Computing EL minimal modules: a combined approach
Hui Yang, Yue Ma, Nicole Bidoit

To cite this version:
Hui Yang, Yue Ma, Nicole Bidoit. Computing EL minimal modules: a combined approach. BDA2022: “Gestion de Données – Principes, Technologies et Applications”, Oct 2022, Clermont-Ferrand, France. hal-04429001

HAL Id: hal-04429001
https://hal.science/hal-04429001
Submitted on 6 Feb 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Computing $\mathcal{EL}$ minimal modules: a combined approach

Hui Yang
yang@lisn.fr
LISN, Univ. Paris-Sud, CNRS,
Université Paris-Saclay
Orsay, France

Yue Ma
ma@lisn.fr
LISN, Univ. Paris-Sud, CNRS,
Université Paris-Saclay
Orsay, France

Nicole Bidoit
nicole.bidoit@lisn.fr
LISN, Univ. Paris-Sud, CNRS,
Université Paris-Saclay
Orsay, France

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, minimal modules have been introduced to provide concise ontology views. However, computing such minimal modules remains highly time-consuming. In this paper, we design a new method combining graph and SAT techniques, to address the computation cost of minimal modules. Our approach first introduces a new abstract notion of invariant to characterize sub-ontologies sharing the same logical information. Then, we construct a finite invariant using graph representations of $\mathcal{EL}$ ontologies. Finally, we develop a SAT-based algorithm to compute minimal modules using this invariant. Finally, in some cases, when the computation is still too time-consuming, we provide approximations of minimal modules. Our experiments on real-world ontologies outperform the state-of-the-art algorithm. Our algorithm provides more compact approximate results than the well-known locality-based modules without losing efficiency.

CCS CONCEPTS
• Computing methodologies → Description logics.

KEYWORDS
Ontology, Description Logic EL, Minimal module

1 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, called module extraction, is to extract sub-ontologies related to the user interests. For example, the well-known biomedical ontology Snomed CT contains 300,000+ axioms. By module extraction, we could provide doctors with small sub-ontologies of Snomed CT based on symptoms to establish a diagnostic. Module extraction has also been used for different problems, like ontology debugging [1], re-use [10], and forgetting [17].

We can distinguish two classes of module extraction. In the first class, methods such as MEX-module [14], AMEX-module [8] and locality-based module [22] are efficient however they are not accurate in the sense that they provide sub-ontologies containing many unnecessary terms. In the second class, methods such as minimal module [6] and justification [5] are precise and provide minimal results, but they suffer from high complexity [21] and are time consuming in practice. We engage these issues by concentrating on computing minimal modules, which is also investigated as minimal deductive module [15] in [16].

Minimal modules provide concise information focusing on one’s interest. They are specified as the minimal sub-ontologies that preserve all the logical entailments over a particular set of items called the signature. The state-of-the-art method [6] for computing all minimal modules of an $\mathcal{EL}$ terminology is based on subsumer and subsume simulations following the idea from [7, 18]. For the more expressive language $\mathcal{ALCH}$, uniform interpolation [19] has been investigated to compute one minimal module at a time [16].

In this paper, we propose a new efficient method for computing all minimal modules of an $\mathcal{EL}$ terminology based on graph representations of ontologies. This method is inspired by the SAT-based approach developed to compute justifications. Justifications are minimal sub-ontologies that preserve one logical entailment. The SAT-based methods [2, 3, 11, 20] are the state of the art methods for computing justifications for specific languages such as $\mathcal{EL}$-ontology. Their main idea is to translate the computation of justifications to a SAT problem and then solve it using SAT tools.

Our contribution is three-fold: (i) we introduce an abstract notion of invariant; a given invariant is meant to capture sub-ontologies sharing the same logical information and, here, we provide a finite invariant specifically relevant for minimal module extraction; (ii) we develop a SAT-based method for computing minimal modules based on our invariant; (iii) to validate the efficiency of our method, we implement a prototype GIMM which outperformed the state-of-the-art algorithm [6] on real-world ontologies.

Building our finite invariant relies on the hyper-graph and direct-graph representations of $\mathcal{EL}$ ontology. Our method also provides an approximation result for minimal modules. This may be helpful, for example, if there are too many different minimal modules making impossible enumerating all of them. An empirical comparison with the locality-based module method implemented by OWL API [9] shows that the GIMM approximation is promising: it is more concise without loss of efficiency.

Due to space limitations, some proofs and details about experiments as well as the description of prototype GIMM are available here: shorturl.at/fwW49.

