents a novel forgetting method for computing signature-restricted modules of $\mathcal{ALCI}$-ontologies. Our approach implements a definer-controlled normalization strategy that avoids the exponential blow-up observed in the state-of-the-art Lethe system, reducing symbol growth to linear scale and enabling a tailored inference process. Empirical results on NCBO BioPortal and Oxford-ISG demonstrate that our method significantly outperforms Lethe in success rate and computation time. Moreover, compared to syntax-restricted (subset-based) modularization approaches, our method produces more compact, semantically faithful modules with comparable efficiency. This provides the community with a robust framework and tooling support for knowledge reuse that adheres to signature constraints, thereby facilitating enhanced knowledge sharing across diverse Semantic Web applications.

1 Introduction

This paper focuses on the 'modular reuse' of Web ontologies [19,18], which refers to the practice of reusing individual, self-contained components (i.e., *modules*) from existing ontologies when constructing or extending new ones. In contrast to developing ontologies entirely from scratch, modular reuse supports the selective integration of domain-specific knowledge, promoting terminological consistency, improving interoperability, and reducing modeling effort [51,44,45].

There are multiple approaches to defining what constitutes a module within an ontology. A foundational set of semantic conditions — ensuring that a module preserves the meaning of a selected set Σ of terms within a logical theory — was initially introduced by Garson [12]. These conditions, though applicable to general logic-based formalisms, were later adapted and applied by Cuenca Grau et al. [19] to the modularization of Web ontologies formulated in Description Logics (DLs) [1] — the logical foundation of OWL. Since then, this notion has

become a well-established formal basis for semantics-preserving knowledge extraction in the Semantic Web [18,29,26,13,49,6,57,31,62]. Although the specific term 'general module' was only coined years later by Nikitina and Glimm [39], the underlying modular approach has been continuously applied and refined across numerous domains since its initial formalization [52,56,16,55,4].

However, this general module notion remains overly generic. While it ensures semantic integrity, it overlooks critical quality criteria such as comprehensibility, modular conciseness, or vocabulary control — factors that are essential for modules to be useful in practice. To address these limitations, subsequent research has proposed additional constraints that refine the core idea of semantics-preserving modularity. Two prominent refinements in this direction are 'syntax-restricted modules' and 'signature-restricted modules':

- *Syntax-restricted modules* restrict the output module to be a syntactic subset of the original ontology. By preserving the original syntactic structure, these modules remain readable and intuitive — an advantage in contexts where human experts are expected to interpret or maintain the extracted knowledge within deployed ontologies.
- *Signature-restricted modules*, on the other hand, restrict the output module to contain only those symbols that belong to a pre-specified signature Σ — the set of terms selected for reuse — while explicitly excluding all others.

Example 1. Consider the following ontology $\mathcal{O}$ and let $\Sigma = \{r, A_0, A_{100}\}$ denote the signature consisting of the *relevant terms* selected for reuse:

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

A possible signature-restricted module of $\mathcal{O}$ w.r.t. Σ is

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

which captures the semantics of Σ while excluding irrelevant intermediate terms. In contrast, the only syntax-restricted (subset) Σ-module of $\mathcal{O}$ is $\mathcal{O}$ itself, as all axioms are syntactically necessary to ensure semantic preservation for Σ.

This example shows that signature-restricted modules can achieve substantially greater compactness compared to their syntax-restricted counterparts. The benefit of such compactness is twofold: (i) Smaller modules reduce the cognitive and computational overhead in downstream tasks such as ontology merging and alignment, reasoning, and maintenance. (ii) By strictly adhering to the target vocabulary, signature-restricted modules establish clearer semantic boundaries and support more focused reuse — particularly valuable in domain-specific deployments and privacy-aware settings, where irrelevant terms may introduce noise, ambiguity, or unintended information leakage.

However, achieving this level of compactness is computationally demanding. It requires reasoning over the entire ontology to derive exactly those entailments that are relevant to the reuse signature, while carefully excluding all other terms.

This entails non-trivial inference — often far beyond what standard subsumption or satisfiability checking can provide. The fundamental challenge lies in navigating the tension between two opposing constraints: preserving full semantic meaning over Σ, and simultaneously eliminating all non-Σ terms.

To address this challenge, this paper adopts a forgetting-based approach to computing signature-restricted modules. Given an ontology $\mathcal{O}$ and a target signature Σ — a set of concept and role names for reuse — forgetting eliminates all non-Σ terms from $\mathcal{O}$ while retaining all logical entailments over Σ (i.e., ensuring both signature restriction and semantic preservation), thereby providing a natural foundation for this task. Formally, letting $\mathsf{sig}(\mathcal{O})$ denote the signature of $\mathcal{O}$, we define the *forgetting signature* as $\mathcal{F} = \mathsf{sig}(\mathcal{O}) \setminus \Sigma$. The result of forgetting $\mathcal{F}$ from $\mathcal{O}$ constitutes a signature-restricted module $\mathcal{M}_\Sigma$ of $\mathcal{O}$ w.r.t. Σ.

In particular, we introduce a novel and highly efficient forgetting method for computing signature-restricted modules within the $\mathcal{ALCI}$ framework — an expressive DL that extends the basic $\mathcal{ALC}$ with inverse roles and underpins many real-world ontologies in biomedical and security domains. Inverse roles are essential for modeling bidirectional relationships such as those between symptoms and diseases or agents and actions. However, their inclusion increases the difficulty of forgetting, as information can propagate symmetrically and non-locally. This propagation requires careful reasoning to ensure that the resulting module remains both compact and semantically faithful.

Our method achieves high efficiency primarily through a novel normalization strategy that avoids the exponential blow-up observed in LETHE— currently the only method for forgetting in $\mathcal{ALCI}$. By carefully controlling the introduction of auxiliary concept names — referred to as definers — we ensure that the number of introduced symbols grows linearly in practice. This transformation necessitates a more fine-grained inference procedure, specifically tailored to handle the linearized normalization output. Empirical results on two benchmark datasets — NCBO BioPortal and Oxford-ISG — show that our method consistently outperforms LETHE by a large margin in both success rate and computation time. Compared to syntax-restricted (subset-based) modularization approaches, our method not only often produces more compact, semantically faithful modules, but also remains comparable in terms of computation time and memory usage.

These findings challenge the long-standing assumption that subset modularization is inherently more tractable, demonstrating that forgetting-based computation of signature-restricted modules can be both feasible and advantageous in practice. By enabling compact and signature-conforming module extraction in expressive ontology languages, our approach supports scalable, privacy-aware knowledge reuse and controlled knowledge sharing in Semantic Web applications.

2 Preliminaries

2.1 The Description Logic $\mathcal{ALCI}$

Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be disjoint and 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 its inverse r^-. *Concept*

descriptions (or simply *concepts*) 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, D range over concepts, and R over 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 $r \in \mathsf{N_R}$.

An *$\mathcal{ALCI}$-ontology* $\mathcal{O}$ is a finite set of *general concept inclusions* (GCIs) of the form $C \sqsubseteq D$, where C and D are concepts. We use $C \equiv D$ as shorthand for the pair $C \sqsubseteq D$ and $D \sqsubseteq C$. The semantics of $\mathcal{ALCI}$ is defined as usual [1].

Let $\mathcal{I}$ be an interpretation. A GCI $C \sqsubseteq D$ holds in $\mathcal{I}$, written $\mathcal{I} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. We say $\mathcal{I}$ is a *model* of an ontology $\mathcal{O}$, denoted $\mathcal{I} \models \mathcal{O}$, if every GCI holds in $\mathcal{O}$. A GCI $C \sqsubseteq D$ is a *logical entailment* of $\mathcal{O}$, written $\mathcal{O} \models C \sqsubseteq D$, if it holds in every model of $\mathcal{O}$.

Our forgetting method operates on $\mathcal{ALCI}$-ontologies in *clausal normal form*.

Definition 1 (Literals and Clauses in $\mathcal{ALCI}$). *A* literal *in $\mathcal{ALCI}$ is a concept of the form A, $\neg A$, $\exists R.C$, or $\forall R.C$, where $A \in \mathsf{N_C}$, R is a role, and C is a concept. A clause in $\mathcal{ALCI}$ is 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 omit the prefix '$\top \sqsubseteq$' and treat clauses as sets, meaning that they contain no duplicates and their order is not important. Hence, when stating that $C^{\mathcal{I}} \cup (\geq m r.D)^{\mathcal{I}}$ holds, we imply that $\top \sqsubseteq C \sqcup (\geq m r.D)$ holds in $\mathcal{I}$. An $\mathcal{ALCI}$-ontology is in clausal normal form if all its GCIs are clauses.*

Clauses are obtained by applying a sequence of equivalence-preserving transformations to GCIs. This process runs in polynomial time. Unless stated otherwise, $\mathcal{ALCI}$-ontologies are assumed to be a finite set of *clauses* in this paper.

2.2 Signature-Restricted Module & Forgetting

We define modules based on the notion of *Inseparability* [26,3]. 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 upholds 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 be a concept, role, GCI, clause, or ontology — we define $\mathsf{sig_C}(X)$ and $\mathsf{sig_R}(X)$ as the sets of concept names and role names occurring in X, respectively, and let $\mathsf{sig}(X) = \mathsf{sig_C}(X) \cup \mathsf{sig_R}(X)$. If $\mathsf{sig}(X) \subseteq \Sigma$, we call X a *Σ-object* (e.g., a Σ-concept or Σ-GCI).

Definition 2 (Inseparability). *Let $\mathcal{O}$ and $\mathcal{M}$ be two ontologies, and let Σ be a signature of concept and role names. $\mathcal{O}$ and $\mathcal{M}$ are said* inseparable *w.r.t. Σ (or Σ-*inseparable*), denoted $\mathcal{O} \equiv_\Sigma \mathcal{M}$, if for every GCI $C \sqsubseteq D$ with $\mathsf{sig}(C \sqsubseteq D) \subseteq \Sigma$, it holds that: $\mathcal{O} \models C \sqsubseteq D$ iff $\mathcal{M} \models C \sqsubseteq D$.*

Definition 3 (General Module for $\mathcal{ALCI}$). *Let $\mathcal{O}$ and $\mathcal{M}$ be $\mathcal{ALCI}$-ontologies and $\Sigma \subseteq \mathsf{sig}(\mathcal{O})$ a signature of concept and role names. $\mathcal{M}$ is a* general module *of $\mathcal{O}$ w.r.t. Σ iff the following conditions hold: (i) $\mathcal{O} \equiv_\Sigma \mathcal{M}$, and (ii) $\mathcal{O} \models \mathcal{M}$.*

Definition 4 (Subset Module for $\mathcal{ALCI}$). *$\mathcal{M}$ is a* subset module *or* syntax-restricted module *of $\mathcal{O}$ w.r.t. Σ iff the following conditions hold: (i) $\mathcal{M}$ is a general module of $\mathcal{O}$, and (ii) $\mathcal{M} \subseteq \mathcal{O}$.*

Building on this definition, 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 of $\mathcal{O}$ w.r.t. Σ. However, this trivial case clearly does not align with our intended goal for modularization. The goal is to ensure that the extracted subset module, while preserving all Σ-relevant information, contains as few axioms as possible.

