Efficient Extraction of $\mathcal{EL}$-Ontology Deductive Modules[*]

Hui Yang, Yue Ma, Nicole Bidoit

LISN, CNRS, Université Paris-Saclay
yang@lisn.fr, ma@lisn.fr, nicole.bidoit@lisn.fr

Abstract

Because widely used real-world ontologies are often complex and large, one important challenge has emerged: designing tools for users to focus on sub-ontologies corresponding to their specific interests. To this end, various modules have been introduced to provide concise ontology views. This work concentrates on extracting deductive modules that preserve logical entailments over a given vocabulary. Existing deductive module proposals are either inefficient from a computing point of view or unsatisfactory from a quality point of view because the modules extracted are not concise enough. For example, *minimal modules* guarantee the most concise results but their computation is highly time-consuming, while $\bot\top^*$*-modules* are easy to compute but usually they contain many redundant items. To overcome computation cost and lack of quality, we propose to compute two different kinds of deductive modules called *pseudo-minimal modules* and *complete modules* for $\mathcal{EL}$-ontology. Our deductive module definitions rely on associating a tree representation with an ontology, and their computation is based on SAT encoding. Our experiments on real-world ontologies show that our pseudo-minimal modules are indeed minimal modules in almost all cases (98.9%), and computing pseudo-minimal modules is more efficient (99.79 times faster on average) than the state-of-the-art method `Zoom` for computing minimal modules. Also, our complete modules are more compact than $\bot\top^*$-modules, but their computation time remains comparable. Finally, note that our proposal applies to $\mathcal{EL}$-ontologies while `Zoom` only works for $\mathcal{EL}$-terminologies.

Introduction

Description logic-based ontologies have been widely studied and used in many areas. However, real-world ontologies are often too big to be handled by humans. The most evident approach for overcoming this problem is to extract sub-ontologies related to user interests. For example, the well-known biomedical ontology *Snomed CT* contains more than 300,000 axioms. By extracting deductive modules, we could provide doctors with small sub-ontologies of *Snomed CT* based on symptoms to establish a diagnosis. Extracting deductive modules has been used for various different areas, like ontology debugging (Arif et al. 2016), re-use (Jiménez-Ruiz et al. 2008), and forgetting (Koopmann and Schmidt 2013).

We can distinguish two classes of deductive modules. The first class is syntactical locality-based modules such as $\bot\top^*$-modules (Sattler, Schneider, and Zakharyaschev 2009) and *AMEX-modules* (Gatens, Konev, and Wolter 2014). Those modules can be computed efficiently, but usually, they contain many unnecessary terms. The second class is subset-minimal modules such as *minimal modules* (Chen et al. 2017). They do not contain any redundant terms but suffer from high complexity and are time-consuming in practice.

In this work, we propose a new method for computing deductive modules for $\mathcal{EL}$ ontologies to balance the computation cost and the result quality. Our method is inspired by the SAT-based approach (Arif, Mencía, and Marques-Silva 2015; Manthey, Peñaloza, and Rudolph 2016) developed to compute *justifications*, which are minimal sub-ontologies that derive a given entailment. The main idea of these SAT-based methods is to encode the derivations of a given entailment as a set of Horn-clauses, then it enumerates all the justifications of this entailment by SAT tools or *resolution* (Kazakov and Skočovský 2018). However, the computation of deductive modules is much more complex: First, the input is a vocabulary instead of an entailment and it could be complicated to generate all the entailments over the given vocabulary; Second, there may exist (even infinitely) many entailments over a given vocabulary. Therefore, instead of using justifications of the entailments directly, one has to find other proper ways to tackle the computation of deductive modules.

Our contribution is twofold: **(i)** We associate a forest with each given ontology and vocabulary to efficiently capture the entailments over the vocabulary. The definition of *forest* is inspired by the *regular tree grammar* developed in (Nikitina and Rudolph 2014). We can regard the forest and the regular tree grammar as a set of *derivation trees* and *derivation rules* that generate entailments over a given vocabulary, respectively. Moreover, we are able to deal with the case of infinitely many entailments by considering a finite subset of trees from our forest. **(ii)** We introduce two novel notions of deductive modules called *pseudo-minimal modules* and *complete modules*, and we develop an efficient SAT-based algorithm to compute them based on the notion

[*]This work is funded by the BPI-France (PSPC AIDA: 2019-PSPC-09).

of the forest. Our pseudo-minimal modules are quite interesting approximations of minimal modules: (1) They are indeed minimal modules when there are finitely many entailments over a given vocabulary; (2) Moreover, our algorithm is 99.79 times faster on average than the state-of-the-art algorithm `Zoom` (Chen et al. 2017) which computes all the minimal modules but only for $\mathcal{EL}$-terminologies. Compared to pseudo-minimal modules, our complete modules are less concise but easier to compute. They are far more concise than the $\bot\top^*$-module, as demonstrated by our experiment, but their calculation time remains comparable.

This paper is organized as follows. First, we introduce our notion of the forest and define *pseudo-minimal modules* based on this notion, while *complete modules* are defined directly from the definition of deductive modules (Definition 2). Next, we compute pseudo-minimal modules and complete modules using Horn-clause encoding. Then, we validate our method by running experiments on real-world ontologies with the prototype `ForMod`. Finally, after the discussion of the related work, we conclude the paper with some future work directions.

Preliminaries

Given finite sets of *atomic concepts* $\mathsf{N}_C=\{A,B,\cdots\}$ and *atomic roles* $\mathsf{N}_R=\{r,s,\cdots\}$, *$\mathcal{EL}$-concepts* C and *$\mathcal{EL}$-axioms* α are built by the grammar rules (i) $C ::= \top \mid A \mid C\sqcap C \mid \exists r.C$ or (ii) $\alpha ::= C\sqsubseteq C$. An *$\mathcal{EL}$-ontology* $\mathcal{O}$ is a finite set of $\mathcal{EL}$-axioms. We denote by $sig(\mathcal{O})$ (resp. $sig(C)$) the set of the atomic concepts and roles that appear in $\mathcal{O}$ (resp. C). For example, $sig(B_1\sqcap\exists r.B_2)=\{r,B_1,B_2\}$.

An *interpretation* $\mathcal{I}=(\Delta^{\mathcal{I}},\cdot^{\mathcal{I}})$ of ontology $\mathcal{O}$ consists of a non-empty set $\Delta^{\mathcal{I}}$ and a mapping from each atomic concept $A\in\mathsf{N}_C$ to a subset $A^{\mathcal{I}}\subseteq\Delta^{\mathcal{I}}$, and from each role $r\in\mathsf{N}_R$ to a subset $r^{\mathcal{I}}\subseteq\Delta^{\mathcal{I}}\times\Delta^{\mathcal{I}}$. For a concept C, we define $C^{\mathcal{I}}$ inductively by: $(\top)^{\mathcal{I}}=\Delta^{\mathcal{I}}$, $(C\sqcap D)^{\mathcal{I}}=C^{\mathcal{I}}\cap D^{\mathcal{I}}$, $(\exists r.C)^{\mathcal{I}}=\{a\in\Delta^{\mathcal{I}} \mid \exists\, b\in C^{\mathcal{I}},(a,b)\in r^{\mathcal{I}}\}$. An interpretation is a *model* of $\mathcal{O}$ if it is compatible with all axioms in $\mathcal{O}$, i.e., for all $C\sqsubseteq D\in\mathcal{O}$, we have $C^{\mathcal{I}}\subseteq D^{\mathcal{I}}$. We say $\mathcal{O}\models\alpha$ where α is an axiom iff any model of $\mathcal{O}$ is compatible with α.

An $\mathcal{EL}$-ontology $\mathcal{O}$ is *normalized* if all its axioms are of the form: $A\sqsubseteq B_1\sqcap\cdots\sqcap B_n$, $B_1\sqcap\cdots\sqcap B_n\sqsubseteq A$, $\exists r.B\sqsubseteq A$, $A\sqsubseteq\exists r.B$, where $A,B,B_i\in\mathsf{N}_C, r\in\mathsf{N}_R, n\geq 1$. Every $\mathcal{EL}$-ontology can be normalized in polynomial time by introducing new atomic concepts.