2 PRELIMINARY
2.1 Ontology and minimal module
In this paper, we focus on the $\mathcal{EL}$-ontology defined as follows. Given finite sets of atomic concepts $N_C = \{A, B, \cdots\}$ and atomic roles $N_R = \{r, s, \cdots\}$, the set of $\mathcal{EL}$ concepts $C$ and axioms $\alpha$ are built by the grammar rules (i) $C ::= \top | A | C \sqsubseteq \exists r.C$ or (ii) $\alpha ::= C = C | C \sqsubseteq C$. We denote by $\text{sig}(C)$ the atomic concepts and roles that compose $C$. For example, $\text{sig}(\exists r.(B_1 \sqcap B_2)) = \{r, B_1, B_2\}$.

An $\mathcal{EL}$-ontology $O$ is a finite set of $\mathcal{EL}$-axioms. An interpretation $I = (\Delta^I, \cdot^I)$ of $O$ consists of a non-empty set $\Delta^I$ and a mapping
from atomic concepts $AeN_C$ to a subset $A^f \subseteq A^f$ and from roles $r \in R_N$ to a subset $r^f \subseteq A^f \times A^f$. For a concept C built from the grammar rules, we define $C^f$ inductively by: $(T^f) = A^f$, $(\land D^f) = C^f \cap D^f$, $(\lor E^f) = \{a, b \mid C^f \cap D^f \}$. An interpretation is a model $O$ if it is compatible with all axioms in $O$, i.e., for all $C \subseteq D$, $C \subseteq D$, we have $C^f \subseteq D^f$, $C^f \subseteq D^f$ respectively.

An $E_L$-ontology $O$ is normalized if all its axioms are of the form $A= \{B \mid \neg r \cdot m \cdot m \cdot m \} \land B$, where $\neg \{e \}, A, B \in N_C, m \in R_N$.

Every $E_L$-ontology can be normalised in polynomial time by introducing new atomic concepts. Moreover, we say the $O$ is a terminology if any atomic concept $A$ appears at most once on the left-hand side of axiom in $O$.

We say $O \models \alpha$ where $\alpha$ is an axiom if and only if each model of $O$ is compatible with $\alpha$.

**Definition 1. (Justification)** Given $O$ such that $O \models A \subseteq B$. A justification of $A \subseteq B$ in a minimal subset $J \subseteq O$ such that $J \models A \subseteq B$.

Given two ontologies $O_1, O_2$ and a signature $\Sigma \subseteq N_C \cup R_N$, the logical difference $^1$ between $O_1, O_2$ over $\Sigma$ is the set:

$$\Delta_2(O_1, O_2) = \{ \alpha | \text{sign}(\alpha) \subseteq \Sigma, O_1 \models \alpha, O_2 \models \neg \alpha, \{1, 2\} \}$$

**Definition 2. (Minimal module)** A sub-ontology $M \subseteq O$ is a minimal module for $\Sigma$ if $\Delta_2(M, O) = \emptyset$ and there is no $M' \subseteq M$ such that $\Delta_2(M', O) = \emptyset$.

When $O_1, O_2$ are terminologies, we have:

**Theorem 1.** [13] Given two terminologies $O_1, O_2$ and a signature $\Sigma$. If $\Delta_2(O_1, O_2)$ is not empty, then there exists an axiom in $\Delta_2(O_1, O_2)$ of the form $A \subseteq C \subseteq A$, where $A$ is an atomic concept.

**Example 1.** For terminology $O_0 = \{B_1 \subseteq A_1 \cap A_2, B_2 \subseteq A_1, B_3 \subseteq A_3 \}

\beta_4 \subseteq A_2, \beta_5 \subseteq A_3 \cap A_4, \beta_6 \subseteq A_2 \cap A_5, \beta_7 \subseteq B_4 \cap B_5 \}

we have $O_0 \models B_2 \subseteq A_1$, and $J_0 = \{B_1 \}$ is the only justification for $B_2 \subseteq A_1$.

If we consider a signature $\Sigma_0 = \{B_1, B_2, B_3, B_4, B_5 \}$, there is only one minimal module for $\Sigma_0$: $M_0 = O_0 \setminus \{B_2 \}$.

### 2.2 Hyper-graph

In this paper, we associate $E_L$-ontologies with hyper-graphs. A (directed) hyper-graph $[4] \mathcal{H} = \{V, E\}$ consists of a node set $V = \{v_1, v_2, \ldots, v_n\}$ and an edge set $E = \{e_1, e_2, \ldots, e_n\}$, $e_i = \langle T(e_i), f(e_i) \rangle$, where $T(e_i) \subseteq V$ is a subset and $f(e_i) \in V$ is a node.