Definition 5 (Minimal Subset Module for $\mathcal{ALCI}$). *$\mathcal{M}$ is a* minimal subset module *of $\mathcal{O}$ w.r.t. Σ iff the following conditions hold: (i) $\mathcal{M}$ is a subset module of $\mathcal{O}$, and (ii) there does exist any proper subset $\mathcal{M}' \subset \mathcal{M}$ such that $\mathcal{M}'$ is a subset module of $\mathcal{O}$ w.r.t. Σ.*

Definition 6 (Signature-Restricted Module for $\mathcal{ALCI}$). *$\mathcal{M}$ is a* signature-restricted module *of $\mathcal{O}$ w.r.t. Σ iff the following conditions hold: (i) $\mathcal{M}$ is a general module of $\mathcal{O}$, and (ii) $\mathsf{sig}(\mathcal{M}) \subseteq \Sigma$.*

Computing a signature-restricted module of an ontology $\mathcal{O}$ w.r.t. a signature $\Sigma \subseteq \mathsf{sig}(\mathcal{O})$ is equivalent to *forgetting* the complementary signature $\mathsf{sig}(\mathcal{O}) \setminus \Sigma$ from $\mathcal{O}$ [38,60,67]. Thus, forgetting constitutes dual characterizations of the same computational task [38].

Definition 7 (Forgetting for $\mathcal{ALCI}$). *Let $\mathcal{O}$ be an $\mathcal{ALCI}$-ontology and $\mathcal{F} \subseteq \mathsf{sig}(\mathcal{O})$ a set of concept and role names, called the* forgetting signature*. An $\mathcal{ALCI}$-ontology $\mathcal{M}$ is a* result of forgetting $\mathcal{F}$ from $\mathcal{O}$ *if the following conditions hold: (i) $\mathcal{O} \equiv_\Sigma \mathcal{M}$ and (ii) $\mathsf{sig}(\mathcal{M}) \subseteq \mathsf{sig}(\mathcal{O}) \setminus \mathcal{F}$.*

Two important properties follow from this definition: (i) the forgetting process can be performed incrementally — by eliminating symbols in $\mathcal{F}$ one at a time in any order; and (ii) forgetting results are unique up to logical equivalence — any results obtained from the same forgetting process are logically equivalent, despite potential syntactic differences in their explicit representations.

3 Related Work

Forgetting can be formalized in two closely related ways. Lin and Reiter initially proposed a model-theoretic notion of forgetting in first-order logic [34], where forgetting a predicate P from a theory $\mathcal{O}$ yields a new theory $\mathcal{O}'$ whose models coincide with those of $\mathcal{O}$ modulo the interpretation of P. The authors also showed that, in finite theories, forgetting equates to eliminating existential second-order

quantifiers [11]. This implies that results of forgetting may transcend first-order definability and can be computed through second-order quantifier elimination.

Zhang and Zhou [63] later distinguished model-theoretic forgetting from a weaker, deductive notion. The latter is defined as preserving only the first-order consequences of a theory that are irrelevant to the forgotten P. In general, weak forgetting produces a theory $\mathcal{O}_1$ that is logically weaker than the result $\mathcal{O}_2$ of model-theoretic forgetting (i.e., $\mathcal{O}_2 \models \mathcal{O}_1$), although the two coincide ($\mathcal{O}_1 \equiv \mathcal{O}_2$) when the latter is first-order definable. While $\mathcal{O}_1$ is always first-order definable, it may feature an infinite set of formulas [63]. In logic, weak forgetting has been studied as the dual of uniform interpolation [58,9,20], a notion closely related to Craig interpolation [8], but stronger. In this paper, we adopt the weak notion of forgetting as the basis for computing signature-restricted modules.

Forgetting has been generalized to various DLs through model-theoretic or deductive notions of inseparability and conservative extension [14,18,37,26]. The central research problems in this context include determining whether a given DL $\mathcal{L}$ is closed under forgetting; establishing the decidability and representational properties of forgetting results; analyzing the computational complexity of deriving such results; and developing practical algorithms for computing them.

Significant theoretical results reveal that very few DLs are closed under either strong or weak forgetting — even in lightweight ones such as $\mathcal{EL}$. For example, forgetting in $\mathcal{EL}$ or $\mathcal{ALC}$ may yield results that cannot be expressed in the original language [37,14]. It is undecidable whether a strong forgetting result exists in $\mathcal{EL}$ or $\mathcal{ALC}$ [26], while the corresponding existence problem for weak forgetting is ExpTime-complete in $\mathcal{EL}$ [36,40] and 2ExpTime-complete in $\mathcal{ALC}$ [38]. Moreover, even when a weak forgetting result exists within the source language, its size can be triple-exponential in the size of the original ontology [36,40,38].

Currently, practical forgetting systems are Lethe and Fame. Lethe [30] employs a resolution-based calculus [47] to compute forgetting in $\mathcal{ALC}$ and several of its extensions, including $\mathcal{ALCI}$. In contrast, Fame [68] employs a generalization of Ackermann's Lemma to perform strong forgetting in the more expressive logic $\mathcal{ALCOIH}$. Other systems, such as NUI [28] and the resolution-based method of [61] for $\mathcal{EL}$ and $\mathcal{SHQ}$, are no longer maintained. We therefore adopt Lethe and Fame as the state-of-the-art baselines for comparison in this paper.

Subset modularization, analogous to forgetting, can be categorized into model-theoretic and deductive approaches. Our empirical experiments compare all major existing subset modularization methods. Due to space limits, an account of these methods is provided in an *extended version* of this paper. Previous research has shown that computing signature-restricted modules (results of forgetting) is at least one exponential harder than computing subset modules [17,18,38,3,57]. Despite its proven utility in numerous ontology-based knowledge management tasks, including debugging and repair [46,54], merging and alignment [59,43], versioning [22,23,50], semantic difference [27,28,35,66], abduction and explanation generation [10,32], and interactive ontology revision [41], forgetting's full potential can only be realized through the development of a highly efficient algorithm and its corresponding implementation for computing such modules.

4 Normalization: Linear Definer Introduction Strategy

Our method operates on two specialized normal forms for $\mathcal{ALCI}$-ontologies: the *A-reduced form*, for eliminating a single concept name A, and the *r-reduced form*, for eliminating a single role name r.

Definition 8 (A-Reduced Form). *Let $A \in \mathsf{N_C}$. A clause is in* A-reduced form *if it has the form $C \sqcup L$, where $L \in \{A, \neg A, Qr.A, Qr.\neg A, Qr^-.A, Qr^-.\neg A\}$ for $Q \in \{\exists, \forall\}$ and $r \in \mathsf{N_R}$, and C is a concept with $A \notin \mathsf{sig}(C)$. 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 appears at most once per clause and only in syntactically normalized positions — namely, as a direct disjunct or as the immediate filler of an $\exists$ or $\forall$ restriction. This normal form captures all structural contexts in which a concept name may occur in an $\mathcal{ALCI}$-clause.

Definition 9 (r-Reduced Form). *Let $r \in \mathsf{N_R}$. A clause is in* r-reduced form *if it has the form $C \sqcup L$, where $L \in \{Qr.D, Qr^-.D\}$ for $Q \in \{\exists, \forall\}$, and C and D are concepts with $r \notin \mathsf{sig}(C) \cup \mathsf{sig}(D)$. An $\mathcal{ALCI}$-ontology is in* r-reduced form *if all its r-clauses are in r-reduced form.*

The r-reduced form ensures that r appears at most once per clause and only in a normalized position — namely, as the role filler of an $\exists$ or $\forall$ restriction.

Any clause not conforming to these forms can be transformed into reduced form in polynomial time through a set of equivalence-preserving rewriting steps. Unlike standard transformations to clausal normal form, this reduction process may introduce fresh concept names — called '*definers*' — which serve to isolate non-conforming subformulas [33].

- For each A-clause instance $L_1 \sqcup \ldots \sqcup L_n$ where A appears more than once, if there is a literal L_i $(1 \leq i \leq n)$ of the form $\exists R.C$ or $\forall R.C$ with $A \in \mathsf{sig}(C)$, replace C with a fresh $Z \in \mathsf{N_C}$, and add $\neg Z \sqcup C$ to $\mathcal{O}$.
- For each A-clause instance $L_1 \sqcup \ldots \sqcup L_n$ where A appears exactly once, if there is a literal L_i $(1 \leq i \leq n)$ of the form $\exists R.C$ or $\forall R.C$ with $A \in \mathsf{sig}(C)$ and $C \neq A$, replace C with a fresh $Z \in \mathsf{N_C}$, and add $\neg Z \sqcup C$ to $\mathcal{O}$.
- For each r-clause instance $L_1 \sqcup \ldots \sqcup L_n$, if there is a literal L_i $(1 \leq i \leq n)$ with $r \in \mathsf{sig}(L_i)$ and r appears elsewhere in the clause, replace L_i with a fresh definer $Z \in \mathsf{N_C}$ and add $\neg Z \sqcup L_i$ to $\mathcal{O}$.
- For each r-clause instance $L_1 \sqcup \ldots \sqcup L_n$, if there is a literal L_i of the form $\exists R.C$ or $\forall R.C$ with $r \in \mathsf{sig}(C)$, replace C with a fresh definer $Z \in \mathsf{N_C}$ and add $\neg Z \sqcup C$ to $\mathcal{O}$.

Lemma 1. *Let $\mathcal{O}$ be an $\mathcal{ALCI}$-ontology and $\mathcal{O}'$ its reduced form obtained using the normalization rules described above. Then $\mathcal{O} \equiv_{\mathsf{sig}(\mathcal{O})} \mathcal{O}'$.*

Lemma 2. *For any $\mathcal{ALCI}$-ontology $\mathcal{O}$, a reduced form $\mathcal{O}'$ (either A-reduced or r-reduced) can be computed via a linear number of normalization steps. Moreover, $|\mathcal{O}'|$ is linear in $|\mathcal{O}|$, where $|\mathcal{O}|$ denotes the number of clauses in $\mathcal{O}$.*

Lemma 1 establishes soundness of the normalization, i.e., semantic inseparability from the original ontology $\mathcal{O}$ w.r.t. its signature $\mathsf{sig}(\mathcal{O})$, while Lemma 2 guarantees its completeness, termination, and complexity.

A contribution of this paper is our linear definer introduction strategy for efficient ontology normalization. Rather than a standalone technique, this strategy operates within an integrated framework that incorporates purpose-built normal forms and a specialized inference calculus. The normal forms — A-reduced and r-reduced —precisely structure ontologies for the systematic elimination of individual concept or role names. These forms critically determine the normalization pathway and directly shape the design and application of subsequent inference rules. Together, these components constitute a coherent forgetting system that effectively computes signature-restricted modules.