Let $L_A=\{C \mid C\sqsubseteq A\in\mathcal{O}\}, R_A=\{D \mid A\sqsubseteq D\in\mathcal{O}\}$. We say an atomic concept A is *primitive* iff (i) $L_A=\emptyset$ **or** (ii) $L_A=R_A$ and $|L_A|=1$. $\mathcal{O}$ is a *terminology* iff all the atomic concepts in $\mathcal{O}$ are primitive. The notion terminology defined here equals to that introduced in (Konev, Walther, and Wolter 2009). We state the definition in a different way because we use a different form of normalized ontologies.

Example 1. *$\mathcal{O}$ defined below is a normalized ontology. $\mathcal{O}$ is a terminology because all its atomic concepts are primitive.*

$$\mathcal{O}=\{\alpha_1{:}A\sqsubseteq\exists r.B_1,\ \alpha_2{:}B_1\sqsubseteq A_1\sqcap A_2,\ \alpha_3{:}A\sqsubseteq\exists r.A_1 \\ \alpha_4{:}\exists r.B_2\sqsubseteq A_2,\ \alpha_5{:}A_3\sqcap A_4\sqsubseteq B_2\}.$$

Definition 1 (Justification)**.** *Given an ontology $\mathcal{O}$ such that $\mathcal{O}\models A\sqsubseteq B$. A justification of $A\sqsubseteq B$ is a minimal sub-ontology $J\subseteq\mathcal{O}$ such that $J\models A\sqsubseteq B$.*

Given two ontologies $\mathcal{O}_1,\mathcal{O}_2$ and a *signature* $\Sigma\subseteq\mathsf{N}_C\cup\mathsf{N}_R$ of atomic concepts and roles, the *logical difference* (Konev et al. 2012) between $\mathcal{O}_1$ and $\mathcal{O}_2$ w.r.t. Σ is defined as the set of axioms inferred by $\mathcal{O}_1$ but not inferred by $\mathcal{O}_2$:

$$\mathtt{cDiff}_\Sigma(\mathcal{O}_1,\mathcal{O}_2)=\{\alpha \mid sig(\alpha)\subseteq\Sigma,\mathcal{O}_1\models\alpha,\mathcal{O}_2\not\models\alpha\}.$$

Definition 2 (Deductive module)**.** *A* deductive module *for an ontology $\mathcal{O}$ and a signature Σ is a sub-ontology $\mathcal{M}\subseteq\mathcal{O}$ such that $\mathtt{cDiff}_\Sigma(\mathcal{O},\mathcal{M})=\emptyset$. Moreover:*

- *$\mathcal{M}$ is a* ***minimal module*** *for $\mathcal{O}$ and Σ if $\mathcal{M}$ is a minimal (under inclusion) deductive module for $\mathcal{O}$ and Σ.*
- *$\mathcal{M}$ is a* ***complete module*** *for $\mathcal{O}$ and Σ if $\mathcal{M}$ contains all minimal modules for $\mathcal{O}$ and Σ.*

It is clear that the union of all minimal modules is a complete module, and so is the ontology itself. There can be multiple deductive, minimal, and complete modules.

Example 2 (Example 1 cont'd)**.** *Assume that Σ is the signature given by $\{A,A_1,A_2,A_3,A_4,r\}$. Then $\mathcal{O}\setminus\{\alpha_3\}$ is a deductive module for $\mathcal{O}$ and Σ. $\mathcal{O}\setminus\{\alpha_3\}$ is also a minimal module because no proper subset of $\mathcal{O}\setminus\{\alpha_3\}$ is a deductive module. Moreover, $\mathcal{O}\setminus\{\alpha_3\}$ is also a complete module since it is the unique minimal module for $\mathcal{O}$ and Σ.*

We now introduce the notion of *uniform interpolation*, which provides a way to capture entailments over a given signature w.r.t. an ontology $\mathcal{O}$ as defined below:

Definition 3 (Uniform interpolation)**.** *Given an ontology $\mathcal{O}$ and a signature Σ, an ontology $\mathcal{U}^{\mathcal{O}}_{\Sigma}$ is a uniform interpolation* ***(hereafter UI)*** *for $\mathcal{O}$ and Σ iff (i) $sig(\mathcal{U}^{\mathcal{O}}_{\Sigma})\subseteq\Sigma$; (ii) $\mathcal{O}\models\alpha\Leftrightarrow\mathcal{U}^{\mathcal{O}}_{\Sigma}\models\alpha$ for any axiom α with $sig(\alpha)\subseteq\Sigma$.*

Example 3 (Example 2 cont'd)**.** *The ontology $\mathcal{U}^{\mathcal{O}}_{\Sigma}=\{\beta_1 : A\sqsubseteq\exists r.(A_1\sqcap A_2),\ \beta_2 : A\sqsubseteq\exists r.A_1,\ \beta_3 : \exists r.(A_3\sqcap A_4)\sqsubseteq A_2\}$ is a uniform interpolation for $\mathcal{O}$ and Σ.*

Given an ontology and a signature, a UI does not always exist. To overcome this problem, several approaches (Lutz, Piro, and Wolter 2010; Calvanese, Giacomo, and Lenzerini 1999) have been proposed to provide approximations of UI.

In the following, our definitions and theorems always assume that $\mathcal{O}$ is a normalized $\mathcal{EL}$-ontology and Σ a signature unless otherwise stated.

Introducing Forest $F^{\mathcal{O}}_{\Sigma}$

Now, we analyze possible ways to compute deductive models and then introduce our notion of forest $F^{\mathcal{O}}_{\Sigma}$.

Motivation

Assuming $\mathcal{U}^{\mathcal{O}}_{\Sigma}$ is a UI for an ontology $\mathcal{O}$ and a signature Σ, we can compute deductive modules for $\mathcal{O}$ and Σ using the justifications of each axiom $\beta\in\mathcal{U}^{\mathcal{O}}_{\Sigma}$ (see Figure 1). More precisely, consider the following collection of sub-ontologies defined as the union of justifications:

$$\mathcal{S}=\{\bigcup_{\alpha\in\mathcal{U}^{\mathcal{O}}_{\Sigma}} J_\alpha \mid J_\alpha \text{ is a justification of } \alpha\}.$$

Each element $\mathcal{M}$ in $\mathcal{S}$ is a sub-ontology of $\mathcal{O}$ and contains exactly one justification for each element in $\mathcal{U}^{\mathcal{O}}_{\Sigma}$. Thus it is a

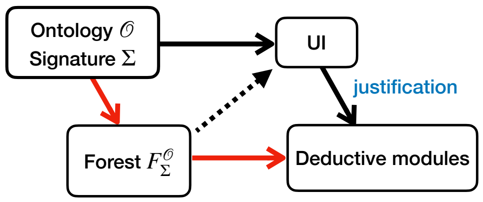
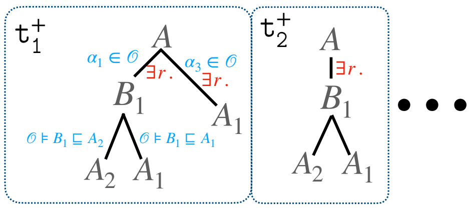


Figure 1: The main schema


t^+_1
A
$\alpha_1 \in \mathcal{O}$
$\exists r.$
$\alpha_3 \in \mathcal{O}$
$\exists r.$
B_1
A_1
$\mathcal{O} \models B_1 \sqsubseteq A_2$
$\mathcal{O} \models B_1 \sqsubseteq A_1$
A_2
A_1
t^+_2
A
$\exists r.$
B_1
A_2
A_1
Figure 2: f-trees (blue words are the label of edges).

deductive module for $\mathcal{O}$ and Σ. Moreover, we can conclude that the minimal modules for $\mathcal{O}$ and Σ are identical to the subset-minimal sub-ontologies in the collection $\mathcal{S}$.