**Definition 3.** Given a hyper-graph $\mathcal{H} = \{V, E\}$, assume $\Sigma \subseteq V$ and $w \in V$. A hyper-path from $S$ to $v_1$ is a sequence $h = \{e_1, e_2, \ldots, e_n\}$ of hyper-edges satisfying:

1. $f(e_n) = \{v_1\}$
2. For $i=1, \ldots, n, T(e_i) \subseteq V(f(e_i), \ldots, f(e_{i+1}))$
3. For $i=1, \ldots, n, f(e_i) \in \bigcup_{j<i} T(e_j)$. If $w \in T(e_1)$, we say $h$ is a loop. If $h$ does not contain any loop, we say $h$ is loop-free.

For simplicity, in the following we also write edges in $\{\{\{v_1\}, v_2\}, \{v_1, \ldots, v_k\}\}$ as $v_1 \rightarrow v_2, v_1, \ldots, v_k \rightarrow v$. Notice that our definition of logical difference is symmetric and generalizes the classical one [12].

### 3 CONSTRUCTION OF INVARIANT

In this section, we propose the notion of invariant that provides means to characterize terminologies sharing the same logical information with respect to a given signature.

**Definition 4.** A map $\text{Inv}$ taking as input a terminology and a signature is an invariant if for any two terminologies $O_1 \subseteq O_2$,

$$\Delta_2(O_1, O_2) = \emptyset \iff \text{Inv}(O_1, \Sigma) = \text{Inv}(O_2, \Sigma).$$

Note that, the elements of $\text{Inv}(O, \Sigma)$ can be of any kinds. For example, a trivial invariant can be defined by the set of axioms $\{\alpha \models \text{sign}(\alpha) \subseteq \Sigma\}$. Because this trivial invariant is infinite and thus has no practical interest, we introduce next another invariant (a finite one) set whose elements are triples $(A, S_1, S_2)$ where $A$ is a concept and $S_1$ and $S_2$ are finite sets.

#### 3.1 Main idea

Similarly to [6], our construction of finite invariant is based on Theorem 1. Given two terminologies $O_1, O_2$, a signature $\Sigma$ and a concept $A$, we consider:

$$\Delta_2^{\Sigma}(O_1, O_2) = \{e \subseteq A \subseteq C \subseteq D \leq O \}$$

**Definition 5.** A map from the tuple $(O, \Sigma, A)$ to a set $\text{Inv}^\Sigma(O, \Sigma, A)$ is a right invariant if for any two terminologies $O_1 \subseteq O_2$, we have:

$$\Delta_2^{\Sigma}(O_1, O_2) = \emptyset \iff \text{Inv}^\Sigma(O_1, \Sigma, A) = \text{Inv}^\Sigma(O_2, \Sigma, A).$$

The left invariant $\text{Inv}_L(O, \Sigma, A)$ is defined in a dual manner based on $\Delta_2^{\Sigma}(O_1, O_2)$.

Then as a corollary of Theorem 1, we have:

**Corollary 1.** If $\text{Inv}^\Sigma$ (resp. $\text{Inv}^L$) is a right (resp. left) invariant, then the map $\text{Inv}$ defined below is an invariant:

$$\text{Inv}(O, \Sigma) = \{A, \text{Inv}^\Sigma(O, \Sigma, A), \text{Inv}^L(O, \Sigma, A)\} \subseteq \Sigma.$$ 

Thus, to construct a finite invariant, it is enough to build a finite right invariant and a finite left invariant, at first.

#### 3.2 Building a finite right invariant

Given a terminology $O$, a signature $\Sigma$ and a concept $A$, in the following we associate a hyper-graph $\mathcal{H}_O$ to $O$ and then we construct a finite invariant using the hyper-paths of $\mathcal{H}_O$.

**2.3.1 (1) Hyper-graph and hyper-paths.** The hyper-graph $\mathcal{H}_O = (N, E)$ associated to $O$ consists of the node set $N = \{N_A | A \in N_C\}$ and the edge set:

$$E = \{N_{A_1} \rightarrow N_{A_2} | A_1 \subseteq A_2 \subseteq N_{A_1} \} \cup \{N_{A_1} \rightarrow B \subseteq B \in \Sigma \} \subseteq \{N_{A_1} \rightarrow B | B \subseteq \Sigma \} \subseteq \{N_{A_1} \rightarrow B | B \subseteq \Sigma \}.$$

where $N_{A_1} \rightarrow N_{A_2}$ is an edge with the index $r \in N_{A_2}$.