To underscore the advantages of our approach, we contrast it with the definer introduction strategy of LETHE, the current state-of-the-art forgetting system, and demonstrate significant gains in scalability and computational efficiency.

LETHE transforms ontologies into clauses of the form $L_1 \sqcup \cdots \sqcup L_n$, where each literal L_i is of the form A, $\neg A$, $\exists r.Z$, $\forall r.Z$, $\exists r^-.Z$, or $\forall r^-.Z$, with $r \in \mathsf{N_R}$ and $A, Z \in \mathsf{N_C}$. A key restriction in LETHE is that every subconcept Z directly under a role restriction must be a definer, regardless of its structure. In contrast, our method allows Z to be either A itself or any A-free concept in concept forgetting, or any r-free concept in role forgetting, thereby reducing definer proliferation. Although this flexibility requires a more refined inference calculus, it leads to significantly greater efficiency in practice.

We define the following sets: $\mathsf{sig_D}(\mathcal{O})$ denotes the set of definers introduced in $\mathcal{O}$; $\mathsf{sub}^{\forall}_{\exists}(\mathcal{O})$ denotes 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 concept; $\mathsf{sub}_X(\mathcal{O})$ denotes the set of all subconcepts X in $\mathcal{O}$ such that $\exists r^{(-)}.X \in \mathsf{sub}^{\forall}_{\exists}(\mathcal{O})$ or $\forall r^{(-)}.X \in \mathsf{sub}^{\forall}_{\exists}(\mathcal{O})$.

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

Beyond normalization, LETHE performs saturation-based reasoning using a resolution calculus Res [30]. This involves a *pre-resolution phase*, where definers are linearly introduced to transform $\mathcal{O}$ into normal form, and an *intra-resolution phase*, where Res iteratively derives new entailments until saturation.

During the intra-resolution phase, LETHE exhaustively applies Res's inference rules to $\mathcal{O}$ until saturation is reached, yielding the saturated ontology $\mathsf{Res}(\mathcal{O})$. 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 $f' : \mathsf{sig_D}(\mathsf{Res}(\mathcal{O})) \to \mathsf{sub}_X(\mathsf{Res}(\mathcal{O}))$. This implies the size of the codomain $|\mathsf{sub}_X(\mathsf{Res}(\mathcal{O}))| = 2^{|\mathsf{sub_D}(\mathcal{O})|}$. The number of definers is bounded by $\mathsf{O}(2^n)$, where n denotes 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.

Because definers must be eliminated to ensure signature restriction, LETHE may need to forget up to $2^n+|\mathcal{F}|$ names, leading to up to $\mathcal{O}(2^n)+|\mathcal{F}|$ iterations of the resolution calculus in the worst case. In comparison, our method introduces at most n definers and performs at most $n+|\mathcal{F}|$ rounds of the forgetting calculus, yielding substantial gains in efficiency and scalability.

5 The Elimination Calculi

Our forgetting method eliminates a single concept or role name using two independent calculi, each based on a generalized inference rule. These rules function as replacement rules, whereby the premises (i.e., clauses above the inference line) are replaced by the conclusion (i.e., the clause below the line).

5.1 Single Concept Name Elimination

The elimination of a single concept name $A \in \mathsf{sig_C}(\mathcal{O})$ from A-reduced ontologies is based on the *combination rule* in Figure 1. A-reduced clauses with exactly one A occurrence exhibit at most 10 distinct forms, categorized as *Positive Premises* (where A occurs positively) and *Negative Premises* (where A occurs negatively),[1] with specific forms and notation detailed in Figure 1. Note that 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 [65]. Consequently, we exclude these two cases from the A-reduced form for concept name elimination; therefore, there are $4 \times 4 = 16$ combination cases. For the combination process, we define $\mathcal{P}^+(A)$ as the union of all positive premise sets and $\mathcal{P}^-(A)$ as the union of all negative premise sets, while $\mathcal{O}^{-A}$ denotes the set of non-A-clauses.

The core idea of the combination rule is to resolve each positive clause with each negative one on the name A, generating all entailments over $\mathsf{sig}(\mathcal{O}) \setminus \{A\}$. Each resolution step yields a finite set of A-free clauses, denoted by $\mathsf{combine}(\alpha, \beta)$ for any clause pair α and β. Since clauses are grouped by syntactic form, combinations are computed at the group level (e.g., $\mathsf{combine}(\mathcal{P}^+_{\forall}(A), \mathcal{P}^-_{\exists}(A))$), and the overall result is the union $\mathsf{combine}(\mathcal{P}^+(A), \mathcal{P}^-(A))$. After the resolution, all A-clauses are removed, yielding a rewritten ontology $\mathcal{M}$, devoid of A.

Lemma 3. *Let $\mathcal{O}$ be an A-reduced $\mathcal{ALCI}$-ontology. If $\mathcal{M}$ is the ontology derived by applying the combination rule in Figure 1, then $\mathcal{O} \equiv_{\mathsf{sig}(O)\setminus\{A\}} \mathcal{M}$.*

[1] Specifically, an occurrence of a symbol $\mathcal{S}$ in a clause is said to be *positive* if it appears under an even number of (explicit or implicit) negations, and *negative* otherwise.

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

($1 \leq h \leq l$, $1 \leq i \leq m$, $1 \leq j \leq n$, $1 \leq k \leq o$, $1 \leq h' \leq l'$, $1 \leq i' \leq m'$, $1 \leq j' \leq n'$, $1 \leq k' \leq o'$): B_h, C_i, D_j, ϕ_k, $E_{h'}$, $F_{i'}$, $G_{j'}$ and $\psi_{k'}$ are concepts not containing A.

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

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

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

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

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

Fig. 1: The combination rule for eliminating a concept name $\mathsf{A} \in \mathsf{sig_C}(\mathcal{O})$

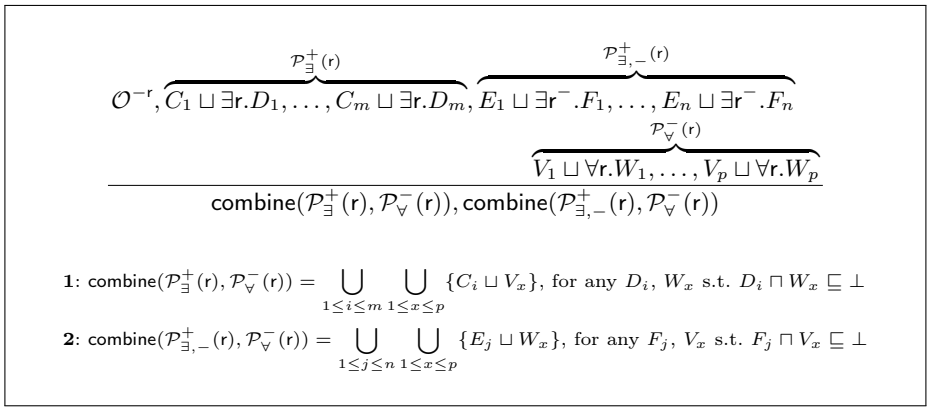


Fig. 2: The combination rule for eliminating a single role name $\mathsf{r} \in \mathsf{sig}_\mathsf{R}(\mathcal{O})$

Lemma 3 establishes the partial soundness of the calculus. In particular, the derived ontology $\mathcal{M}$ satisfies the first condition required for being a signature-restricted module of $\mathcal{O}$ w.r.t. Σ (Definition 6), where $\Sigma = \mathsf{sig}(\mathcal{O}) \setminus \{A\}$. However, $\mathcal{M}$ may contain definers that fall outside of Σ, potentially violating the second condition. We address these definers in a later section.

5.2 Single Role Name Elimination

The elimination of a single role name $r \in \mathsf{sig}_\mathsf{R}(\mathcal{O})$ from r-reduced ontologies is based on the *combination rule* in Figure 2. As with concept name elimination, this involves resolving positive clauses (where r occurs positively) with negative ones (where r occurs negatively) on the role name r to generate all entailments over $\mathsf{sig}(\mathcal{O}) \setminus \{r\}$. Because of Galois connections, r-clauses of the form $C \sqcup \forall r^-.D$ are excluded from the r-reduced form, yielding $2 \times 1 = 2$ combination cases.

Lemma 4. *Let $\mathcal{O}$ be an r-reduced $\mathcal{ALCI}$-ontology. If $\mathcal{M}$ is the ontology derived by applying the combination rule in Figure 2, then $\mathcal{O} \equiv_{\mathsf{sig}(O)\setminus\{r\}} \mathcal{M}$.*

An external DL reasoner [15] is employed to check the side conditions of the combination rules. Subsumption checking in $\mathcal{ALCI}$ is ExpTime-complete [53].

5.3 The Forgetting Process

As shown in Algorithm 1, our method takes as input an $\mathcal{ALCI}$-ontology $\mathcal{O}$ and a forgetting signature $\mathcal{F}$ comprising the concept and role names to be eliminated. The algorithm iteratively applies normalization and elimination calculus to each name in $\mathcal{F}$, producing intermediate results that may introduce definers. After processing $\mathcal{F}$, it attempts to eliminate any introduced definers, which are treated as regular concept names and eliminated using the same calculus.

Algorithm 1 The Forgetting Process

Require: An $\mathcal{ALCI}$-ontology $\mathcal{O}$, a forgetting signature $\mathcal{F} \subseteq \mathsf{sig}(\mathcal{O})$
Ensure: An $\mathcal{ALCI}$-ontology $\mathcal{V}$ with $\mathsf{sig}(\mathcal{V}) \cap \mathcal{F} = \emptyset$

Global: $\mathcal{D} \leftarrow \emptyset$ ▷ Track introduced definers globally
function FORGET($\mathcal{O}, \mathcal{F}$)
  $\mathcal{V} \leftarrow$ SIMPLIFY($\mathcal{O}$) ▷ Apply standard simplifications
  for each name $X \in \mathcal{F}$ **do**
    $\mathcal{O}_n \leftarrow$ NORMALIZE($\mathcal{V}, X$) ▷ Apply the normalization rules
    $\mathcal{V} \leftarrow$ ELIMINATE($\mathcal{O}_n, X$) ▷ Apply combination rules; updates global $\mathcal{D}$
  end for
  if $\mathcal{D} \neq \emptyset$ **then**
    for each definer $Z \in \mathcal{D}$ **do** ▷ Eliminate definers
      $\mathcal{V} \leftarrow$ ELIMINATE($\mathcal{V}, Z$) ▷ Apply combination rules; updates global $\mathcal{D}$
    end for
  end if
  return $\mathcal{V}$
end function

However, forgetting is not guaranteed to be complete for $\mathcal{ALCI}$ [38], as finite results may not always exist within the source logic. For ontologies with cyclic definitions (e.g., $\{B \sqcup A, \neg A \sqcup \exists r.A\}$), forgetting A would yield an infinite set of clauses ($\{B \sqcup \exists r.\top, B \sqcup \exists r.\exists r.\top, ...\}$) that cannot be finitely axiomatized within $\mathcal{ALCI}$. While fixpoints could handle such cycles [5], as implemented in LETHE and FAME, mainstream reasoners and the OWL API lack fixpoint support. Our method ensures finite representation of the result by ceasing attempts to forget the introduced definer and retaining it in the final result.

