Computing EL minimal modules: a combined approach

Hui Yang, Yue Ma, Nicole Bidoit

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 subsumee 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/iwW49.

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 $\mathsf{N}_C=\{A, B, \cdots\}$ and atomic roles $\mathsf{N}_R=\{r, s, \cdots\}$, the set of $\mathcal{EL}$ concepts C and axioms α are built by the grammar rules (i) $C ::= \top \mid A \mid C \sqcap C \mid \exists r.C$ or (ii) $\alpha ::= C \sqsubseteq C \mid C \equiv C$. We denote by $sig(C)$ the atomic concepts and roles that compose C. For example, $sig(\exists r.(B_1 \sqcap B_2))=\{r, B_1, B_2\}$.

An $\mathcal{EL}$-ontology $\mathcal{O}$ is a finite set of $\mathcal{EL}$-axioms. An interpretation $\mathcal{I}=(\triangle^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{O}$ consists of a non-empty set $\triangle^{\mathcal{I}}$ and a mapping

from atomic concepts $A \in \mathsf{N}_C$ to a subset $A^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}}$ and from roles $r \in \mathsf{N}_R$ to a subset $r^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$. For a concept C built from the grammar rules, we define $C^{\mathcal{I}}$ inductively by: $(\top)^{\mathcal{I}} = \triangle^{\mathcal{I}}$, $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(\exists r.C)^{\mathcal{I}} = \{a \in \triangle^{\mathcal{I}} \mid \exists b \in C^{\mathcal{I}}, (a,b) \in r^{\mathcal{I}}\}$. An interpretation is a *model* of $\mathcal{O}$ if it is compatible with all axioms in $\mathcal{O}$, i.e., for all $C \sqsubseteq D, C \equiv D$, we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}, C^{\mathcal{I}} = D^{\mathcal{I}}$ respectively.

An $\mathcal{EL}$-ontology $\mathcal{O}$ is normalized if all its axioms are of the form $A \bowtie B_1 \sqcap B_2 \sqcap \cdots \sqcap B_m, A \bowtie \exists r.B$, where $\bowtie \in \{\equiv, \sqsubseteq\}$, $A, B, B_i \in \mathsf{N}_C, r \in \mathsf{N}_R$. Every $\mathcal{EL}$-ontology can be normalised in polynomial time by introducing new atomic concepts. Moreover, we say the $\mathcal{O}$ is a ***terminology*** if any atomic concept A appears at most once on the left-hand side of axiom in $\mathcal{O}$.

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

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

Given two ontologies $\mathcal{O}_1, \mathcal{O}_2$ and a *signature* $\Sigma \subseteq \mathsf{N}_C \cup \mathsf{N}_R$, the *logical difference* [1] between $\mathcal{O}_1, \mathcal{O}_2$ over Σ is the set:

$$\mathcal{D}_\Sigma(\mathcal{O}_1, \mathcal{O}_2) = \{\alpha | sig(\alpha) \subseteq \Sigma, \mathcal{O}_i \models \alpha, \mathcal{O}_j \not\models \alpha, \{i,j\} = \{1,2\}\}.$$

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

When $\mathcal{O}_1, \mathcal{O}_2$ are terminologies, we have:

Theorem 1. *[13] Given two terminologies $\mathcal{O}_1, \mathcal{O}_2$ and a signature Σ. If $\mathcal{D}_\Sigma(\mathcal{O}_1, \mathcal{O}_2)$ is not empty, then there exists an axiom in $\mathcal{D}_\Sigma(\mathcal{O}_1, \mathcal{O}_2)$ of the form $A \sqsubseteq C$ or $C \sqsubseteq A$, where A is an atomic concept.*

Example 1. *For terminology*

$$\mathcal{O}_0 = \{\beta_1 : B_2 \sqsubseteq A_1 \sqcap A_2,\ \beta_2 : B_1 \sqsubseteq A_1,\ \beta_3 : A_3 \equiv \exists s.A_1,\ \beta_4 : A_4 \equiv \exists s.A_2,$$
$$\beta_5 : B_3 \equiv A_3 \sqcap A_4,\ \beta_6 : A_4 \sqsubseteq \exists s.A_5,\ \beta_7 : A_5 \sqsubseteq B_4 \sqcap B_5\},$$

we have $\mathcal{O}_0 \models B_2 \sqsubseteq A_1$, and $J_0 = \{\beta_1\}$ is the only justification for $B_2 \sqsubseteq A_1$.