Example 4 (Example 3 cont'd)**.** *Recall the ontology $\mathcal{O}$, the signature Σ and the UI $\mathcal{U}^{\mathcal{O}}_{\Sigma}$ of our running example. We know that there are one justification for β_1: $J_{\beta_1} = \{\alpha_1, \alpha_2\}$, two justifications for β_2: $J^1_{\beta_2} = \{\alpha_1, \alpha_2\}, J^2_{\beta_2} = \{\alpha_3\}$ and one justification for β_3: $J_{\beta_3} = \{\alpha_4, \alpha_5\}$. Therefore, the collection $\mathcal{S} = \{\mathcal{O}, \mathcal{O}\backslash\{\alpha_3\}\}$ contains two different deductive modules for $\mathcal{O}$ and Σ. Moreover, $\mathcal{O}\backslash\{\alpha_3\}$ is the unique minimal module for $\mathcal{O}$ and Σ.*

In Figure 1, computing deductive modules through UI and justifications is shown by the black bold arrows. There are two main difficulties in implementing this simple idea: **(i)** computing UIs is hard, and **(ii)** a UI does not always exist. Our contribution to overcome these difficulties is to proceed to the computation of deductive modules through a new object $F^{\mathcal{O}}_{\Sigma}$, called a *forest*, associated with the ontology $\mathcal{O}$ and the signature Σ. This alternative computation process is depicted by the red arrows in Figure 1.

The notion of forest $F^{\mathcal{O}}_{\Sigma}$, formally introduced hereafter, is inspired by the *regular tree grammar* developed in (Nikitina and Rudolph 2014). The forest $F^{\mathcal{O}}_{\Sigma}$ and the regular tree grammar can be regarded respectively as a set of *derivation trees* and *derivation rules* that generate candidate axioms of a UI. Our method for computing deductive modules relies on the structure of the trees in $F^{\mathcal{O}}_{\Sigma}$ instead of using justifications, and has mainly three benefits: **(i)** the computation of $F^{\mathcal{O}}_{\Sigma}$ is efficient; **(ii)** our method works no matter a UI exists or not; **(iii)** our method can be easily encoded as a SAT-problem and thus solved by efficient SAT-tools.

Next, we develop the presentation of forests $F^{\mathcal{O}}_{\Sigma}$ and show the relation between $F^{\mathcal{O}}_{\Sigma}$ and UI (i.e., the dotted arrow in Figure 1). In the following sections, we introduce *pseudo-minimal modules* using $F^{\mathcal{O}}_{\Sigma}$ and describe `ForMod`, our SAT-based algorithm, to compute them.

Definition of Forest $F^{\mathcal{O}}_{\Sigma}$

Next, we consider *labeled trees*. Each labeled tree $\mathtt{t}$ is associated with a *label map* "$\mathtt{lab}$" that maps **(i)** node $\mathtt{n}$ in $\mathtt{t}$ to an atomic concept; **(ii)** edge e in $\mathtt{t}$ to a condition of the form $\alpha{\in}\mathcal{O}$ or $\mathcal{O}{\models}A{\sqsubseteq}B$. For simplicity, we use $\mathtt{r}(\mathtt{t})$ to denote the root of $\mathtt{t}$, and $\mathtt{A_t}$ the label of the root (i.e., $\mathtt{A_t} = \mathtt{lab}(\mathtt{r}(\mathtt{t}))$).

The forest $F^{\mathcal{O}}_{\Sigma}$ consists of two kinds of trees, *forward trees* and *backward trees*, defined as follows:

Definition 4. *A labeled tree $\mathtt{t}^+$ is a* ***forward tree*** *(hereafter f-tree) from A to Σ over $\mathcal{O}$ iff:*

1. *the label of the root of $\mathtt{t}^+$ is A (i.e., $\mathtt{A}_{\mathtt{t}^+} = A$);*
2. *for node $\mathtt{n} \in \mathtt{t}^+$ distinct from the root, $\mathtt{lab}(\mathtt{n}) \in \Sigma$ iff $\mathtt{n}$ is a leaf of $\mathtt{t}^+$,*
3. *if the child set of a node $\mathtt{n}_0 \in \mathtt{t}^+$ is $\{\mathtt{n}_1, \cdots, \mathtt{n}_m\}$ and $B_i = \mathtt{lab}(\mathtt{n}_i), 0 \leq i \leq m$, then for each $0 \leq i \leq m$, one of the following conditions holds:*
   - *(a) $B_0 \sqsubseteq \exists r.B_i \in \mathcal{O}$ and $r \in \Sigma$,*
   - *(b) $\mathcal{O} \models B_0 \sqsubseteq B_i$, where B_i is not primitive or $B_i \in \Sigma$.*

   The edge e from $\mathtt{n}_0$ to $\mathtt{n}_i$ is labeled by the condition[1] that generates e.

For the sake of the presentation, edges labeled by $B_0{\sqsubseteq}\exists r.B_i{\in}\mathcal{O}$ (i.e., case 3(a)) are called $\exists r$-edges.

Example 5 (Example 4 cont'd)**.** *For simplicity and without ambiguity, in the figure, we represent a node $\mathtt{n}$ by its label $\mathtt{lab}(\mathtt{n})$ and mark $\exists r$-edges by "$\exists r$". In Figure 2, $\mathtt{t}^+_1$, $\mathtt{t}^+_2$ are two f-trees from A to Σ over $\mathcal{O}$, and $\mathtt{t}^+_2$ is a sub-tree of $\mathtt{t}^+_1$. There are still 5 other f-trees from A to Σ over $\mathcal{O}$ that are proper sub-trees of $\mathtt{t}^+_1$.*

For $\Sigma'{=}\Sigma \cup \{B_1\}$, the trees $\mathtt{t}^+_1, \mathtt{t}^+_2$ are no longer f-trees from A to Σ' over $\mathcal{O}$, because $\mathtt{t}^+_1, \mathtt{t}^+_2$ both contain an internal node labeled by $B_1 \in \Sigma'$, which violates the requirement 2 in Definition 4.

Definition 5. *A labeled tree $\mathtt{t}^-$ is a* ***backward tree*** *(hereafter b-tree) from A to Σ over $\mathcal{O}$ iff:*

1. *the label of the root of $\mathtt{t}^-$ is A (i.e., $\mathtt{A}_{\mathtt{t}^-} = A$);*
2. *for node $\mathtt{n} \in \mathtt{t}^-$ distinct from the root, $\mathtt{lab}(\mathtt{n}) \in \Sigma$ iff $\mathtt{n}$ is a leaf of $\mathtt{t}^-$;*
3. *if the child set of a node $\mathtt{n}_0 \in \mathtt{t}^-$ is $\{\mathtt{n}_1, \cdots, \mathtt{n}_m\}$ and $B_i = \mathtt{lab}(\mathtt{n}_i), 0 \leq i \leq m$, then one of the following conditions holds:*
   - *(a) $m = 1$, $\exists r.B_1 \sqsubseteq B_0 \in \mathcal{O}, r \in \Sigma$;*
   - *(b) $m = 1$, $\mathcal{O} \models B_1 \sqsubseteq B_0$, B_1 is not primitive or $B_1{\in}\Sigma$;*
   - *(c) $m > 1$, $B_1 \sqcap \cdots \sqcap B_m \sqsubseteq B_0 \in \mathcal{O}$.*

   Again, here, the edge e from $\mathtt{n}_0$ to $\mathtt{n}_i$ is labeled by the condition generating e.

Similarly, edges labeled with $\exists r.B_1{\sqsubseteq}B_0{\in}\mathcal{O}$ are called $\exists r$-edges. Note that for b-trees, all edges from a node to its children are generated by the same condition.

Example 6 (Example 4 cont'd)**.** *In Figure 3, $\mathtt{t}^-_1$ is the only b-tree from A_2 to Σ over $\mathcal{O}$. Note that the tree on the right of Figure 3 is not a b-tree from A_2 to Σ over $\mathcal{O}$ because we do not have $A_3 \sqsubseteq B_2 \in \mathcal{O}$ nor $\mathcal{O} \models A_3 \sqsubseteq B_2$, which violates the requirement 3 of Definition 5.*

[1]The label $\mathtt{lab}(e)$ is $B{\sqsubseteq}\exists r.B_i \in \mathcal{O}$ for the case 3(a); $\mathtt{lab}(e)$ is $\mathcal{O}{\models}B{\sqsubseteq}B_i$ for the case 3(b).