Note that cycles do not necessarily prevent finding finite forgetting results, as some names may be removed during the forgetting of other names, eliminating many cyclic dependencies. Our approach does not simply detect cycles, but employs a blocking technique that identifies when forgetting a definer produces a pattern syntactically identical to that of the previous intermediate result. This indicates the algorithm has entered a forgetting cycle where eliminating one definer introduces another that takes its place. In such cases, our method halts computation and declares forgetting unsuccessful.

Theorem 1. *For any $\mathcal{ALCI}$-ontology $\mathcal{O}$ and a forgetting signature $\mathcal{F} \subseteq \mathsf{sig}(\mathcal{O})$, our method always terminates and returns an $\mathcal{ALCI}$-ontology $\mathcal{M}$. If $\mathcal{M}$ is definer-free, then $\mathcal{M}$ is the result of forgetting $\mathcal{F}$ from $\mathcal{O}$.*

6 Empirical Evaluation

We have developed a prototype (Proto) of our method in Java using OWL API Version 5.1.7.[2] We benchmarked this prototype against two state-of-the-art for-

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

getting methods LETHE[3] and FAME[4], alongside several established subset modularization tools. The evaluation utilized two comprehensive corpora of real-world ontologies. The first corpus, sourced from the Oxford ISG repository,[5] comprises diverse ontologies from multiple domains. The second one consists of specialized biomedical ontologies from a March 2017 snapshot of NCBO BioPortal.[6]

From the Oxford ISG repository, we identified 488 ontologies containing at most 10,000 GCIs. To focus on non-trivial cases, we filtered out ontologies with cyclic dependencies or those lacking role restrictions and inverse roles, yielding 177 ontologies. These were then reduced to their $\mathcal{ALCI}$-fragments by removing GCIs not expressible within $\mathcal{ALCI}$. The same strategy applied to BioPortal produced 76 suitable ontologies. Complete statistical characterization of the refined corpora is provided in the extended version of this paper.

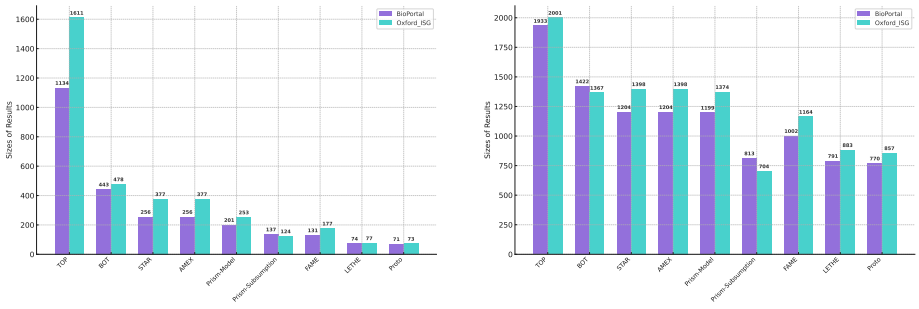
The composition of the forgetting signature $\mathcal{F}$ varies according to the specific requirements of different tasks. To address this variability, we designed two evaluation configurations to forget 90% and 50% of names in the signature of each ontology. A shuffling algorithm was used to ensure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

Notably, our prototype showcased complete effectiveness with a 100% success rate across all evaluation tracks, significantly outperforming LETHE (see Table 2 in the extended version). Specifically, on Oxford-ISG ontologies, LETHE achieved success rates of 70.03% and 82.04% when forgetting 90% and 50% of the Σ-names, respectively. On the BioPortal cases, LETHE's success rates were 69.99% and 80.97% 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 forgetting:

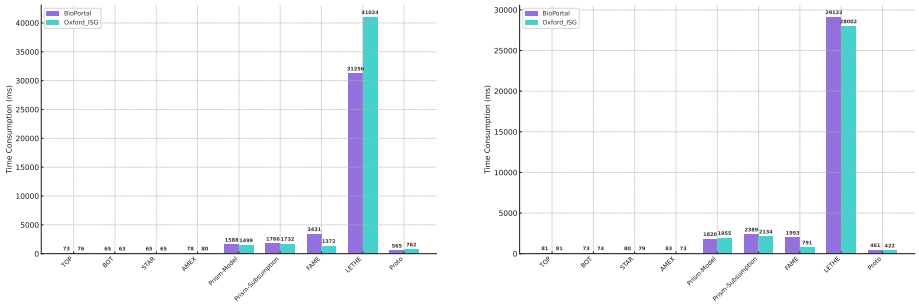
– First, the observed output sizes (Figure 3) contradict theoretical predictions of triple exponential growth. The forgetting results demonstrate remarkable

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



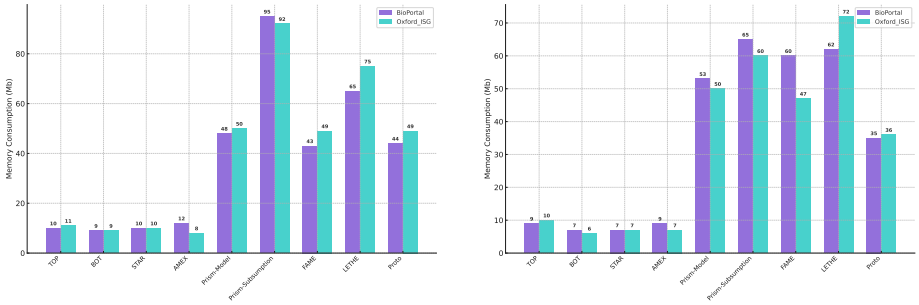
(a) Forgetting 90% of names in $\mathsf{sig}(\mathcal{O})$ (b) Forgetting 50% of names in $\mathsf{sig}(\mathcal{O})$

Fig. 3: Average |Onto| in output ontologies



(a) Forgetting 90% of names in $\mathsf{sig}(\mathcal{O})$ (b) Forgetting 50% of names in $\mathsf{sig}(\mathcal{O})$

Fig. 4: Average computation time



(a) Forgetting 90% of names in $\mathsf{sig}(\mathcal{O})$ (b) Forgetting 50% of names in $\mathsf{sig}(\mathcal{O})$

Fig. 5: Average memory usage

compactness, even outperforming modularization techniques that merely extract syntactic subsets. This finding challenges prior assessments [38,40] that dismissed forgetting as impractical due to its exponential space complexity.
– Second, our method's runtime performance (Figure 4) significantly exceeds Lethe and even matches the fastest modularization methods, despite forgetting's inherently higher computational complexity [57,3].
– 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 5), primarily through optimized definer introduction.

To understand if this efficiency comes from our novel linear definer introduction mechanism, we monitored the frequency of definers used in each forgetting attempt. The experiment results have confirmed our conjectures. In summary, on Oxford-ISG ontologies, when forgetting 90% and 50% of the signature names, Lethe required definers in 67.3% and 62.1% of the tasks respectively, while our method reduced these demands to 24.2% and 14.7%. For BioPortal cases, while Lethe introduced definers in 28.2% of the tasks across both settings, our method dramatically reduced this requirement to 9.7% and 2.9%, respectively. Detailed descriptions of these analyses are available in the extended version. This significant reduction in definer introduction — which necessitates additional computational steps for their subsequent elimination — explains our method's superior runtime and memory efficiency, rivaling even subset modularization approaches.

7 Conclusions and Future Work

Forgetting refers to a non-standard reasoning service that can be used 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 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.

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

Supplemental Material Statement: An extended version of this paper, containing all missing proofs, a couple of illustrative examples, more experimental results and the corresponding in-depth analyses, is available at `https://github.com/anonymous-ai-researcher/iswc2025`. The repository also includes our complete source code implementation and test datasets used in the evaluation.

References

1. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An Introduction to Description Logic. Cambridge University Press (2017)
2. Baader, F., Lutz, C., Suntisrivaraporn, B.: CEL - A polynomial-time reasoner for life science ontologies. In: Proc. IJCAR'06. Lecture Notes in Computer Science, vol. 4130, pp. 287–291. Springer (2006)
3. Botoeva, E., Konev, B., Lutz, C., Ryzhikov, V., Wolter, F., Zakharyaschev, M.: Inseparability and Conservative Extensions of Description Logic Ontologies: A Survey. In: Proc. RW'16. Lecture Notes in Computer Science, vol. 9885, pp. 27–89. Springer (2016)
4. Caldarola, E.G., Rinaldi, A.M.: An approach to ontology integration for ontology reuse. In: Proc. IRI'16. pp. 384–393. IEEE Computer Society (2016)
5. Calvanese, D., Giacomo, G.D., Lenzerini, M.: Reasoning in Expressive Description Logics with Fixpoints based on Automata on Infinite Trees. In: Proc. IJCAI'99. pp. 84–89. Morgan Kaufmann (1999)
6. Chen, J., Alghamdi, G., Schmidt, R.A., Walther, D., Gao, Y.: Ontology extraction for large ontologies via modularity and forgetting. In: K-CAP'19. pp. 45–52. ACM (2019)
7. Chen, J., Ludwig, M., Walther, D.: Computing minimal subsumption modules of ontologies. In: Proc. GCAI'18. EPiC Series in Computing, vol. 55, pp. 41–53. EasyChair (2018)
8. Craig, W.: Three uses of the herbrand-gentzen theorem in relating model theory and proof theory. J. Symb. Log. **22**(3), 269–285 (1957)
9. D'Agostino, G., Hollenberg, M.: Logical questions concerning the μ-calculus: Interpolation, lyndon and los-tarski. J. Symb. Log. **65**(1), 310–332 (2000)
10. Del-Pinto, W., Schmidt, R.A.: ABox Abduction via Forgetting in $\mathcal{ALC}$. In: Proc. AAAI'19. pp. 2768–2775. AAAI Press (2019)
11. Gabbay, D.M., Schmidt, R.A., Szalas, A.: Second-Order Quantifier Elimination - Foundations, Computational Aspects and Applications, Studies in logic : Mathematical logic and foundations, vol. 12. College Publications (2008)
12. Garson, J.W.: Modularity and relevant logic. Notre Dame J. Formal Log. **30**(2), 207–223 (1989)
13. Gatens, W., Konev, B., Wolter, F.: Lower and upper approximations for depleting modules of description logic ontologies. In: Proc. ECAI'14. Frontiers in Artificial Intelligence and Applications, vol. 263, pp. 345–350. IOS Press (2014)
14. Ghilardi, S., Lutz, C., Wolter, F.: Did I Damage My Ontology? A Case for Conservative Extensions in Description Logics. In: Proc. KR'06. pp. 187–197. AAAI Press (2006)
15. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: HermiT: An OWL 2 Reasoner. J. Autom. Reasoning **53**(3), 245–269 (2014)
16. Grau, B.C., Halaschek-Wiener, C., Kazakov, Y., Suntisrivaraporn, B.: Incremental classification of description logics ontologies. J. Autom. Reason. **44**(4), 337–369 (2010)
17. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: Just the right amount: extracting modules from ontologies. In: Proc. WWW'07. pp. 717–726. ACM (2007)
18. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. J. Artif. Intell. Res. **31**, 273–318 (2008)
19. Grau, B.C., Parsia, B., Sirin, E., Kalyanpur, A.: Modularity and web ontologies. In: Proc. KR'06. pp. 198–209. AAAI Press (2006)