Among the hyper-paths in $\mathcal{H}_O$, we specifically consider the set $\mathcal{P}_N^\Sigma$ of all the hyper-paths $h$ from $\Sigma = \{N_B | B \in \Sigma\}$ to $A_N$ in $\mathcal{H}_O$ such that:

(1) For any edge $e \in h$, either $e$ has no index or its index $r \in \Sigma$.
(2) $h$ does not contain trivial loops of the form:
\[ \{ (N_{A1}, \cdots, N_{A_n}) \to N_{B1}, N_{B2} \to N_{A1} \} \text{ for some } i \in \{1, n\}; \]

(3) $h$ does not contain repeated edges.

Item 3 above implies that the length of an hyper-path $h \in P^O_{\Sigma_A}$ is at most $\#|E|$ and therefore $P^O_{\Sigma_A}$ is finite.

**Example 2 (Example 1 cont’d).** The hyper-graph associated with the ontology $O_0$ is shown in Figure 1. Notice that $e_0 = \{N_{A1}, N_{A_n}\} \to N_{B1}$ is the only complex edge in this hyper-graph. There are two hyper-paths in $P^O_{\Sigma_B}$: $h_1 = [e_1, e_4, e_3, e_5, e_6]$, and $h_2 = [e_2, e_4, e_3, e_5, e_6]$.

![Figure 1: $\mathcal{H}^{O_0}$, concepts and roles in $\Sigma_0$ are in red.](image)

Note that $l_1 = [e_6, e_5]$, $l_2 = [e_6, e_10]$ are trivial loops. Thus any hyper-path containing $l_1$ or $l_2$ is excluded from $P^O_{\Sigma_B}$.

(2.2) Our finite right invariant. Each hyper-path $h \in P^O_{\Sigma_A}$ can be interpreted as an ontology:
\[ O_h := \{ A_1 \cap \cdots \cap A_n \mid \{ N_{A1}, \cdots, N_{A_n} \} \to N_{A} \in h \} \cup \{ A \cap B \mid N_{A} \to N_{B} \in h \} \cup \{ \exists r. A \cap B \mid N_{A} \to N_{B} \in h \}. \]

Indeed, the existence of a hyper-path $h$ determines whether $O \subseteq \Sigma_A$ in the following sense:

**Theorem 2.** $O \subseteq \Sigma_A, sig(C) \subseteq \Sigma$ iff there exists a hyper-path $h \in P^O_{\Sigma_A}$ such that $O_h \subseteq \Sigma_A$.

Now, one could expect that $P^O_{\Sigma_A}$ is a good candidate for defining a right invariant. However, we still need to refine it in order to avoid redundant and repetitive elements. Let us illustrate this with Example 3 for redundancy, and with Example 4 for repetition.

Before, let us consider $I(h)$ to be the set of axioms entailed by $O_h$:
\[ I(h) := \{ C \subseteq A \mid O_h \subseteq \Sigma_A, sig(C) \subseteq \Sigma \}. \]

**Example 3 (Example 1 and 2 cont’d).** For terminology $O_0$, we have $P^O_{\Sigma_B} = \{ h_1, h_2 \}$. Note that $h_2$ is one of the axioms corresponding to $\{ h_1, h_2 \}$ but $h_2$ does not belong to the unique minimal module $M_0$. The reason is that $I(h_1) \subseteq I(h_2)$, which means that $h_2$ provides strictly more logical information than $h_1$. Therefore, $h_1$ is redundant and should not be considered when computing the minimal module $M_0$.

**Example 4.** Assume $O_1 := \{ \beta_1, \beta_2 : A \subseteq B, \beta_3 : B \subseteq A \} \text{ and } \Sigma_1 = \{ B_1, B_2 \}$. Then
\[ P^O_{\Sigma_1} := \{ h_1 \mid N_{B2} \to N_{A1}, N_{A2} \to N_{B1} \}. \]

We have $I(h_1) \subseteq I(h_2)$, meaning that $h_1, h_2$ provide the same logical information and thus a repetition. There are two minimal modules for $\Sigma_1$: $M_1 = \{ h_1 \}$ and $M_2 = \{ h_2, h_3 \}$ corresponding to $h_1$ and $h_2$, respectively. Indeed here, a minimal module should “contain” one and only one of $h_1, h_2$.