If we consider a signature $\Sigma_0 = \{s, B_1, B_2, B_3, B_4, B_5\}$, there is only one minimal module for Σ_0: $\mathcal{M}_0 = \mathcal{O}_0 \setminus \{\beta_2\}$.

2.2 Hyper-graph

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

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

(1) $f(e_n) = \{v\}$;
(2) for $i = 1, \cdots, n$, $T(e_i) \subseteq S \cup \{f(e_1; \cdots, f(e_{i-1}))\}$
(3) for $i = 1, \cdots, n$, $f(e_i) \in \bigcup_{i<j\leq n} T(e_j)$.

If $v \in T(e_1)$, 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 $\langle \{v_1\}, v \rangle$, $\langle \{v_1, \cdots, v_k\}, v \rangle$ as $v_1 \rightarrow v$, $\{v_1, \cdots, v_k\} \rightarrow v$.

[1] 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 Inv taking as input a terminology and a signature is an **invariant** if for any two terminologies $\mathcal{O}_1 \subseteq \mathcal{O}_2$,

$$\mathcal{D}_\Sigma(\mathcal{O}_1, \mathcal{O}_2) = \emptyset \text{ iff } \mathtt{Inv}(\mathcal{O}_1, \Sigma) = \mathtt{Inv}(\mathcal{O}_2, \Sigma).$$

Note that, the elements of $\mathtt{Inv}(\mathcal{O}, \Sigma)$ can be of any kinds. For example, a trivial invariant can be defined by the set of axioms $\{\alpha \mid \mathcal{O} \models \alpha, sig(\alpha) \subseteq \Sigma\}$. 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_i are finite sets.

3.1 Main idea

Similarly to [6], our construction of finite invariant is based on Theorem 1. Given two terminologies $\mathcal{O}_1, \mathcal{O}_2$, a signature Σ and a concept A, we consider:

$$\mathcal{D}^r_{\Sigma,A}(\mathcal{O}_1, \mathcal{O}_2) = \{C \sqsubseteq A \mid C \sqsubseteq A \in \mathcal{D}_\Sigma(\mathcal{O}_1, \mathcal{O}_2)\},$$
$$\mathcal{D}^l_{\Sigma,A}(\mathcal{O}_1, \mathcal{O}_2) = \{A \sqsubseteq C \mid A \sqsubseteq C \in \mathcal{D}_\Sigma(\mathcal{O}_1, \mathcal{O}_2)\}.$$

Definition 5. A map from the tuple $(\mathcal{O}, \Sigma, A)$ to a set $\mathtt{Inv}^r(\mathcal{O}, \Sigma, A)$ is a **right invariant** if for any two terminologies $\mathcal{O}_1 \subseteq \mathcal{O}_2$, we have:

$$\mathcal{D}^r_{\Sigma,A}(\mathcal{O}_1, \mathcal{O}_2) = \emptyset \text{ iff } \mathtt{Inv}^r(\mathcal{O}_1, \Sigma, A) = \mathtt{Inv}^r(\mathcal{O}_2, \Sigma, A).$$

The **left invariant** $\mathtt{Inv}^l(\mathcal{O}, \Sigma, A)$ is defined in a dual manner based on $\mathcal{D}^l_{\Sigma,A}(\mathcal{O}_1, \mathcal{O}_2)$.