20. Herzig, A., Mengin, J.: Uniform interpolation by resolution in modal logic. In: Proc. JELIA'08. Lecture Notes in Computer Science, vol. 5293, pp. 219–231. Springer (2008)
21. Hu, P., Motik, B., Horrocks, I.: Modular materialisation of datalog programs. Artif. Intell. **308**, 103726 (2022)
22. Klein, M.C.A., Fensel, D.: Ontology versioning on the Semantic Web. In: Proc. SWWS'01. pp. 75–91 (2001)
23. Klein, M.C.A., Fensel, D., Kiryakov, A., Ognyanov, D.: Ontology Versioning and Change Detection on the Web. In: Proc. EKAW'02. Lecture Notes in Computer Science, vol. 2473, pp. 197–212. Springer (2002)
24. Konev, B., Lutz, C., Walther, D., Wolter, F.: Semantic modularity and module extraction in description logics. In: Proc. ECAI'08. Frontiers in Artificial Intelligence and Applications, vol. 178, pp. 55–59. IOS Press (2008)
25. Konev, B., Lutz, C., Walther, D., Wolter, F.: Formal Properties of Modularisation. In: Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization, Lecture Notes in Computer Science, vol. 5445, pp. 25–66. Springer (2009)
26. Konev, B., Lutz, C., Walther, D., Wolter, F.: Model-theoretic inseparability and modularity of description logic ontologies. Artif. Intell. **203**, 66–103 (2013)
27. Konev, B., Walther, D., Wolter, F.: The Logical Difference Problem for Description Logic Terminologies. In: Proc. IJCAR'14. Lecture Notes in Computer Science, vol. 5195, pp. 259–274. Springer (2008)
28. Konev, B., Walther, D., Wolter, F.: Forgetting and Uniform Interpolation in Large-Scale Description Logic Terminologies. In: Proc. IJCAI'09. pp. 830–835. IJCAI/AAAI Press (2009)
29. Kontchakov, R., Wolter, F., Zakharyaschev, M.: Logic-based ontology comparison and module extraction, with an application to dl-lite. Artif. Intell. **174**(15), 1093–1141 (2010)
30. Koopmann, P.: Practical Uniform Interpolation for Expressive Description Logics. Ph.D. thesis, The University of Manchester, UK (2015)
31. Koopmann, P., Chen, J.: Deductive module extraction for expressive description logics. In: Bessiere, C. (ed.) IJCAI'20. pp. 1636–1643. ijcai.org (2020)
32. Koopmann, P., Del-Pinto, W., Tourret, S., Schmidt, R.A.: Signature-Based Abduction for Expressive Description Logics. In: Proc. KR'20. pp. 592–602 (2020)
33. Koopmann, P., Schmidt, R.A.: Uniform interpolation of $\mathcal{ALC}$-ontologies using fixpoints. In: Proc. FroCoS'13. Lecture Notes in Computer Science, vol. 8152, pp. 87–102. Springer (2013)
34. Lin, F., Reiter, R.: Forget It! In: Proc. AAAI Fall Symposium on Relevance. pp. 154–159. AAAI Press (1994)
35. Ludwig, M., Konev, B.: Practical Uniform Interpolation and Forgetting for $\mathcal{ALC}$ TBoxes with Applications to Logical Difference. In: Proc. KR'14. pp. 318–327. AAAI Press (2014)
36. Lutz, C., Seylan, I., Wolter, F.: An automata-theoretic approach to uniform interpolation and approximation in the description logic EL. In: Proc. KR'12. pp. 286–296. AAAI Press (2012)
37. Lutz, C., Wolter, F.: Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. J. Symb. Comput. **45**(2), 194–228 (2010)
38. Lutz, C., Wolter, F.: Foundations for Uniform Interpolation and Forgetting in Expressive Description Logics. In: Proc. IJCAI'11. pp. 989–995. IJCAI/AAAI Press (2011)

39. Nikitina, N., Glimm, B.: Hitting the sweetspot: Economic rewriting of knowledge bases. In: Proc. ISWC'12. vol. 7649, pp. 394–409. Springer (2012)
40. Nikitina, N., Rudolph, S.: (Non-)Succinctness of uniform interpolants of general terminologies in the description logic $\mathcal{EL}$. Artif. Intell. **215**, 120–140 (2014)
41. Nikitina, N., Rudolph, S., Glimm, B.: Reasoning-supported interactive revision of knowledge bases. In: Proc. IJCAI'11. pp. 1027–1032. IJCAI/AAAI (2011)
42. Nortje, R., Britz, K., Meyer, T.: Reachability modules for the description logic $\mathcal{SRIQ}$. In: Proc. LPAR'19. Lecture Notes in Computer Science, vol. 8312, pp. 636–652. Springer (2013)
43. Noy, N.F., Musen, M.A.: PROMPTDIFF: A Fixed-Point Algorithm for Comparing Ontology Versions. In: Proc. AAAI/IAAI'02. pp. 744–750. AAAI/MIT Press (2002)
44. Noy, N.F., Musen, M.A.: The PROMPT suite: interactive tools for ontology merging and mapping. Int. J. Hum. Comput. Stud. **59**(6), 983–1024 (2003)
45. Pinto, H.S.A.N.P., Martins, J.P.: A methodology for ontology integration. In: Proc. K-CAP'01. pp. 131–138. ACM (2001)
46. Ribeiro, M.M., Wassermann, R.: Base revision for ontology debugging. J. Log. Comput. **19**(5), 721–743 (2009)
47. Robinson, J.A.: A machine-oriented logic based on the resolution principle. J. ACM **12**(1), 23–41 (1965)
48. Romero, A.A.: Ontology module extraction and applications to ontology classification. Ph.D. thesis, University of Oxford, UK (2015)
49. Romero, A.A., Kaminski, M., Grau, B.C., Horrocks, I.: Module extraction in expressive ontology languages via datalog reasoning. J. Artif. Intell. Res. **55**, 499–564 (2016)
50. Schrimpsher, D., Wu, Z., Orme, A.M., Etzkorn, L.H.: Dynamic ontology version control. In: Proc. ACMse'10. p. 25. ACM (2010)
51. Simperl, E.: Reusing ontologies on the semantic web: A feasibility study. Data Knowl. Eng. **68**(10), 905–925 (2009)
52. Stuckenschmidt, H., Klein, M.C.A.: Reasoning and change management in modular ontologies. Data Knowl. Eng. **63**(2), 200–223 (2007)
53. Tobies, S.: Complexity results and practical algorithms for logics in knowledge representation. Ph.D. thesis, RWTH Aachen University, Germany (2001)
54. Troquard, N., Confalonieri, R., Galliani, P., Peñaloza, R., Porello, D., Kutz, O.: Repairing Ontologies via Axiom Weakening. In: Proc. AAAI'18. pp. 1981–1988. AAAI Press (2018)
55. Tsarkov, D.: Incremental and persistent reasoning in fact++. In: Proc. ORE'14'. CEUR Workshop Proceedings, vol. 1207, pp. 16–22. CEUR-WS.org (2014)
56. Tun, N.N., Dong, J.S.: Ontology generation through the fusion of partial reuse and relation extraction. In: Proc. KR'08. pp. 318–328. AAAI Press (2008)
57. Vescovo, C.D., Horridge, M., Parsia, B., Sattler, U., Schneider, T., Zhao, H.: Modular structures and atomic decomposition in ontologies. J. Artif. Intell. Res. **69**, 963–1021 (2020)
58. Visser, A.: Bisimulations, Model Descriptions and Propositional Quantifiers. Logic Group Preprint Series, Utrecht University (1996)
59. Wang, K., Antoniou, G., Topor, R., Sattar, A.: Merging and aligning ontologies in dl-programs. In: Proc. RuleML'05. Lecture Notes in Computer Science, vol. 3791, pp. 160–171. Springer (2005)
60. Wang, K., Wang, Z., Topor, R.W., Pan, J.Z., Antoniou, G.: Eliminating Concepts and Roles from Ontologies in Expressive Descriptive Logics. Computational Intelligence **30**(2), 205–232 (2014)

61. Xiang, Y., Wu, X., Lu, C., Zhao, Y.: Creating signature-based views for description logic ontologies with transitivity and qualified number restrictions. In: Proc. WWW'22. pp. 808–817. ACM (2022)
62. Yang, H., Koopmann, P., Ma, Y., Bidoit, N.: Efficient Computation of General Modules for $\mathcal{ALC}$ Ontologies. In: Proc. IJCAI'23. pp. 3356–3364. ijcai.org (2023)
63. Zhang, Y., Zhou, Y.: Forgetting revisited. In: KR. AAAI Press (2010)
64. Zhao, Y.: Automated Semantic Forgetting for Expressive Description Logics. Ph.D. thesis, The University of Manchester, UK (2018)
65. Zhao, Y., A., S.R.: Role forgetting for $\mathcal{ALCOQH}(\triangledown)$-ontologies using an ackermann-based approach. In: Proc. IJCAI'17. pp. 1354–1361. IJCAI/AAAI Press (2017)
66. Zhao, Y., Alghamdi, G., A., S.R., Feng, H., Stoilos, G., Juric, D., Khodadadi, M.: Tracking Logical Difference in Large-Scale Ontologies: A Forgetting-Based Approach. In: Proc. AAAI'19. pp. 3116–3124. AAAI Press (2019)
67. Zhao, Y., Schmidt, R.A.: Forgetting Concept and Role Symbols in $\mathcal{ALCOIH}\mu^+(\triangledown, \sqcap)$-Ontologies. In: Proc. IJCAI'16. pp. 1345–1352. IJCAI/AAAI Press (2016)
68. Zhao, Y., Schmidt, R.A.: FAME: an automated tool for semantic forgetting in expressive description logics. In: Proc. IJCAR'18. Lecture Notes in Computer Science, vol. 10900, pp. 19–27. Springer (2018)

A Additional Related Work — Subset Module Extraction

Subset modules, like forgetting-based modules, can be characterized from both a model-theoretic and a deductive perspective, giving rise to *model subset modules* and *subsumption subset modules*, respectively. These notions have been further refined into three commonly studied variants: *plain*, *self-contained*, and *depleting* modules. Plain modules capture the core idea of modularity as characterized by Definition 4, whereas self-contained and depleting modules impose additional structural constraints, ensuring, for instance, that the extracted module forms a logically independent or non-redundant fragment of the original ontology.