Formally, for any two hyper-paths $h_1, h_2 \in P^O_{\Sigma_A}$, we say:
\begin{itemize}
  \item $h_1$ is redundant if $I(h_1) \subseteq I(h')$ for some $h' \in P^O_{\Sigma_A}$.
  \item $h_1, h_2$ are equivalent if $I(h_1) = I(h_2)$.
\end{itemize}

Next, to eliminate redundancy and repetition, we extract from $P^O_{\Sigma_A}$ the set $\{ h \mid h \in P^O_{\Sigma_A} \text{ is not redundant} \}$, where $\tilde{h}$ denotes the class of all hyper-paths equivalent to $h$. Finally, our finite right invariant is defined by:

**Theorem 3.** The map $Inv^r$ defined below is a right invariant:
\[ Inv^r(O, \Sigma, A) := \{ \tilde{h \mid h \in P^O_{\Sigma_A} \text{ is not redundant} \}. \]

Clearly, $Inv^r(O, \Sigma, A)$ is finite since $P^O_{\Sigma_A}$ is a finite set.

**Example 5 (Example 3 cont’d).** Since $I(h_1) \subseteq I(h_2)$, the hyper-path $h_1$ is redundant and $Inv^r(O_0, \Sigma_0, B_3) = \{ \tilde{h} \}$.

**3.2.3 (3) Checking equivalence and redundancy.** Computing our right invariant requires an algorithm for checking the equivalence and the redundancy of hyper-paths (i.e., whether $I(h_1) = I(h_2)$ or $I(h_1) \subseteq I(h_2)$), thus indeed, an algorithm for checking $I(h_1) \subseteq I(h_2)$.

The graph $G^O := (N, E)$ associated with $O$ consists of a node set $N := \{ N_A \mid A \in \Sigma \}$ and an edge set $E := \{ N_A \to N_B \mid O = A \subseteq B \}$.

One may consider to construct a finite left invariant using paths in $G^O$ as in Section 3.2. However, the paths in $G^O$ are not sufficient for instance to capture the entailment $O \models A \subseteq B \cap C$. Indeed, a path like $[N_A \to N_{A_1}, N_{A_1} \to N_{A_2}, N_{A_2} \to N_{A_3}]$ in $G^O$ corresponds to $O = A \subseteq B \cap C$, where $B$ is an atomic concept.

To overcome such limitation, we introduce clusters as unions of paths.

**Definition 6.** (Cluster) A cluster $t$ from $A$ to $\Sigma$, is a union of paths, which do not contain a loop twice$^2$, from $N_A$ to some $N_B, B \in \Sigma$ in $G^O$. We say $t$ is loop-free if it does not contain loops.

Let us define $T^O_{\Sigma_A}$ as the collection of all clusters from $A$ to $\Sigma$. Then $T^O_{\Sigma_A}$ is finite, because the length of a path without repetitive loops is bounded$^3$ and thus there are finitely many such paths.

**Example 6 (Example 1 cont’d).** For the terminology $O_0$, $G^O$ is shown in Fig 2. There are two paths $p_1 = [e_{10}, e_{13}, e_7], p_2 = [e_{10}, e_{13}, e_8]$ from $N_{B1}$ to some node in $\Sigma$ and three loop-free clusters $t_1 = p_1, t_2 = p_2, t_3 = p_1 \cup p_2$.

$^2$For example, $[N_1 \to N_2, N_2 \to N_1, N_1 \to N_2]$ contains the loop $[N_1 \to N_2, N_2 \to N_1]$ twice.

$^3$The upper bound is $(m+2)!$ if $G^O$ has $m$ edges since a path with length $m+1$ contains a loop, and thus are at most $m!$ such loops.
of the invariant;
(2) We add two clauses
\[ \land_{\forall \alpha \in \Inv (O, \Sigma, A)} (l_\alpha) \rightarrow t_\alpha^* \land t_{\forall \alpha \in \Inv (O, \Sigma, A)} (l_\alpha) \rightarrow t_\alpha^* \]
for each \( A \in \Sigma \);
(3) We add the clause \( \land_{\forall e \in \Sigma} (l_\epsilon) \rightarrow t_\epsilon^* \) (resp. \( \land_{\forall e \in \Sigma} (l_\epsilon) \rightarrow t_\epsilon^* \)), for each non-redundant \( h \epsilon \in \Inv (O, \Sigma, A) \) (resp. \( t \in T_{\Sigma, A} \)) and \( A \in \Sigma \);
(4) For each edge \( e \in \Sigma \) (resp. \( e \in \Sigma \)), where \( h \in H_{\Sigma, A} \) (resp. \( t \in T_{\Sigma, A} \)) is non-redundant and \( A \in \Sigma \):
(a) if \( e = N_A \rightarrow N_B \), then we add clause \( \land_{\forall \beta \in \Sigma} (l_\beta) \rightarrow l_\epsilon \) for each justification \( f \) of \( A \subseteq B \);
(b) if \( e = N_A \rightarrow N_B \) or \( \{ N_A, \ldots, N_{n} \} \rightarrow N_B \), then we add the clause \( l_\beta \rightarrow l_\epsilon \), where \( \beta \in \Sigma \) is the axiom corresponding to \( e \).