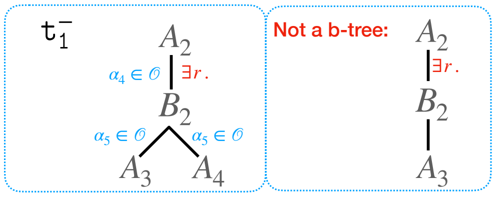


Figure 3: b-tree (an example and a counter-example)

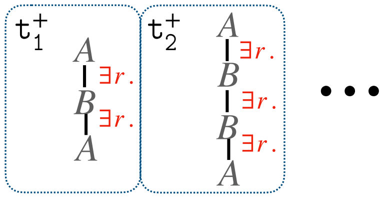


Figure 4: Some trees of $F_{\Sigma_1}^{\mathcal{O}_1}$

The forest $F_\Sigma^\mathcal{O}$ associated with an ontology $\mathcal{O}$ and a signature Σ is defined as follows:

Definition 6. *The forest $F_\Sigma^\mathcal{O}$ is the collection of all f-trees and b-trees from A to Σ over $\mathcal{O}$, where $A \in sig(\mathcal{O})$.*

Notice that $F_\Sigma^\mathcal{O}$ can be an infinite set. For example:

Example 7. *Let $\mathcal{O}_1{=}\{A{\sqsubseteq}\exists r.B,\ B{\sqsubseteq}\exists r.B,\ B{\sqsubseteq}\exists r.A\}$ and $\Sigma_1{=}\{A, r\}$. Then, $F_{\Sigma_1}^{\mathcal{O}_1}$ contains infinitely many f-trees $\mathtt{t}_1^+, \mathtt{t}_2^+, \cdots$ from A to Σ_1 over $\mathcal{O}_1$ as shown in Figure 4.*

The f-trees in Figure 4 are generated by the loop starting at the node labeled by B. More precisely, we say that a tree $\mathtt{t} \in F_\Sigma^\mathcal{O}$ contains a **loop** if there is a path p from root $\mathtt{r}(\mathtt{t})$ to a leaf of $\mathtt{t}$ with two different nodes having the same label. For the same reason, there could be infinitely many b-trees in a forest $F_\Sigma^\mathcal{O}$. We have that $F_\Sigma^\mathcal{O}$ is infinite iff some $\mathtt{t} \in F_\Sigma^\mathcal{O}$ contains a loop.

Generating UI from Forest $F_\Sigma^\mathcal{O}$

Although our computation of deductive modules does not require computing UI, we investigate here how to generate a UI from a forest. These discussions (definitions and results) are necessary for presenting our method.

First, we associate each f-tree or b-tree $\mathtt{t}$ with an $\mathcal{EL}$-concept $\mathtt{C_t}$ defined as follows:

Definition 7. *For a f-tree or b-tree $\mathtt{t}$, the corresponding $\mathcal{EL}$-concept $\mathtt{C_t}$ is defined inductively by **(i)** if the child set of the root $\mathtt{r}(\mathtt{t})$ is empty, then $\mathtt{C_t} = \mathtt{A_t}$; **(ii)** if the child set of the root $\mathtt{r}(\mathtt{t})$ is $\{\mathtt{n}_1, \cdots, \mathtt{n}_m\}$, and there exists $0 \le k \le m$ such that the edge from $\mathtt{r}(\mathtt{t})$ to $\mathtt{n}_i$ is a $\exists r_i$-edge for $i \le k$. Then*

$$\mathtt{C_t} = \exists r_1 \mathtt{C}_{\mathtt{t}_1} \sqcap \cdots \sqcap \exists r_k \mathtt{C}_{\mathtt{t}_k} \sqcap \mathtt{C}_{\mathtt{t}_{k+1}} \sqcap \cdots \sqcap \mathtt{C}_{\mathtt{t}_m},$$

where $\mathtt{t}_i$ is the maximal sub-tree of $\mathtt{t}$ rooted at $\mathtt{n}_i$.

For instance, we have $\mathtt{C}_{\mathtt{t}_1^+}{=}\exists r.(A_2 \sqcap A_1) \sqcap \exists r.A_1$ for $\mathtt{t}_1^+$ in Example 5 and we have $\mathtt{C}_{\mathtt{t}_1^-}{=}\exists r.(A_3 \sqcap A_4)$ for $\mathtt{t}_1^-$ in Example 6. Now, the UI candidate generated from $F_\Sigma^\mathcal{O}$ is defined as follows.

Definition 8. *Given a forest F of b-trees and f-trees, the set of axioms $\mathcal{U}_\Sigma(F)$ associated with F is the union of the three axiom sets below:*

$$\mathcal{U}_\Sigma^1(F){=}\{\mathtt{C}_{\mathtt{t}^-} \sqsubseteq \mathtt{A}_{\mathtt{t}^-} \mid \mathtt{t}^- \in F,\ \ \mathtt{A}_{\mathtt{t}^-} \in \Sigma\}$$
$$\mathcal{U}_\Sigma^2(F){=}\{\mathtt{A}_{\mathtt{t}^+} \sqsubseteq \mathtt{C}_{\mathtt{t}^+} \mid \mathtt{t}^+ \in F,\ \ \mathtt{A}_{\mathtt{t}^+} \in \Sigma\}$$
$$\mathcal{U}_\Sigma^3(F){=}\{\mathtt{C}_{\mathtt{t}^-} \sqsubseteq \mathtt{C}_{\mathtt{t}^+} \mid \mathtt{t}^-, \mathtt{t}^+ \in F,\ \ \mathtt{A}_{\mathtt{t}^-}{=}\mathtt{A}_{\mathtt{t}^+} \notin \Sigma\}.$$

$\mathcal{U}_\Sigma(F)$ is a UI candidate because of the following result:

Theorem 1. *For any axiom α with $sig(\alpha) \subseteq \Sigma$, we have $\mathcal{O} \models \alpha$ iff $\mathcal{U}_\Sigma(F_\Sigma^\mathcal{O}) \models \alpha$. Therefore, $\mathcal{U}_\Sigma(F_\Sigma^\mathcal{O})$ is a UI for $\mathcal{O}$ and Σ if $F_\Sigma^\mathcal{O}$ is finite.*

Next, we omit Σ in $\mathcal{U}_\Sigma(F)$ and $\mathcal{U}_\Sigma^i(F), i \in \{1,2,3\}$ if there is no ambiguity.

Example 8 (Example 5 and 6 cont'd)**.** *Note that $F_\Sigma^\mathcal{O}$ is finite because no b-tree or f-tree over the ontology $\mathcal{O}$ contains a loop. We have $\mathcal{U}^1(F_\Sigma^\mathcal{O}) = \{\beta_1' : \exists r.(A_3{\sqcap}A_4) \sqsubseteq A_2\}$,*

$$\mathcal{U}^2(F_\Sigma^\mathcal{O}) = \{\beta_2' : A \sqsubseteq \exists r.(A_2{\sqcap}A_1) \sqcap \exists r.A_1,$$
$$\beta_3' : A \sqsubseteq \exists r.(A_2{\sqcap}A_1),\ \beta_4' : A \sqsubseteq \exists r.A_2 \sqcap \exists r.A_1,$$
$$\beta_5' : A \sqsubseteq \exists r.A_1,\ \beta_6' : A \sqsubseteq \exists r.A_2\},$$

and $\mathcal{U}^3(F_\Sigma^\mathcal{O}) = \emptyset$. Here, β_1' is generated from $\mathtt{t}_1^- \in F_\Sigma^\mathcal{O}$ and β_2', β_3' are generated from $\mathtt{t}_1^+$, $\mathtt{t}_2^+ \in F_\Sigma^\mathcal{O}$ respectively. β_4', β_5', β_6' are generated from sub-trees of $\mathtt{t}_1^+$ other than $\mathtt{t}_2^+$. Then, $\mathcal{U}(F_\Sigma^\mathcal{O}) = \mathcal{U}^1(F_\Sigma^\mathcal{O}) \cup \mathcal{U}^2(F_\Sigma^\mathcal{O}) \cup \mathcal{U}^3(F_\Sigma^\mathcal{O})$ is a UI for $\mathcal{O}$ and Σ since $F_\Sigma^\mathcal{O}$ is finite.

It may happen that $F_\Sigma^\mathcal{O}$ is infinite and nevertheless, a UI for $\mathcal{O}$ and Σ exists. In that case, we can still obtain a UI from $F_\Sigma^\mathcal{O}$ by extracting a finite subset of $\mathcal{U}(F_\Sigma^\mathcal{O})$ as in (Nikitina and Rudolph 2014). We do not provide further details here as computing UI is not our main goal.

Pseudo-Minimal Modules

This section introduces a novel notion of deductive modules, called pseudo-minimal modules, based on the forest $F_\Sigma^\mathcal{O}$. The definition of pseudo-minimal modules relies on *tree-support* and a *finite representative subset of $F_\Sigma^\mathcal{O}$* defined below.