Definition 10 (Self-Contained Module for $\mathcal{ALCI}$). *Let $\mathcal{O}$ and $\mathcal{M}$ be $\mathcal{ALCI}$-ontologies and $\Sigma \subseteq \mathsf{sig}(\mathcal{O})$ a signature of concept and role names. $\mathcal{M}$ is a* self-contained module *of $\mathcal{O}$ w.r.t. Σ iff the following conditions hold: (i) $\mathcal{M}$ is a subset module of $\mathcal{O}$, and (ii) $\mathcal{M} \equiv_{\Sigma \cup \mathsf{sig}(\mathcal{M})} \mathcal{O}$.*

As shown by the second condition above, a self-contained module $\mathcal{M}$ must not only preserve all logical entailments over Σ, but must also maintain all logical entailments over the combined signature $\Sigma \cup \mathsf{sig}(\mathcal{M})$. This constitutes a stringent requirement, particularly since $\mathcal{M}$ necessarily incorporates names from outside Σ to preserve the semantic equivalence for the names within Σ.

Definition 11 (Depleting Module for $\mathcal{ALCI}$). *$\mathcal{M}$ is a* depleting module *of $\mathcal{O}$ w.r.t. Σ iff the following conditions hold: (i) $\mathcal{M}$ is a subset module of $\mathcal{O}$, and (ii) $\mathcal{O} \setminus \mathcal{M} \equiv_{\Sigma \cup \mathsf{sig}(\mathcal{M})} \emptyset$.*

Beyond self-contained modules, a depleting module $\mathcal{M}$ imposes an even more stringent restriction: the remaining axioms in $\mathcal{O}$ (i.e., $\mathcal{O} \setminus \mathcal{M}$) must be completely independent of $\mathcal{M}$, containing no logical entailments involving any names in $\mathsf{sig}(\mathcal{M})$. These requirements arise from the practical demands of effective knowledge reuse in ontology engineering.

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.

Several approaches for ontology modularization have been developed based on these foundational notions, each with its own strengths and limitations. These methods vary in their underlying algorithms, theoretical guarantees, and supported DLs. In the following, we render a systematic account of these approaches, examining their technical underpinnings and comparative advantages.

MEX[7] implements a polynomial-time algorithm for extracting minimal, self-contained, and depleting modules from acyclic $\mathcal{EL}$- and $\mathcal{ELI}$-terminologies [24]. Within the same work, this approach was adapted into a Π_2^p algorithm for acyclic $\mathcal{ALCI}$-terminologies that satisfy a natural syntactic condition [24], and was subsequently extended to DL-Lite ontologies [29]. The foundational work by Konev et al. [26] established the theoretical underpinnings for model-theoretic inseparability and modularity, providing rigorous semantic characterizations that guarantee strong module properties. AMEX[8] further extends this approach to acyclic $\mathcal{ALCQI}$-ontologies, preserving the self-containment and depletingness properties while necessarily sacrificing computational efficiency [13].

Locality-based methods (LBMs), which have been implemented within the OWL API,[9] extract self-contained, depleting modules from $\mathcal{SROIQ}$-ontologies in polynomial time [18]. LBMs comprise three variants based on their GCI locality testing [29]: $\bot$ locality (BOT), $\top$ locality (TOP), and their nested variant, $\bot\top^*$ locality (STAR). The syntactic nature of these approaches facilitates efficient implementation while maintaining strong theoretical guarantees for the extracted modules. LBMs have given rise to two significant extensions: reachability-based methods (RBMs) and Datalog-based methods (DBMs). RBMs, developed for $\mathcal{SRIQ}$-ontologies [42], introduce a novel module type motivated by the optimization techniques already employed in the DL reasoner CEL [2]. These modules come in $\bot$, $\top$, and $\bot\top^*$ variants. While $\bot$-reachability corresponds to $\bot$-locality for ontologies in certain normal forms, other reachability variants typically produce more compact modules than their LBM counterparts but sacrifice self-containment and depletingness, thereby failing to preserve all Σ-related information. DBMs [49,21], implemented in PrisM[10] [48], preserve self-containment and depletingness while extracting modules from $\mathcal{SROIQ}$-ontologies in polynomial time. DBMs constitute a substantial advancement by reformulating mod-

[7] https://cgi.csc.liv.ac.uk/ konev/software/
[8] https://github.com/sadger/module-extraction
[9] http://owlapi.sourceforge.net
[10] https://github.com/anaphylactic/PrisM

ule extraction as a Datalog reasoning problem, thus leveraging highly optimized Datalog engines for efficient computation.

While MEX computes exact solutions, all other approaches focus on deriving approximations of the solutions [3]. These approximations offer a pragmatic balance between semantic completeness and computational tractability, particularly for highly expressive ontologies. Subsumption modules, despite their theoretical significance, remain underexplored primarily because their deductive characteristics fail to guarantee *robustness under replacement* — a critical property for safely reusing modules across different contexts [25,26]. This robustness property ensures that modules can be reliably substituted for their source ontologies in various reasoning scenarios without affecting reasoning results. To the best of our knowledge, the methods proposed by [7] and [31] constitute the only approaches for extracting subsumption modules, but their implementations are not publicly available. Therefore, we compare our approach against the following established baselines: TOP, BOT, STAR, AMEX, and PrisM, excluding MEX as it supports only $\mathcal{ELI}$ but not $\mathcal{ALCI}$.

B Missing Proofs

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

Definition 12 (Conservative Extension). *Given two ontologies $\mathcal{O}$ and $\mathcal{O}'$, we say that $\mathcal{O}'$ is a* conservative extension *of $\mathcal{O}$ if*

1. $\mathsf{sig}(\mathcal{O}) \subseteq \mathsf{sig}(\mathcal{O}')$,
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 $\mathsf{sig}(\mathcal{O})$ coincide in $\mathcal{I}$ and $\mathcal{I}'$, i.e.,*
   - $A^{\mathcal{I}} = A^{\mathcal{I}'}$ *for all concept names $A \in \mathsf{sig}(\mathcal{O})$, and*
   - $r^{\mathcal{I}} = r^{\mathcal{I}'}$ *for all role names $r \in \mathsf{sig}(\mathcal{O})$.*

Conservative extension has two key properties:

- *Transitivity*: If $\mathcal{O}''$ is a conservative extension of $\mathcal{O}'$ and $\mathcal{O}'$ is a conservative extension of $\mathcal{O}$, then $\mathcal{O}''$ is a conservative extension of $\mathcal{O}$; this is quite obvious.
- *Subsumption Preservation*: If $\mathcal{O}'$ is a conservative extension of $\mathcal{O}$, then for all clauses constructed using only the names from $\mathsf{sig}(\mathcal{O})$, subsumption w.r.t. $\mathcal{O}$ and $\mathcal{O}'$ coincide; see Proposition 1.

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

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

Proof. We prove both directions by contrapositive:
($\Rightarrow$) 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 $\mathsf{sig}(\mathcal{O})$ coincide in $\mathcal{I}$ and $\mathcal{I}'$. Since $L_1 \sqcup \ldots \sqcup L_n$ contains only names from $\mathsf{sig}(\mathcal{O})$, 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. *Let $\mathcal{O}$ be an $\mathcal{ALCI}$-ontology and $\mathcal{O}'$ its reduced form obtained using the normalization rules described above. Then $\mathcal{O} \equiv_{\mathsf{sig}(\mathcal{O})} \mathcal{O}'$.*

Proof. The proof establishes that each normalization rule yields a conservative extension, and by transitivity of conservative extensions, the complete reduction process preserves logical equivalence over the original signature.

We detail this for the first rule; similar proofs apply to the other rules. Consider the application of the first rule to an A-clause instance $L_1 \sqcup \cdots \sqcup L_n$ where A appears more than once. Suppose there exists a literal L_i ($1 \leq i \leq n$) of the form $\exists R.C$ or $\forall R.C$ with $A \in \mathsf{sig}(C)$. Let $\mathcal{O}'$ be the ontology obtained by replacing C with a fresh definer $Z \in \mathsf{N_C}$ and adding the clause $\neg Z \sqcup C$ to the present ontology. Thus, $\mathcal{O}'$ contains the modified clause $(L_1 \sqcup \cdots \sqcup L_n)^{C \mapsto Z}$ where C is replaced by Z, together with the defining clause $\neg Z \sqcup C$.

The signature condition holds since $\mathsf{sig}(\mathcal{O}') = \mathsf{sig}(\mathcal{O}) \cup \{Z\}$ where Z is fresh, ensuring $\mathsf{sig}(\mathcal{O}) \subseteq \mathsf{sig}(\mathcal{O}')$.

For the semantic direction from $\mathcal{O}'$ to $\mathcal{O}$, we establish that every model of $\mathcal{O}'$ is a model of $\mathcal{O}$. Suppose, toward a contradiction, that there exists a model $\mathcal{I}'$ of $\mathcal{O}'$ such that $\mathcal{I}'$ is not a model of $\mathcal{O}$. This means $\mathcal{I}' \not\models L_1 \sqcup \cdots \sqcup L_n$. Therefore, there exists a domain element $d \in \Delta^{\mathcal{I}'}$ such that $d \notin (L_i)^{\mathcal{I}'}$ for all literals L_i ($1 \leq i \leq n$) in the original clause. Without loss of generality, assume the literal containing the substituted concept is $L_i = \exists R.C$, so the transformed literal is $\exists R.Z$. Since $\mathcal{I}'$ is a model of $\mathcal{O}'$, it satisfies $\top \sqsubseteq (L_1 \sqcup \cdots \sqcup L_n)^{C \mapsto Z}$, which means $d \in (L_1 \sqcup \cdots \sqcup L_n)^{C \mapsto Z^{\mathcal{I}'}}$. Given that d does not satisfy any of the other literals L_j for $j \neq i$, we must have $d \in (\exists R.Z)^{\mathcal{I}'}$. This implies there exists an element $d' \in \Delta^{\mathcal{I}'}$ such that $(d, d') \in R^{\mathcal{I}'}$ and $d' \in Z^{\mathcal{I}'}$.

Since $\mathcal{I}'$ satisfies the defining clause $Z \sqsubseteq C$, we have $Z^{\mathcal{I}'} \subseteq C^{\mathcal{I}'}$, which implies $d' \in C^{\mathcal{I}'}$. Therefore, $(d, d') \in R^{\mathcal{I}'}$ and $d' \in C^{\mathcal{I}'}$, yielding $d \in (\exists R.C)^{\mathcal{I}'}$. This contradicts our assumption that d does not satisfy the original literal $L_i = \exists R.C$. The case where $L_i = \forall R.C$ follows by analogous reasoning. Therefore, the reduct of $\mathcal{I}'$ to $\mathsf{sig}(\mathcal{O})$ must be a model of $\mathcal{O}$.