The answer literals are defined as the literals \( l_\beta \) such that \( \beta \in O \).

Step 3. We apply resolution over \( C_{\Sigma} \cup \{ \Sigma \rightarrow \emptyset \} \). Then let us consider \( \Sigma_0 \), the collection of resulting minimal\(^4\) clauses composed of answer literals only. From \( \Sigma_0 \), all minimal modules for \( \Sigma \) are extracted easily as stated in Theorem 6.

**Theorem 6.** A subset \( M = \{ \beta_1, \ldots, \beta_k \} \subseteq \Sigma \) is a minimal module for \( \Sigma \) with respect to \( \emptyset \) if \( \land_{i=1}^{k} (l_{\beta_i}) \rightarrow \emptyset \in \Sigma_0 \).

**Remark 1.** In Step 2, Case 4(a), for the sake of simplicity, the added clauses are defined using justifications. Indeed, instead of computing justifications, we have developed a more efficient algorithm for computing these Horn clauses following a method similar to [11]. For details see supplementary materials.

**Example 8.** For our running example, the only non-empty right (left) invariants are \( \Inv^l (O_0, \Sigma_0, B_3) = \{ h_2 \} \), \( \Inv^l (O_0, \Sigma_0, B_3) = \{ \tau_3 \} \). Thus,
\[
\Inv (O_0, \Sigma_0) = \{(B_3, \{ h_2 \}, \{ \tau_3 \}) \cup \{(B_k, \emptyset, \emptyset) \mid k = 1, 2, 4, 5 \}
\]

The set of clauses \( C_{\Sigma_0} \) is shown in Table 1. We obtain
\[
\Sigma_{\Sigma_0} = \{ l_{\beta_1}, \land_{1 \leq i \leq 2} (l_{\beta_i}) \land (l_{\beta_1}) \land (l_{\beta_0}) \rightarrow \emptyset \}
\]

by resolution over \( C_{\Sigma_0} \cup \{ \Sigma_0 \rightarrow \emptyset \} \). Therefore, the only minimal module for \( \Sigma_0 \) is \( \Sigma_0 \setminus \{ l_{\beta_0} \} \).

Approximation. Now, let \( O^{ap} = \{ \beta \in \Sigma \mid l_{\beta} \text{ appears in } C_{\Sigma} \} \), then \( O^{ap} \) contains all the minimal modules of \( \Sigma \) and thus \( O^{ap} (O^{ap} \cup \Sigma) = \emptyset \). Next, we regard \( O^{ap} \) as an approximation of minimal modules. Indeed, \( O^{ap} \) can be computed from the invariant without Step 3.

**4.0.1 Optimization.** In general, there can be exponentially many hyper-paths (resp. clusters) in \( P_{\Sigma, A} \) (resp. \( T_{\Sigma, A} \)) with respect to the size of \( H^O \) (resp. \( G^O \)). In order to reduce the computation cost, we can restrict the search space to hyper-paths (resp. clusters) having particular forms.

\(^4c_1 \) is smaller than \( c_2 \) if all literals of \( c_1 \) are in \( c_2 \).
Table 1: Clause set $C_{\Sigma A}$

1. \( \land_{i=1}^{n} (F_{B_i} \land \neg F_{B_i}) \rightarrow F_{B_1} \);
2. \( F_{B_2} \rightarrow F_{B_1}, F_{B_3} \rightarrow F_{B_1}, 0 \rightarrow F_{B_k}, 0 \rightarrow F_{B_1}, k \in \{1, 2, 4, 5\} \);
3. \( e_3 \land e_5 \land e_7 \land e_9 \rightarrow e_{r_1} \);
4. (a) \( e_2 \land e_3 \land e_4 \land e_5 \land e_7 \land e_8 \land e_9 \rightarrow e_{r_1} \);
   (b) \( e_4 \land e_5 \land e_6 \land e_7 \land e_8 \land e_9 \rightarrow e_{r_1} \).

(1) Hyper-paths. We say a hyper-path $h$ over $\mathcal{H}^O$ is compact when it does not contain sub-paths of either form:
(1) \( \{ (N_{B_1}, \ldots, N_{B_n}) \rightarrow R_{B_1}, N_{B_2} \rightarrow N_{A_1} \} \), where $B \not\in \Sigma$.
(2) \( [N_{B_1} \rightarrow R_{B_2}, N_{B_3} \rightarrow N_{A_1}] \), where $B \not\in \Sigma$.