Tree-Support

Intuitively, tree-supports can be seen as analogs to justifications, although tree-supports are related to tree derivation, whereas justifications are related to axiom inference.

Definition 9. *A* tree-support *of a tree $\mathtt{t}{\in}F_\Sigma^\mathcal{O}$ is a sub-ontology of $\mathcal{O}$ defined as the union of the following axiom sets: **(i)** $\{\alpha\}$, for each edge $e \in \mathtt{t}$ labeled by $\alpha{\in}\mathcal{O}$; **(ii)** a justification $J_{A \sqsubseteq B} \subseteq \mathcal{O}$ of $A \sqsubseteq B$, for each edge $e \in \mathtt{t}$ labeled by $\mathcal{O} \models A \sqsubseteq B$. Let $\mathtt{Supp}(\mathtt{t})$ be the collection of all tree-supports of $\mathtt{t}$.*

For example, in Example 6, the only tree-support of $\mathtt{t}_1^-$ is $\{\alpha_4, \alpha_5\}$ since all edges in $\mathtt{t}_1^-$ are labeled by $\alpha_4{\in}\mathcal{O}$ or $\alpha_5{\in}\mathcal{O}$. Different tree-supports can be obtained depending on the different choices of justification $J_{A \sqsubseteq B}$.

Compared to justifications of $\alpha{\in}\mathcal{U}(F_\Sigma^\mathcal{O})$, tree-supports are easier to compute since **(i)** the first component $\{\alpha\}$ of tree-supports is obtained directly; **(ii)** computing justifications of $A \sqsubseteq B$, where $A, B \in \mathsf{N}_C$, is easier than computing the justifications of an arbitrary axiom $\alpha \in \mathcal{U}(F_\Sigma^\mathcal{O})$; **(iii)** the computation of tree-supports can be encoded as *Horn-clauses* and solved by efficient SAT tools as shown in the next section.

Finding a Finite Representative Subset of $F_\Sigma^\mathcal{O}$

Now, we extract a finite subset of $F_\Sigma^\mathcal{O}$. First, we partition $F_\Sigma^\mathcal{O}$ into three disjoint sets F_1, F_2 and F_3 as follows:

$$\begin{aligned} F_1 &= \{\mathtt{t} \in F_\Sigma^\mathcal{O} \mid \mathtt{t} \text{ contains a loop}\} \\ F_2 &= \{\mathtt{t}^+ \in F_\Sigma^\mathcal{O} \setminus F_1 \mid \exists\, \mathtt{t}^- \in F_1 \text{ such that } \mathtt{A}_{\mathtt{t}^+} = \mathtt{A}_{\mathtt{t}^-} \notin \Sigma\} \\ &\quad \cup \{\mathtt{t}^- \in F_\Sigma^\mathcal{O} \setminus F_1 \mid \exists\, \mathtt{t}^+ \in F_1 \text{ such that } \mathtt{A}_{\mathtt{t}^-} = \mathtt{A}_{\mathtt{t}^+} \notin \Sigma\} \\ F_3 &= F_\Sigma^\mathcal{O} \setminus (F_1 \cup F_2). \end{aligned}$$

Then F_2, F_3 are finite sets since they do not contain trees with loop, F_1 is infinite iff $F_\Sigma^\mathcal{O}$ is infinite.

Second, we select a finite subset F_1^* of F_1 as follows. Let us say that two f-trees (or b-trees) are *equivalent* iff they share the same set of edge labels. Then $F_1^* \subseteq F_1$ is obtained by selecting one representative tree for each equivalent class in F_1. Then, because the number of labels of the form $\alpha \in \mathcal{O}$ or $\mathcal{O} \models A \sqsubseteq B$ is finite given an ontology $\mathcal{O}$, the number of equivalent classes is finite and thus F_1^* is finite.

Finally, we obtain a finite representative subset $F_1^* \cup F_2 \cup F_3 \subseteq F_\Sigma^\mathcal{O}$. Note that if $F_\Sigma^\mathcal{O}$ is finite, then $F_1 = F_2 = F_1^* = \emptyset$, $F_3 = F_\Sigma^\mathcal{O}$ and thus $F_1^* \cup F_2 \cup F_3$ is $F_\Sigma^\mathcal{O}$ itself.

Pseudo-Minimal Modules

Now, we formally introduce pseudo-minimal modules as follows:

Definition 10. *A pseudo-minimal module for $\mathcal{O}$ and Σ is a minimal element in the collection* $\mathcal{S}_\Sigma^\mathcal{O} := \{\bigcup_{\mathtt{t} \in F_1^* \cup F_2 \cup F'} S_\mathtt{t} \mid S_\mathtt{t} \in \mathtt{Supp}(\mathtt{t}), F' \subseteq F_3, \mathcal{U}(F_2 \cup F') \models \mathcal{U}(F_2 \cup F_3)\}$.

We can regard pseudo-minimal modules as an approximation of minimal modules because of the following result:

Theorem 2. *A pseudo-minimal module for $\mathcal{O}$ and Σ is a deductive module for $\mathcal{O}$ and Σ. Moreover, if $F_\Sigma^\mathcal{O}$ is finite, then $\mathcal{M}$ is a pseudo-minimal module for $\mathcal{O}$ and Σ iff $\mathcal{M}$ is a minimal module for $\mathcal{O}$ and Σ.*

Notice that, in Definition 10 of $\mathcal{S}_\Sigma^\mathcal{O}$, if we do not consider $F' \subseteq F_3$ satisfying $\mathcal{U}(F' \cup F_2) \models \mathcal{U}(F_2 \cup F_3)$ (i.e., let $F' = F_3$), all the elements in $\mathcal{S}_\Sigma^\mathcal{O}$ are still deductive modules for $\mathcal{O}$ and Σ. However, in this case, we can not guarantee that pseudo-minimal modules are minimal modules when $F_\Sigma^\mathcal{O}$ is finite. For example:

Example 9 (Example 8 cont'd)**.** *Recall that $F_\Sigma^\mathcal{O}$ is finite, then $F_\Sigma^\mathcal{O} = F_3$. If we require $F' = F_3$, then $\mathcal{S}_\Sigma^\mathcal{O} = \left\{\bigcup_{\mathtt{t} \in F_\Sigma^\mathcal{O}} S_\mathtt{t} \;\middle|\; S_\mathtt{t} \in \mathtt{Supp}(\mathtt{t})\right\}$. Since $\mathtt{t}_1^+ \in F_\Sigma^\mathcal{O}$ has a unique tree-support $S_{\mathtt{t}_1^+} = \{\{\alpha_1, \alpha_2, \alpha_3\}\}$, all the sets in $\mathcal{S}_\Sigma^\mathcal{O}$ are super sets of $S_{\mathtt{t}_1^+}$ and thus contain α_3. However, as shown in Example 4, α_3 does not belong to any minimal module for $\mathcal{O}$ and Σ. On the other hand, if we allow $F' \subseteq F_3$, we can get rid of $\mathtt{t}_1^+$ because $\mathcal{U}(F') \models \mathcal{U}(F_\Sigma^\mathcal{O})$ for $F' = F_\Sigma^\mathcal{O} \setminus \{\mathtt{t}_1^+\}$. Therefore, we can get rid of α_3.*

When $F_\Sigma^\mathcal{O}$ is infinite, we show by our evaluation (see Table 4) that pseudo-minimal modules are still very concise. This provides an experimental validation of the minimal module approximation defined by pseudo-minimal modules.

`ForMod`: A SAT-Based Algorithm

Now, we present our algorithm `ForMod` that constructs a set of Horn-clauses $\mathcal{C}_\Sigma$ and computes pseudo-minimal modules and complete modules using $\mathcal{C}_\Sigma$. Assume that we have computed the finite representative subset $F_1^* \cup F_2 \cup F_3$ of $F_\Sigma^\mathcal{O}$ by enumeration of b-trees and f-trees.

Encoding by Horn-Clauses

Recall that all pseudo-minimal modules are minimal elements in the collection $\mathcal{S}_\Sigma^\mathcal{O}$. In the following, we encode the extraction of these minimal elements by a set of Horn-clauses: $\mathcal{C}_\Sigma$.