Conversely, let $\mathcal{I}$ be any model of $\mathcal{O}$, satisfying $\top^{\mathcal{I}} \subseteq (L_1 \sqcup \cdots \sqcup L_n)^{\mathcal{I}}$. We construct an expansion $\mathcal{I}'$ that agrees with $\mathcal{I}$ on all symbols in $\mathsf{sig}(\mathcal{O})$ and interprets the fresh concept name as $Z^{\mathcal{I}'} = C^{\mathcal{I}}$. Since $\top^{\mathcal{I}} \subseteq (L_1 \sqcup \cdots \sqcup L_n)^{\mathcal{I}}$ and Z does not occur in the original clause or in C, we have $\top^{\mathcal{I}'} \subseteq (L_1 \sqcup \cdots \sqcup L_n)^{\mathcal{I}'}$. Moreover, by construction, $Z^{\mathcal{I}'} = C^{\mathcal{I}} = C^{\mathcal{I}'}$, which ensures satisfaction of the

defining axiom $\neg Z \sqcup C$. The substitution property guarantees that $\top^{\mathcal{I}'} \subseteq ((L_1 \sqcup \cdots \sqcup L_n)^{C \mapsto Z})^{\mathcal{I}'}$, establishing that $\mathcal{I}'$ is a model of $\mathcal{O}'$.

This bidirectional correspondence between models restricted to the original signature demonstrates that $\mathcal{O} \equiv_{\mathsf{sig}(\mathcal{O})} \mathcal{O}'$ under a single rule application. Since the complete normalization process involves a finite sequence of such applications, and conservative extension is preserved under composition, the lemma follows.

For the semantic direction from $\mathcal{O}'$ to $\mathcal{O}$, let $\mathcal{I}'$ be any model of $\mathcal{O}'$. Then $\mathcal{I}'$ satisfies both $\top \sqsubseteq (L_1 \sqcup \cdots \sqcup L_n)[C/Z]$ and $Z \sqsubseteq C$. Since Z occurs only in positive positions within $(L_1 \sqcup \cdots \sqcup L_n)[C/Z]$, and given that $Z^{\mathcal{I}'} \subseteq C^{\mathcal{I}'}$, monotonicity of concept constructors ensures that $\top^{\mathcal{I}'} \subseteq (L_1 \sqcup \cdots \sqcup L_n)^{\mathcal{I}'}$. Therefore, the restriction of $\mathcal{I}'$ to $\mathsf{sig}(\mathcal{O})$ satisfies the original axiom, making it a model of $\mathcal{O}$.

For the converse direction, let $\mathcal{I}$ be a model of $\mathcal{O}$, so $\top^{\mathcal{I}} \subseteq (L_1 \sqcup \cdots \sqcup L_n)^{\mathcal{I}}$. We construct an expansion $\mathcal{I}'$ of $\mathcal{I}$ to $\mathsf{sig}(\mathcal{O}')$ by setting $Z^{\mathcal{I}'} = C^{\mathcal{I}}$ and maintaining all other interpretations unchanged. Since Z does not appear in the original formula $L_1 \sqcup \cdots \sqcup L_n$ or in C, we have $(L_1 \sqcup \cdots \sqcup L_n)^{\mathcal{I}} = (L_1 \sqcup \cdots \sqcup L_n)[C/Z]^{\mathcal{I}'}$. Thus $\top^{\mathcal{I}'} \subseteq (L_1 \sqcup \cdots \sqcup L_n)[C/Z]^{\mathcal{I}'}$, and by construction $Z^{\mathcal{I}'} = C^{\mathcal{I}} = C^{\mathcal{I}'}$, so $Z^{\mathcal{I}'} \subseteq C^{\mathcal{I}'}$. Therefore $\mathcal{I}'$ is a model of $\mathcal{O}'$.

This establishes that $\mathcal{O}'$ is a conservative extension of $\mathcal{O}$ over $\mathsf{sig}(\mathcal{O})$. Since each normalization rule produces a conservative extension, and conservative extensions are transitive, the complete normalization process yields $\mathcal{O} \equiv_{\mathsf{sig}(\mathcal{O})} \mathcal{O}'$.

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. $\mathsf{sig}(\mathcal{O}') = \mathsf{sig}(\mathcal{O}) \cup \{Z\}$, so $\mathsf{sig}(\mathcal{O}) \subseteq \mathsf{sig}(\mathcal{O}')$.
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 [64], 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.

Lemma 2. *For any $\mathcal{ALCI}$-ontology $\mathcal{O}$, a reduced form $\mathcal{O}'$ (either A-reduced or r-reduced) can be computed via a linear number of normalization steps. Moreover, $|\mathcal{O}'|$ is linear in $|\mathcal{O}|$, where $|\mathcal{O}|$ denotes the number of clauses in $\mathcal{O}$.*

To prove Lemma 3, we first introduce the following notions:

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

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

We define the *frequency* $\mathsf{fq}(\mathsf{A}, L_1 \sqcup \ldots \sqcup L_n)$ of A in an 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 $D \star \mathcal{Q}R.E$, where $\star \in \{\sqcap, \sqcup\}$, $\mathcal{Q} \in \{\exists, \forall\}$, 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. Similarly, $\mathsf{dp}(\mathsf{r}^*, C)$ counts the number of role restrictions guarding a role occurrence r^* 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 $\mathcal{Q} \in \{\exists, \forall\}$, we have $\mathsf{dp}(\mathsf{A}, \mathcal{Q}R.C) = n + 1$ 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: $\mathsf{dp}(\mathsf{A}, \mathsf{NLZ}_{k-1}(\mathcal{O})) - \mathsf{dp}(\mathsf{A}, \mathsf{NLZ}_k(\mathcal{O})) = 1$.

Lemma 3. *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. Each definer replaces a subconcept immediately below an $\exists$-restriction and triggers one new clause. Therefore, both the number of definers and new clauses introduced in $\mathcal{O}$'s normalization are bounded by $\mathsf{O}(n)$, where n is the count of $\exists$-restrictions in $\mathcal{O}$. This ensures termination and completeness.

Lemma 4 (Soundness of rule). *Let $\mathcal{O}$ be an A-reduced $\mathcal{ALCI}$-ontology with $A \in \mathsf{sig}_C(\mathcal{O})$. If $\mathcal{M}$ is the ontology derived by applying the combination rule in Figure 1, then $\mathcal{O} \equiv_{\mathsf{sig}(O)\setminus\{A\}} \mathcal{M}$.*

Proof. Cases 1, 2, 3, 4, 5, 9, and 13 reverse the normalization process, thereby directly preserving inseparability. We now focus on proving Case 8 in detail, as the other cases follow similar proof patterns. We demonstrate that the premises $\mathcal{O}$ of the case constitutes a conservative extension of its conclusion $\mathcal{O}^{-\mathsf{A}}$. First, observe that $\mathsf{sig}(\mathcal{O}) = \mathsf{sig}(\mathcal{O}^{-\mathsf{A}}) \cup \mathsf{A}$, which establishes $\mathsf{sig}(\mathcal{O}^{-\mathsf{A}}) \subseteq \mathsf{sig}(\mathcal{O})$, satisfying Condition (1) of Definition 12.

Next we prove Condition (2) of Definition 12. Assume for the sake of contradiction that there exists an interpretation $\mathcal{I}$ such that:

$$\mathcal{I} \models \top \sqsubseteq C_i \sqcup \exists r_i.A \tag{P1}$$
$$\mathcal{I} \models \top \sqsubseteq \psi_{k'} \sqcup \forall w_{k'}.\neg A \tag{P2}$$
$$\mathcal{I} \not\models \top \sqsubseteq C_i \sqcup \exists r_i.\forall w_{k'}^{-}.\psi_{k'} \tag{C1}$$

From (C1), there exists some $d \in \Delta^{\mathcal{I}}$ such that:

$$d \notin (C_i \sqcup \exists r_i.\forall w_{k'}^{-}.\psi_{k'})^{\mathcal{I}}$$

This means:

$$d \notin C_i^{\mathcal{I}} \tag{1}$$
$$d \notin (\exists r_i.\forall w_{k'}^{-}.\psi_{k'})^{\mathcal{I}} \tag{2}$$

From (P1) and the fact that $d \in \Delta^{\mathcal{I}} = \top^{\mathcal{I}}$, we have:

$$d \in (C_i \sqcup \exists r_i.A)^{\mathcal{I}} = C_i^{\mathcal{I}} \cup (\exists r_i.A)^{\mathcal{I}}$$

Since $d \notin C_i^{\mathcal{I}}$ by (1), we must have:

$$d \in (\exists r_i.A)^{\mathcal{I}}$$

Therefore, there exists some $e \in \Delta^{\mathcal{I}}$ such that:

$$(d, e) \in r_i^{\mathcal{I}} \tag{3}$$
$$e \in A^{\mathcal{I}} \tag{4}$$

From (2), we know that $d \notin (\exists r_i.\forall w_{k'}^{-}.\psi_{k'})^{\mathcal{I}}$, and therefore $d \in (\forall r_i.\exists w_{k'}^{-}.\neg\psi_{k'})^{\mathcal{I}}$. Since $(d, e) \in r_i^{\mathcal{I}}$ by (3), we have:

$$e \in (\exists w_{k'}^{-}.\neg\psi_{k'})^{\mathcal{I}}$$

Therefore, there exists some $f \in \Delta^{\mathcal{I}}$ such that:

$$(e, f) \in (w_{k'}^{-})^{\mathcal{I}} \tag{5}$$
$$f \in (\neg\psi_{k'})^{\mathcal{I}} \tag{6}$$

From (5) and the definition of inverse roles:

$$(f, e) \in w_{k'}^{\mathcal{I}} \tag{7}$$

From (P2) and the fact that $f \in \Delta^{\mathcal{I}} = \top^{\mathcal{I}}$, we have:

$$f \in (\psi_{k'} \sqcup \forall w_{k'}.\neg A)^{\mathcal{I}} = \psi_{k'}^{\mathcal{I}} \cup (\forall w_{k'}.\neg A)^{\mathcal{I}}$$

Since $f \in (\neg\psi_{k'})^{\mathcal{I}}$ by (6), we have $f \notin \psi_{k'}^{\mathcal{I}}$. Therefore, we must have:

$$f \in (\forall w_{k'}.\neg A)^{\mathcal{I}}$$

Since $(f, e) \in w_{k'}^{\mathcal{I}}$ by (7), we have:

$$e \in (\neg A)^{\mathcal{I}}$$

But this contradicts (4), which states that $e \in A^{\mathcal{I}}$. Therefore, our assumption was false, and the entailment holds.