When $O$ is a terminology, the hyper-paths over $\mathcal{H}^O$ satisfy the following property:

Proposition 2. For any hyper-path $h \in \mathcal{P}_{\Sigma A}$, there exists a compact hyper-path $h' \in \mathcal{P}_{\Sigma A}$ such that $l(h) \subseteq l(h')$.

According to the above property, we can restrict $\mathcal{P}_{\Sigma A}$ to compact hyper-paths. Then, both Theorem 2 and 3 that define our right invariant still hold.

(2) Clusters. We say that a cluster $c = p_1 \cup p_2 \cup \cdots \cup p_m$ in $T_{\Sigma A}$ is compact if either $n = 1$ or the paths $p_1, \ldots, p_m$ have a common prefix path $[e_1, \ldots, e_k]$ with at least one indexed edge. For instance, in Example 6, the cluster $t_3$ is compact because $p_1, p_2$ share the prefix $[e_{10}, e_{13}]$ and $e_{13}$ is an indexed edge. Then, the following proposition holds:

Proposition 3. For $C$ with sig($C$)$\subseteq \Sigma$ of the form $B \lor \exists C_1, O_{\Sigma A} = A_{\Sigma C}$ iff there exists a compact cluster $c \in \mathcal{C}_{\Sigma A}$ such that $O_1 = A_{\Sigma C}$.

In proposition 3, we only consider concepts of the form $B \lor \exists C_1$, because any concept $C$ can be written as $B_1 \land \cdots \land B_n \lor \exists C_1 \land \cdots \land \exists B_m$, where $B_1, \ldots, B_n$ are atomic concepts. Thus, we know that $O \models A_{\Sigma C} \iff O \models A_{\Sigma B_1}$, $O \models A_{\Sigma \exists C_1}$, for any $1 \leq i \leq n, 1 \leq j \leq m$.

Therefore, by Proposition 3, we can require all clusters in $T_{\Sigma A}$ to be compact: Theorem 5 that defines our left invariant still holds.

5 EXPERIMENT

To evaluate the efficiency of our algorithm for computing minimal modules and the quality of our approximation results, we implemented in Python a prototype called GIMM. We evaluated GIMM using the $\mathcal{E}$,$\mathcal{L}$-fragment of two prominent biomedical ontologies: Snomed CT (version Jan 2016)$^4$, a terminology with 317891 axioms, and NCI (version 16.03d)$^5$, a terminology having 165341

---

Table 2: Running GIMM on the signature sets: statistics

<table>
<thead>
<tr>
<th>Signature (loop/trivial/non-trivial)</th>
<th>Number (max/min/mean)</th>
<th>Size (max/min/mean)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Sigma_{\text{nci}}^{50,10}$</td>
<td>497 / 26 / 1215</td>
<td>130 / 0 / 50</td>
</tr>
<tr>
<td>$\Sigma_{\text{nci}}^{100,10}$</td>
<td>747 / 29 / 797</td>
<td>130 / 0 / 50</td>
</tr>
<tr>
<td>$\Sigma_{\text{nci}}^{\infty}$</td>
<td>105 / 0 / 20</td>
<td>130 / 0 / 50</td>
</tr>
</tbody>
</table>

Time cost: GIMM vs. Zooms. First, we compare GIMM with the state-of-the-art algorithm [6], called Zooms. For each signature, we set the run-time limit to 600s. For simplicity, we regard signatures with loop as timed out samples for GIMM. Zooms is actually unable to solve any of these signatures.

As shown in Table 3, the successful rate of GIMM is between +12.8% and +46.6% higher than Zooms. As shown in Table 4, for the signatures solved by both algorithms, GIMM is 30 to 50 times faster than Zooms on average. The maximal time-cost of GIMM is close to or even smaller than the minimal time-cost of Zooms except for the dataset $\Sigma_{\text{nci}}^{50,10}$. Fig. 3 illustrates the time-cost of the two algorithms for all the signatures solved by them: GIMM is faster than Zooms on all the signatures.

One reason GIMM runs so fast is that the size of our invariants were usually small. On average, we have 0.95 non-redundant hyper-paths and 11.51 non-redundant clusters for each signature. For more than half of the cases, the set of non-redundant hyper-paths is empty. But in some rare cases, there exists up to 953 non-redundant hyper-paths.