We associate to each $\alpha \in \mathcal{O}$, $\beta \in \mathcal{U}(F_1^* \cup F_2 \cup F_3)$, $\mathtt{t} \in F_\Sigma^\mathcal{O}$ and each edge $e \in \mathtt{t}$ a literal $l_\alpha, l_\beta, l_\mathtt{t}, l_e$, respectively. Also, we introduce a new literal l_Σ to capture pseudo-minimal modules for $\mathcal{O}$ and Σ. Now, the encoding is decomposed into two parts: **(i)** The first part (items 1, 2 and 3 below) encodes the computation of the subsets $F' \subseteq F_3$ such that $\mathcal{U}(F_2 \cup F') \models \mathcal{U}(F_2 \cup F_3)$; **(ii)** The second part (items 4 and 5 below) encodes the computation of the tree-supports $S_\mathtt{t} \in \mathtt{Supp}(\mathtt{t})$. In details, $\mathcal{C}_\Sigma$ consists of:

1. $(\wedge_{\beta \in \mathcal{U}(F_2 \cup F_3)} \boldsymbol{l_\beta}) \wedge (\wedge_{\mathtt{t} \in F_1^* \cup F_2} \boldsymbol{l_\mathtt{t}}) \rightarrow \boldsymbol{l_\Sigma}$;
2. for each axiom $\beta \in \mathcal{U}(F_2 \cup F_3)$, a Horn-clause is built depending on the provenance of β in $\mathcal{U}(F_2 \cup F_3)$ (recall Definition 8):
   (a) $\boldsymbol{l_{\mathtt{t}^-}} \rightarrow \boldsymbol{l_\beta}$ if $\beta = \mathtt{C}_{\mathtt{t}^-} \sqsubseteq \mathtt{A}_{\mathtt{t}^-} \in \mathcal{U}^1(F_2 \cup F_3)$,
   (b) $\boldsymbol{l_{\mathtt{t}^+}} \rightarrow \boldsymbol{l_\beta}$ if $\beta = \mathtt{A}_{\mathtt{t}^+} \sqsubseteq \mathtt{C}_{\mathtt{t}^+} \in \mathcal{U}^2(F_2 \cup F_3)$,
   (c) $\boldsymbol{l_{\mathtt{t}^+}} \wedge \boldsymbol{l_{\mathtt{t}^-}} \rightarrow \boldsymbol{l_\beta}$ if $\beta = \mathtt{C}_{\mathtt{t}^-} \sqsubseteq \mathtt{C}_{\mathtt{t}^+} \in \mathcal{U}^3(F_2 \cup F_3)$,

   for some $\mathtt{t}^-, \mathtt{t}^+ \in F_2 \cup F_3$.
3. $\boldsymbol{l_{\beta_1}} \wedge \cdots \wedge \boldsymbol{l_{\beta_n}} \rightarrow \boldsymbol{l_\beta}$, for each $\beta \in \mathcal{U}(F_2 \cup F_3)$ and each justification $\{\beta_1, \cdots, \beta_n\}$ of β w.r.t. $\mathcal{U}(F_2 \cup F_3)$;
4. $(\wedge_{e \in \mathtt{t}} \boldsymbol{l_e}) \rightarrow \boldsymbol{l_\mathtt{t}}$, for each $\mathtt{t} \in F_1^* \cup F_2 \cup F_3$;
5. For each edge $e \in \mathtt{t}$, where $\mathtt{t} \in F_1^* \cup F_2 \cup F_3$:
   (a) $(\wedge_{\alpha \in J_{A \sqsubseteq B}} \boldsymbol{l_\alpha}) \rightarrow \boldsymbol{l_e}$, if $\mathtt{lab}(e)$ is $\mathcal{O} \models A \sqsubseteq B$, where $J_{A \sqsubseteq B} \subseteq \mathcal{O}$ is a justification of $A \sqsubseteq B$;
   (b) $\boldsymbol{l_\alpha} \rightarrow \boldsymbol{l_e}$, if $\mathtt{lab}(e)$ is $\alpha \in \mathcal{O}$.

Above, for clarity, building the clauses of item 5(a) is presented using justifications. In the implementation, the computation of these justifications is itself encoded by a set of Horn-clauses as in (Yang, Ma, and Bidoit 2022).

Example 10 (Example 6 and 8 cont'd)**.** *For simplicity, we assume $F_\Sigma^\mathcal{O} = \{\mathtt{t}_1^+, \mathtt{t}_2^+, \mathtt{t}_1^-\}$ and $\mathcal{U}(F_\Sigma^\mathcal{O}) = \{\beta_1', \beta_2', \beta_3'\}$. Indeed, this simplification is based on the optimization steps that are explained later. Let $e_1, \cdots, e_7$ be the edges of the trees in $F_\Sigma^\mathcal{O}$ as illustrated in Figure 5. Then the set of Horn-clauses $\mathcal{C}_\Sigma$ is shown in Table 1.*

1:	$l_{\beta'_1} \wedge l_{\beta'_2} \wedge l_{\beta'_3} \rightarrow l_\Sigma$
2(a):	$l_{\mathtt{t}_1^-} \rightarrow l_{\beta'_1}$,
(b):	$l_{\mathtt{t}_1^+} \rightarrow l_{\beta'_2},\ l_{\mathtt{t}_2^+} \rightarrow l_{\beta'_3}$
3:	$l_{\beta'_2} \rightarrow l_{\beta'_3},\ l_{\beta'_3} \rightarrow l_{\beta'_2};$
4:	$l_{e_1} \wedge l_{e_2} \wedge l_{e_3} \wedge l_{e_4} \rightarrow l_{\mathtt{t}_1^+}$,
	$l_{e_1} \wedge l_{e_2} \wedge l_{e_3} \rightarrow l_{\mathtt{t}_2^+},\ \ l_{e_5} \wedge l_{e_6} \wedge l_{e_7} \rightarrow l_{\mathtt{t}_1^-}$
5(a):	$l_{\alpha_2} \rightarrow l_{e_2},\ l_{\alpha_2} \rightarrow l_{e_3}$,
(b):	$l_{\alpha_1} \rightarrow l_{e_1},\ l_{\alpha_3} \rightarrow l_{e_4}$,
	$l_{\alpha_4} \rightarrow l_{e_5},\ l_{\alpha_5} \rightarrow l_{e_6},\ \ l_{\alpha_5} \rightarrow l_{e_7}$

Table 1: Clause set $\mathcal{C}_\Sigma$

Complete and Pseudo-Minimal Modules

Let the **answer literals** be the literals l_α associated with axioms $\alpha \in \mathcal{O}$. We can extract a complete module directly from $\mathcal{C}_\Sigma$ as follows.

Theorem 3. *$\mathcal{M}^c = \{\alpha \in \mathcal{O} \mid l_\alpha$ is an answer literal in $\mathcal{C}_\Sigma\}$ is a complete module for $\mathcal{O}$ and Σ.*

For instance, in Example 10, the complete module $\mathcal{M}^c$ for $\mathcal{O}$ and Σ is $\mathcal{O}$ itself.

Pseudo-minimal modules are computed from $\mathcal{C}_\Sigma$ as in (Kazakov and Skočovskỳ 2018). We apply *resolution* over $\mathcal{C}_\Sigma \cup \{\neg l_\Sigma\}$ which leads to a set of clauses denoted by $\mathcal{C}^f_\Sigma$. Assuming $\mathtt{M}_\Sigma \subseteq \mathcal{C}^f_\Sigma$ is the subset of minimal[2] clauses composed of answer literals only, we have:

Corollary 1. *$\mathcal{M} = \{\alpha_1, \cdots, \alpha_k\}$ is a pesudo-minimal module for $\mathcal{O}$ and Σ iff $(\wedge_{i=1}^k l_{\alpha_i}) \rightarrow \perp \in \mathtt{M}_\Sigma$.*

For instance, in Example 10, we obtain $\mathtt{M}_\Sigma = \{l_{\alpha_1} \wedge l_{\alpha_2} \wedge l_{\alpha_4} \wedge l_{\alpha_5} \rightarrow \perp\}$ by resolution over $\mathcal{C}_\Sigma \cup \{\neg l_\Sigma\}$. Therefore, the only pseudo-minimal module for $\mathcal{O}$ and Σ is $\{\alpha_1, \alpha_2, \alpha_4, \alpha_5\}$, which is also the unique minimal module for $\mathcal{O}$ and Σ since the corresponding forest $F^\mathcal{O}_\Sigma$ is finite.