Finally we prove Condition (3) of Definition 12.

$$\begin{aligned}
\alpha &: \top \sqsubseteq C_i \sqcup \exists r_i.\forall w_{k'}^{-}.\psi_{k'} \\
\beta_1 &: \top \sqsubseteq C_i \sqcup \exists r_i.A \\
\beta_2 &: \top \sqsubseteq \psi_{k'} \sqcup \forall w_{k'}.\neg A
\end{aligned}$$

We need to prove that for every model $\mathcal{I}$ of α, there exists a model $\mathcal{I}'$ of $\{\beta_1, \beta_2\}$ such that the interpretations of concept and role names from $\mathsf{sig}(\alpha)$ coincide in $\mathcal{I}$ and $\mathcal{I}'$. We construct $\mathcal{I}'$ as follows ($A \notin \mathsf{sig}(\alpha)$ since A does not appear in α.):

- $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}}$
- For all concept names $X \in \mathsf{sig}(\alpha)$: $X^{\mathcal{I}'} = X^{\mathcal{I}}$
- For all role names $r \in \mathsf{sig}(\alpha)$: $r^{\mathcal{I}'} = r^{\mathcal{I}}$
- $A^{\mathcal{I}'} = (\forall w_{k'}^{-}.\psi_{k'})^{\mathcal{I}}$

Verification that $\mathcal{I}'$ satisfies β_1:
Since $\mathcal{I} \models \alpha$, we have:

$$\mathcal{I} \models \top \sqsubseteq C_i \sqcup \exists r_i.\forall w_{k'}^{-}.\psi_{k'}$$

By our construction, $\mathcal{I}'$ coincides with $\mathcal{I}$ on all names appearing in this GCI. Therefore:

$$\mathcal{I}' \models \top \sqsubseteq C_i \sqcup \exists r_i.\forall w_{k'}^{-}.\psi_{k'}$$

Since $A^{\mathcal{I}'} = (\forall w_{k'}^{-}.\psi_{k'})^{\mathcal{I}} = (\forall w_{k'}^{-}.\psi_{k'})^{\mathcal{I}'}$, we have:

$$\mathcal{I}' \models \top \sqsubseteq C_i \sqcup \exists r_i.A$$

Verification that $\mathcal{I}'$ satisfies β_2:
We need to show that $\mathcal{I}' \models \top \sqsubseteq \psi_{k'} \sqcup \forall w_{k'}.\neg A$.

Since $A^{\mathcal{I}'} = (\forall w_{k'}^{-}.\psi_{k'})^{\mathcal{I}'}$, this is equivalent to showing:

$$\mathcal{I}' \models \top \sqsubseteq \psi_{k'} \sqcup \forall w_{k'}.\neg(\forall w_{k'}^{-}.\psi_{k'})$$

We proceed by contradiction. Assume that $\mathcal{I}'$ does not satisfy this GCI. Then there exists some $d \in \Delta^{\mathcal{I}'}$ such that:

$$d \notin (\psi_{k'} \sqcup \forall w_{k'}.\neg(\forall w_{k'}^{-}.\psi_{k'}))^{\mathcal{I}'}$$

This means:

$$d \notin \psi_{k'}^{\mathcal{I}'} \tag{8}$$

$$d \notin (\forall w_{k'}.\neg(\forall w_{k'}^{-}.\psi_{k'}))^{\mathcal{I}'} \tag{9}$$

From (9), we have:

$$d \in (\exists w_{k'}.(\forall w_{k'}^{-}.\psi_{k'}))^{\mathcal{I}'}$$

Therefore, there exists some $e \in \Delta^{\mathcal{I}'}$ such that:

$$(d, e) \in w_{k'}^{\mathcal{I}'} \tag{10}$$

$$e \in (\forall w_{k'}^{-}.\psi_{k'})^{\mathcal{I}'} \tag{11}$$

From (10) and the definition of inverse roles:

$$(e, d) \in (w_{k'}^{-})^{\mathcal{I}'} \tag{12}$$

From (11) and (12), we have:

$$d \in \psi_{k'}^{\mathcal{I}'} \tag{13}$$

But (13) contradicts (8). Therefore, our assumption was false, and $\mathcal{I}' \models \beta_2$. Using the transitivity of conservative extension and Proposition 1, we conclude that $\mathcal{O}$ is a conservative extension of $\mathcal{M}$.

Likewise, Lemma 5 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 $\mathsf{sig}(O) \setminus \{\mathsf{r}\}$, potentially failing to fulfill the first condition.

Lemma 5 (Soundness of rule). *Let $\mathcal{O}$ be an r-reduced $\mathcal{ALCI}$-ontology with $r \in \mathsf{sig}_R(\mathcal{O})$. If $\mathcal{M}$ is the ontology derived by applying the combination rule in Figure 2, then $\mathcal{O} \equiv_{\mathsf{sig}(O)\setminus\{r\}} \mathcal{M}$.*

Proof. For the 1st combination case (with the other cases following similar reasoning), we prove that premises $\mathcal{O}$ constitutes a conservative extension of conclusion $\mathcal{O}^{-\mathsf{r}}$. Observe that $\mathsf{sig}(\mathcal{O}) = \mathsf{sig}(\mathcal{O}^{-\mathsf{r}}) \cup \mathsf{r}$, establishing $\mathsf{sig}(\mathcal{O}^{-\mathsf{r}}) \subseteq \mathsf{sig}(\mathcal{O})$ and thereby satisfying Condition (1) of Definition 7.

We demonstrate that $\mathcal{O}$ is a conservative extension of $\mathcal{O}^{-\mathsf{r}}$. Since $\mathsf{sig}(\mathcal{O}) = \mathsf{sig}(\mathcal{O}^{-\mathsf{r}}) \cup \mathsf{r}$, Condition (1) of Definition 7 is satisfied. Let $\mathcal{I}$ be a model of $\mathcal{O}$. By definition:

$$\Delta^{\mathcal{I}} \subseteq {C_1}^{\mathcal{I}} \cup (\exists r.D_1)^{\mathcal{I}} \tag{1}$$

$$\Delta^{\mathcal{I}} \subseteq {V_1}^{\mathcal{I}} \cup (\forall r.W_1)^{\mathcal{I}} \tag{2}$$

$${D_1}^{\mathcal{I}} \cap {W_1}^{\mathcal{I}} \subseteq \emptyset \tag{3}$$

From these inclusions, we derive:

$$(\neg C_1)^{\mathcal{I}} \subseteq (\exists \mathsf{r}.V_1)^{\mathcal{I}} \text{ from (13)} \tag{4}$$

$$(\neg V_1)^{\mathcal{I}} \subseteq (\forall \mathsf{r}.W_1)^{\mathcal{I}} \text{ from (14)} \tag{5}$$

$$(\neg C_1)^{\mathcal{I}} \cap (\neg V_1)^{\mathcal{I}} \subseteq (\exists \mathsf{r}.D_1)^{\mathcal{I}} \cap (\forall \mathsf{r}.W_1)^{\mathcal{I}} \text{ from (15) (16)} \tag{6}$$

Proof by contradiction: assume $(\neg C_1)^{\mathcal{I}} \cap (\neg V_1)^{\mathcal{I}} \neq \emptyset$. Then there exists $x \in \Delta^{\mathcal{I}}$ where $x \in (\neg C_1)^{\mathcal{I}} \cap (\neg V_1)^{\mathcal{I}}$, implying $x \in (\exists \mathsf{r}.D_1)^{\mathcal{I}} \cap (\forall \mathsf{r}.W_1)^{\mathcal{I}}$. This necessitates the existence of $y \in \Delta^{\mathcal{I}}$ where $(x, y) \in \mathsf{r}^{\mathcal{I}}$, $y \in {D_1}^{\mathcal{I}}$, and $y \in {W_1}^{\mathcal{I}}$, contradicting (15). Therefore, $(\neg C_1)^{\mathcal{I}} \cap (\neg V_1)^{\mathcal{I}} = \emptyset$, establishing $\mathcal{I} \models C_1 \sqcup V_1$. For the converse, let $\mathcal{I}'$ be a model of $\mathcal{O}^{-\mathsf{r}}$. We construct $\mathcal{I}$ coinciding with $\mathcal{I}'$ on all symbols except r, D_1, and W_1, where:

$$(\exists \mathsf{r}.D_1)^{\mathcal{I}} = (\neg C_1)^{\mathcal{I}'} \tag{7}$$

$$(\forall \mathsf{r}.W_1)^{\mathcal{I}} = (\neg V_1)^{\mathcal{I}'} \tag{8}$$

Since $\mathcal{I}' \models \mathcal{O}'$, we have $(\neg C_1)^{\mathcal{I}'} \cap (\neg V_1)^{\mathcal{I}'} = \emptyset$, implying $(\exists \mathsf{r}.D_1)^{\mathcal{I}} \cap (\forall \mathsf{r}.W_1)^{\mathcal{I}} = \emptyset$. As r is absent from C_1, D_1, V_1, and W_1, their interpretations remain invariant from $\mathcal{I}'$ to $\mathcal{I}$: ${C_1}^{\mathcal{I}'} = {C_1}^{\mathcal{I}}$, ${D_1}^{\mathcal{I}'} = {D_1}^{\mathcal{I}}$, ${V_1}^{\mathcal{I}'} = {V_1}^{\mathcal{I}}$, and ${W_1}^{\mathcal{I}'} = {W_1}^{\mathcal{I}}$. Thus we have $(\neg C_1)^{\mathcal{I}} \subseteq (\exists \mathsf{r}.V_1)^{\mathcal{I}}$ and $\mathcal{I} \models C_1 \sqcup \exists \mathsf{r}.D_1$, and $(\neg V_1)^{\mathcal{I}} \subseteq (\forall \mathsf{r}.W_1)^{\mathcal{I}}$ and $\mathcal{I} \models V_1 \sqcup \forall \mathsf{r}.W_1$. Finally, if $\mathcal{I} \not\models D_1 \sqcap W_1 \sqsubseteq \bot$, we can construct an interpretation $\mathcal{I}$ with $\Delta^{\mathcal{I}} = \{a, b\}$, where ${C_1}^{\mathcal{I}} = {V_1}^{\mathcal{I}} = {D_1}^{\mathcal{I}} = {W_1}^{\mathcal{I}} = \{b\}$ and $\mathsf{r}^{\mathcal{I}} = \{(a, b)\}$. This yields $\mathcal{I} \models \exists \mathsf{r}.D_1 \sqcap \forall \mathsf{r}.W_1$, contradicting our earlier result that $(\exists \mathsf{r}.D_1)^{\mathcal{I}} \cap (\forall \mathsf{r}.W_1)^{\mathcal{I}} = \emptyset$.

Therefore, $\mathcal{O}$ is a conservative extension of $\mathcal{O}^{-\mathsf{r}}$. By the transitivity property of conservative extension and Proposition 1, $\mathcal{O}$ is a conservative extension of $\mathcal{M}$.

Theorem 1. *Given any $\mathcal{ALCI}$-ontology $\mathcal{O}$ and any forgetting signature $\mathcal{F} \subseteq \mathsf{sig}(\mathcal{O})$ 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 $\mathcal{O}$.*

Proof. Note that the normalization and inference rules do not introduce new cycles. For cases where cyclic behavior originally exhibits over the names in $\mathcal{F}$, the method terminates upon detecting a cycle. In acyclic cases, termination of the method follows from Lemma 3 and the termination of the forgetting calculi. The method's soundness is ensured by Lemmas 3, 4 and 5.