---

Table 3: Successful rate

<table>
<thead>
<tr>
<th>Time(s)</th>
<th>GIMM</th>
<th>Zooms</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Sigma_{\text{nci}}^{50,10}$</td>
<td>71.3</td>
<td>71.3</td>
</tr>
<tr>
<td>$\Sigma_{\text{nci}}^{100,10}$</td>
<td>71.3</td>
<td>71.3</td>
</tr>
<tr>
<td>$\Sigma_{\text{nci}}^{\infty}$</td>
<td>71.3</td>
<td>71.3</td>
</tr>
</tbody>
</table>

Table 4: Time cost (max/min/mean/median)

<table>
<thead>
<tr>
<th>Time(s)</th>
<th>GIMM</th>
<th>Zooms</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Sigma_{\text{nci}}^{50,10}$</td>
<td>71.3</td>
<td>71.3</td>
</tr>
<tr>
<td>$\Sigma_{\text{nci}}^{100,10}$</td>
<td>71.3</td>
<td>71.3</td>
</tr>
<tr>
<td>$\Sigma_{\text{nci}}^{\infty}$</td>
<td>71.3</td>
<td>71.3</td>
</tr>
</tbody>
</table>
For the signatures solved by both algorithms, **GIMM** spent more time on computing right invariant (75.4% on average) than computing left invariant (21.4% on average). The reason is that the hyper-graph $H^O$ is always much bigger than the graph $G^O$. Thus extracting hyper-paths over $H^O$ is more difficult than extracting clusters over $G^O$, even if there are much more non-redundant clusters than hyper-paths on average.

**Approximation: GIMM vs. locality-based module.** Although **GIMM** is much more efficient than **Zooms**, as shown above, not all samples can be solved within 600s. For example, there are 25.7% timed out samples in $\Sigma_{nt}^{50,10}$ for **GIMM**. This is not surprising as computing minimal modules is a complex task in general. In these cases, **GIMM** was able to compute the approximation result $O^{AP}$ instead, except for 0.5% of $\Sigma_{nt}^{50,10}$ signatures, for which timing out took place during computing the invariant.

To evaluate the approximation results, we compared **GIMM** with the locality-based module as implemented by OWL API [9], which also provides an approximation of minimal modules. The results are summarized in Table 5. We can see that the sizes of our approximations ($O^{AP}$) are usually much smaller than that of locality-based modules (27 to 90 times smaller on average). On the other hand, the computation time of our approximation results using **GIMM** is comparable to the computation time of the locality-based modules except for a few cases (see the maximal time for $\Sigma_{nt}^{100,10}$).

### Table 5: Comparison result (max/min/mean/median)

<table>
<thead>
<tr>
<th>Size</th>
<th>$\Sigma_{nt}^{50,10}$</th>
<th>$\Sigma_{nt}^{100,10}$</th>
<th>$\Sigma^{50,10}$</th>
<th>$\Sigma^{100,10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>$\Sigma_{nt}^{50,10}$</td>
<td>$\Sigma_{nt}^{100,10}$</td>
<td>$\Sigma^{50,10}$</td>
<td>$\Sigma^{100,10}$</td>
</tr>
<tr>
<td>Size</td>
<td>$\Sigma_{nt}^{50,10}$</td>
<td>$\Sigma_{nt}^{100,10}$</td>
<td>$\Sigma^{50,10}$</td>
<td>$\Sigma^{100,10}$</td>
</tr>
<tr>
<td>Size</td>
<td>$\Sigma_{nt}^{50,10}$</td>
<td>$\Sigma_{nt}^{100,10}$</td>
<td>$\Sigma^{50,10}$</td>
<td>$\Sigma^{100,10}$</td>
</tr>
<tr>
<td>Size</td>
<td>$\Sigma_{nt}^{50,10}$</td>
<td>$\Sigma_{nt}^{100,10}$</td>
<td>$\Sigma^{50,10}$</td>
<td>$\Sigma^{100,10}$</td>
</tr>
</tbody>
</table>

**6 FUTURE WORK**

In this paper, we proposed an abstract notion of invariant to characterize sub-terminologies sharing the same logical information with respect to some signature. Based on the construction of a finite invariant, we translated the computation of minimal modules for $EL$-terminologies to a SAT problem. We developed **GIMM**, a prototype and the real-world ontologies experiments showed that our approach greatly improved the state-of-the-art method **Zooms**.

In the future, first, we will further optimize our algorithm for right-invariant computation and extend it to support signatures with loops. Second, we expect that our method (invariant) can be extended to drop the constraint on terminologies $O_1, O_2$ requiring $O_1 \subseteq O_2$. Third, we will investigate how to generalize our method to $EL$ ontologies or even more expressive languages.

### REFERENCES

1. M Fareed Arif, Carlos Mencia, Alexey Ignatiev, Norbert Manthey, Rafael Peñaloza, and Joao Marques-Silva. 2016. BEACON: An Efficient SAT-Based Tool for