Optimization

As mentioned in Example 10, we can reduce the size of $F^\mathcal{O}_\Sigma$ by ignoring some redundant trees. For example, we can ignore those trees that do not contribute to the generation of axioms in $\mathcal{U}(F^\mathcal{O}_\Sigma)$. Furthermore, we can ignore some sub-f-trees based on the following result:

Corollary 2. *Theorem 1, 2 and 3 still hold if we ignore the f-trees $\mathtt{t}^+ \in F^\mathcal{O}_\Sigma$ that satisfies (i) t^+ has an edge, starting from the root $\mathtt{r}(\mathtt{t}^+)$, which is not a $\exists r$-edge;* ***or*** *(ii) $\mathtt{t}^+$ is a proper sub-tree of a f-tree $\mathtt{t}_1^+ \in F^\mathcal{O}_\Sigma$ such that $\mathtt{r}(\mathtt{t}) = \mathtt{r}(\mathtt{t}_1^+)$ and $\not\models \mathtt{C}_{\mathtt{t}^+} \equiv \mathtt{C}_{\mathtt{t}_1^+}$.*

In Example 10, $\mathtt{t}_1^+$ has 6 different sub-trees, but only $\mathtt{t}_2^+$ satisfies $\models \mathtt{C}_{\mathtt{t}_1^+} \equiv \mathtt{C}_{\mathtt{t}_2^+}$. Therefore, we can ignore the other 5 sub-trees and thus now $F^\mathcal{O}_\Sigma = \{\mathtt{t}_1^+, \mathtt{t}_2^+, \mathtt{t}_1^-\}$.

Evaluation

We implemented a prototype `ForMod`[3] of our algorithm in Python and evaluated it over three real-world ontologies:

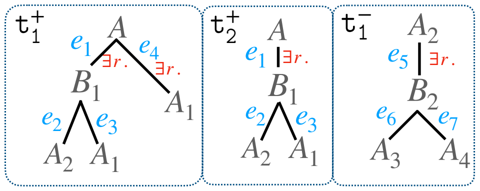


Figure 5: Trees in $F^\mathcal{O}_\Sigma$

	Σ^{sn16}_{50}	Σ^{sn16}_{100}	Σ^{nci}_{50}	Σ^{nci}_{100}	Σ^{sn21}_{50}	Σ^{sn21}_{100}
`Zoom`	57.1	32.5	79.0	57.3	-	-
`ForMod`	**83.8**	**79.2**	**92.3**	**74.2**	**98.9**	**88.1**

Table 2: Success rate (%)

Snomed CT (versions Jan 2016 and Jan 2021) and NCI (version 16.03d)[4]. We denote them as *sn16, sn21, nci*, respectively. Here, *sn16* and *nci* are two $\mathcal{EL}$-terminologies containing 317891 and 165341 axioms respectively. And *sn21* is an $\mathcal{EL}$-ontology with 362638 axioms. All the experiments run on a machine with an Intel Xeon Core 4 Duo CPU 2.50 GHz with 64 GiB of RAM.

For each ontology $\mathcal{O}$, we run the experiments over 2 sets $\Sigma^\mathcal{O}_n$ of 1000 randomly generated signatures, where each signature has n concepts ($n \in \{50, 100\}$) and 10 roles.

Pseudo-Minimal Modules

Recall that `Zoom` (Chen et al. 2017) is the state-of-the-art algorithm that computes all the minimal modules but only for $\mathcal{EL}$-terminologies. First, we compare all pseudo-minimal modules computed by `ForMod` with all minimal modules computed by `Zoom`.

For each signature, we set the total run-time limit of 600s. The success rates (i.e., the percentage of completed experiments within the time limit) of `ForMod` and `Zoom` are summarized in Table 2. We can see that the success rate of `ForMod` is from 13.3% to 46.7% higher than `Zoom`. Note that `Zoom` does not work for *sn21*, which is not an $\mathcal{EL}$-terminology.

Table 3 summarizes the time-cost comparison over signatures that are solved successfully by both `ForMod` and `Zoom`. We highlight that, for these signatures, the corresponding forests are indeed finite. Therefore, the pseudo-minimal modules are indeed minimal modules by Theorem 2. According to Table 3, `ForMod` is 99.79 times faster than `Zoom` on average. Note that, as discussed in (Chen et al. 2017), `Zoom` spends most of its running time (94.6% on average) on computing justifications using `Beacon` (Arif et al. 2016), which is less efficient than the resolution we use. However, even if we ignore the time cost of `Beacon` (i.e. only consider 5.4% computation time of `Zoom`), `ForMod` is still 5.67, 7.48, 3.72, 5.33 times faster than `Zoom` on average for Σ^{sn16}_{50}, Σ^{sn16}_{100}, Σ^{nci}_{50}, Σ^{nci}_{100}, respectively.

Second, in our experiments, there are 66 signatures in $\Sigma^{nci}_{50} \cup \Sigma^{nci}_{100}$ for which $F^\mathcal{O}_\Sigma$ is infinite, and 43 of them are

[2] c_1 is smaller than c_2 if all literals of c_1 are in c_2.

[3] https://gitlab.lisn.upsaclay.fr/yang/formod

[4] https://www.snomed.org,http://evs.nci.nih.gov

Time(s)	ForMod	Zoom
Σ^{sn16}_{50}	**17.20 / 1.09 / 1.77 / 1.73**	558.76 / 34.85 / 186.04 / 143.53
Σ^{sn16}_{100}	**9.75 / 1.29 / 2.18 / 2.10**	563.24 / 71.38 / 302.24 / 294.19
Σ^{nci}_{50}	**8.89 / 0.74 / 1.32 / 1.28**	560.89 / 7.81 / 91.08 / 60.42
Σ^{nci}_{100}	**12.47 / 0.89 / 1.47 / 1.42**	576.35 / 15.49 / 145.32 / 105.92

Table 3: Time cost (max / min / mean / median)

Module size	$\bot\top^*$-module	*pseudo-minimal module*
Σ^{nci}_{50}	7499 / 6340 / 7126.24	80 / 12 / 31.06
Σ^{nci}_{100}	8091 / 1936 / 7317.31	111 / 9 / 52.81

Table 4: Signatures with infinite $F^{\mathcal{O}}_{\Sigma}$ (max / min / mean)

solved within the time limit by ForMod, but all of them are time-out for Zoom. So we compare pseudo-minimal modules of these signatures with $\bot\top^*$-modules implemented by OWL API (Grau et al. 2008) instead. The result is summarized in Table 4. We observe that the pseudo-minimal modules are still very concise in terms of size and are significantly smaller than the $\bot\top^*$-modules.

Complete Modules

Here, we compare our complete modules generated by ForMod with $\bot\top^*$-modules. Note that, for all signatures tested, the corresponding complete modules and $\bot\top^*$-modules are all computed within $32s$, i.e., the success rate is 100%. The size comparison of those modules is shown in Table 5. We can see that the size of complete modules is significantly smaller than that of $\bot\top^*$-modules (103.5 times smaller on average) for all ontologies.

Table 6 summarizes the time cost comparison between complete modules and $\bot\top^*$-modules. It shows that the computation of complete modules is faster than that of $\bot\top^*$-modules except for the maximal time cost.

Related Work

For computing deductive modules, the paper (Chen et al. 2017) developed a recursive algorithm Zoom, which computes minimal modules by computing *subsumption justifications* based on a *logical difference* algorithm proposed in (Ludwig and Walther 2014). Zoom can always compute minimal modules no matter whether a UI exists or not. However, Zoom only works for $\mathcal{EL}$-terminologies since the logical difference algorithm only works for $\mathcal{EL}$-terminologies.

In (Koopmann and Chen 2020), deductive modules are computed w.r.t. $\mathcal{ALCH}$, which is more expressive than $\mathcal{EL}$. The authors compute a single deductive module by collecting all axioms contributing to the generation of a UI obtained by adapting lethe (Koopmann 2020). When a UI exists, they further compute a single minimal module by removing redundant axioms over the above obtained deductive module using a reasoner. We do not compare their method directly with ForMod because the UI and minimal modules are different if different languages (e.g., $\mathcal{EL}$ or $\mathcal{ALCH}$) are used as the underlying semantics, even for the same ontology and

Module Size	$\bot\top^*$-module	complete module	pseudo-minimal
Σ^{sn16}_{50}	6628.21	74.53	8.80
Σ^{snt16}_{100}	13159.89	140.68	20.97
Σ^{nci}_{50}	5560.64	52.02	24.36
Σ^{nci}_{100}	7327.67	73.91	37.53
Σ^{sn21}_{50}	4384.08	88.48	7.60
Σ^{snt21}_{100}	15845.43	81.28	16.11

Table 5: Mean size of different deductive modules

Time(s)	complete module (ForMod)	$\bot\top^*$-module (OWL API)
Σ^{sn16}_{50}	27.10 / **1.09** / **2.92** /**1.95**	**9.36** / 5.60 / 6.55 / 6.45
Σ^{sn16}_{100}	31.60 / **1.29** / **4.76** / **2.60**	**9.68** / 5.76 / 6.74 / 6.65
Σ^{nci}_{50}	3.45 / **0.74** / **1.32** / **1.30**	**2.88** / 1.16 / 1.90 / 1.97
Σ^{nci}_{100}	23.49 / **0.89** / **1.46** / **1.42**	**2.90** / 1.28 / 1.98 / 1.97
Σ^{sn21}_{50}	28.06 / **1.34** / **2.62** / **2.27**	**12.90** / 3.55 / 8.57 / 8.88
Σ^{sn21}_{100}	17.71 / **2.10** / **3.48** / **3.17**	**14.19** / 3.99 / 9.28 / 9.71

Table 6: Time cost (max / min / mean / median)

signature. Moreover, notice that their method always produces only one deductive module while ours computes all (pseudo-)minimal modules.

Recall that Theorem 1 provides a way to compute UI based on forests $F^{\mathcal{O}}_{\Sigma}$ although we do not implement it. There are mainly two different approaches for computing UI: **(i)** *Forgetting-based approach* like lethe and Fame (Zhao and Schmidt 2018). They compute a UI by forgetting all the concepts and roles outside a signature Σ. **(ii)** *Generation approach* like NUI (Konev, Walther, and Wolter 2009). NUI works only for $\mathcal{EL}$-terminology and when $F^{\mathcal{O}}_{\Sigma}$ is finite (has no Σ-loop in their case). It is shown in (Chen et al. 2019) that NUI is much more efficient than lethe and Fame on $\mathcal{EL}$-terminologies. Moreover, generating UI from $F^{\mathcal{O}}_{\Sigma}$ underlined by Theorem 1 can be regarded as a generalization of NUI from $\mathcal{EL}$-terminologies to $\mathcal{EL}$-ontologies:

Proposition 3. *Assume that $\mathcal{O}$ is an $\mathcal{EL}$-terminology, Σ is a signature, and $F^{\mathcal{O}}_{\Sigma}$ is finite. Let $F \subseteq F^{\mathcal{O}}_{\Sigma}$ be the subset consisting of all b-trees and maximal f-trees. Then, the UI for $\mathcal{O}$ and Σ computed by NUI is equivalent to $\mathcal{U}^1_{\Sigma}(F) \cup \mathcal{U}^2_{\Sigma}(F)$ (recall Definition 8).*

Conclusion and Future Work

In this paper, we present pseudo-minimal modules and complete modules for $\mathcal{EL}$-ontologies and a SAT-based algorithm ForMod to compute them. Our method is based on a novel notion of the forest, which enables to capture all the entailments over a given signature. The experiments on real-world ontologies validated the efficiency of our proposal as well as the quality of our deductive modules.

As the next step, we plan to investigate how to generalize our ideas to more expressive ontologies such as $\mathcal{EL}^+$ and $\mathcal{ALC}$. Also, we are interested in investigating such ideas on module notions that are not necessarily deductive modules (e.g., semantic modules (Konev et al. 2008)).

References

Arif, M. F.; Mencía, C.; Ignatiev, A.; Manthey, N.; Peñaloza, R.; and Marques-Silva, J. 2016. BEACON: An Efficient SAT-Based Tool for Debugging $\mathcal{EL}^+$ Ontologies. In *International Conference on Theory and Applications of Satisfiability Testing*, 521–530.

Arif, M. F.; Mencía, C.; and Marques-Silva, J. 2015. Efficient axiom pinpointing with EL2MCS. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, 225–233.

Calvanese, D.; Giacomo, G. D.; and Lenzerini, M. 1999. Reasoning in Expressive Description Logics with Fixpoints based on Automata on Infinite Trees. In *International Joint Conference on Artificial Intelligence*, 84–89.

Chen, J.; Alghamdi, G.; Schmidt, R. A.; Walther, D.; and Gao, Y. 2019. Ontology extraction for large ontologies via modularity and forgetting. In *International Conference on Knowledge Capture*, 45–52.

Chen, J.; Ludwig, M.; Ma, Y.; and Walther, D. 2017. Zooming in on Ontologies: Minimal Modules and Best Excerpts. In *16th International Semantic Web Conference*, 173–189.

Gatens, W.; Konev, B.; and Wolter, F. 2014. Lower and Upper Approximations for Depleting Modules of Description Logic Ontologies. In *European Conference on Artificial Intelligence*, 345–350.

Grau, B. C.; Horrocks, I.; Kazakov, Y.; and Sattler, U. 2008. Modular reuse of ontologies: Theory and practice. *Journal of Artificial Intelligence Research*, 31: 273–318.

Jiménez-Ruiz, E.; Grau, B. C.; Sattler, U.; Schneider, T.; and Berlanga, R. 2008. Safe and economic re-use of ontologies: A logic-based methodology and tool support. In *European Semantic Web Conference*, 185–199.

Kazakov, Y.; and Skočovskỳ, P. 2018. Enumerating justifications using resolution. In *International Joint Conference on Automated Reasoning*, 609–626.

Konev, B.; Ludwig, M.; Walther, D.; and Wolter, F. 2012. The Logical Difference for the Lightweight Description Logic EL. *J. Artif. Intell. Res.*, 44: 633–708.

Konev, B.; Lutz, C.; Walther, D.; and Wolter, F. 2008. Semantic Modularity and Module Extraction in Description Logics. In *European Conference on Artificial Intelligence*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, 55–59.

Konev, B.; Walther, D.; and Wolter, F. 2009. Forgetting and Uniform Interpolation in Large-Scale Description Logic Terminologies. In *21st International Joint Conference on Artificial Intelligence*, 830–835.

Koopmann, P. 2020. LETHE: Forgetting and uniform interpolation for expressive description logics. *KI-Künstliche Intelligenz*, 34(3): 381–387.

Koopmann, P.; and Chen, J. 2020. Deductive Module Extraction for Expressive Description Logics. In Bessiere, C., ed., *International Joint Conference on Artificial Intelligence*, 1636–1643.

Koopmann, P.; and Schmidt, R. A. 2013. Forgetting Concept and Role Symbols in ALCH-Ontologies. In *Logic for Programming, Artificial Intelligence, and Reasoning - 19th International Conference*, 552–567.

Ludwig, M.; and Walther, D. 2014. The Logical Difference for $\mathcal{ELH}^{\mathrm{r}}$-Terminologies using Hypergraphs. In Schaub, T.; Friedrich, G.; and O'Sullivan, B., eds., *European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, 555–560. IOS Press.

Lutz, C.; Piro, R.; and Wolter, F. 2010. Enriching EL-concepts with greatest fixpoints. In *European Conference on Artificial Intelligence*, 41–46.

Manthey, N.; Peñaloza, R.; and Rudolph, S. 2016. Efficient Axiom Pinpointing in EL using SAT Technology. In *Description Logics*.

Nikitina, N.; and Rudolph, S. 2014. (Non-) Succinctness of uniform interpolants of general terminologies in the description logic EL. *Artificial Intelligence*, 215: 120–140.

Sattler, U.; Schneider, T.; and Zakharyaschev, M. 2009. Which kind of module should I extract? *Description Logics*, 477: 78.

Yang, H.; Ma, Y.; and Bidoit, N. 2022. Hypergraph-Based Inference Rules for Computing $\mathcal{EL}^+$-Ontology Justifications. In *International Joint Conference on Automated Reasoning*, 310–328.

Zhao, Y.; and Schmidt, R. A. 2018. FAME: an automated tool for semantic forgetting in expressive description logics. In *International Joint Conference on Automated Reasoning*, 19–27.