Then as a corollary of Theorem 1, we have:

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

$$\mathtt{Inv}(\mathcal{O}, \Sigma) = \{\Big(A, \mathtt{Inv}^r(\mathcal{O}, \Sigma, A), \mathtt{Inv}^l(\mathcal{O}, \Sigma, A)\Big) \mid A \in \Sigma\}.$$

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

3.2.1 (1) Hyper-graph and hyper-paths. The hyper-graph $\mathcal{H}^{\mathcal{O}} = (\mathcal{N}, \mathcal{E})$ associated to $\mathcal{O}$ consists of the node set $\mathcal{N} := \{N_A \mid A \in \mathsf{N}_C\}$ and the edge set

$$\mathcal{E} := \{\{N_{A_1}, \cdots, N_{A_n}\} \rightarrow N_A \mid A \equiv A_1 \sqcap A_2 \sqcap \cdots \sqcap A_n \in \mathcal{O}\}$$
$$\cup \{N_A \xrightarrow{r} N_B \mid B \equiv \exists r.A \in \mathcal{O}\} \cup \{N_A \rightarrow N_B \mid \mathcal{O} \models A \sqsubseteq B\},$$

where $N_A \xrightarrow{r} N_B$ is an edge with the index $r \in \mathsf{N}_R$.

Among the hyper-paths in $\mathcal{H}^{\mathcal{O}}$, we specifically consider the set $P^{\mathcal{O}}_{\Sigma,A}$ of all the hyper-paths h from $S_\Sigma = \{N_B \mid B \in \Sigma\}$ to N_A in $\mathcal{H}^{\mathcal{O}}$ such that:

(1) for an 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: $[\{N_{A_1}, \cdots, N_{A_n}\} \to N_B, N_B \to N_{A_i}]$ for some $i \in [1, n]$;
(3) h does not contain repeated edges.

Item 3 above implies that the length of an hyper-path $h \in P^{\mathcal{O}}_{\Sigma,A}$ is at most $\#|\mathcal{E}|$ and therefore $P^{\mathcal{O}}_{\Sigma,A}$ is finite.

Example 2 (Example 1 cont'd)**.** *The hyper-graph associated with the ontology $\mathcal{O}_0$ is shown in Figure 1. Notice that $e_6 = \{N_{A_3}, N_{A_4}\} \to N_{B_3}$ is the only complex edge in this hyper-graph. There are two hyper-paths in $P^{\mathcal{O}_0}_{\Sigma_0,B_3}$: $h_1 = [e_1, e_4, e_3, e_5, e_6]$, and $h_2 = [e_2, e_4, e_3, e_5, e_6]$.*

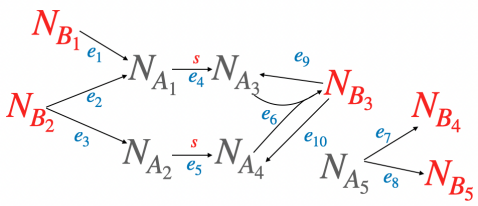


Figure 1: $\mathcal{H}^{\mathcal{O}_0}$, concepts and roles in Σ_0 are in red.

Note that $l_1 = [e_6, e_9]$, $l_2 = [e_6, e_{10}]$ are trivial loops. Thus any hyper-path containing l_1 or l_2 is excluded from $P^{\mathcal{O}_0}_{\Sigma_0,B_3}$.

3.2.2 (2) Our finite right invariant. Each hyper-path $h \in P^{\mathcal{O}}_{\Sigma,A}$ can be interpreted as an ontology:

$$\mathcal{O}_h := \{A_1 \sqcap A_2 \cdots \sqcap A_n \sqsubseteq A \mid \{N_{A_1}, \cdots, N_{A_n}\} \to N_A \in h\} \cup$$
$$\{A \sqsubseteq B \mid N_A \to N_B \in h\} \cup \{\exists r.A \sqsubseteq B \mid N_A \xrightarrow{r} N_B \in h\},$$

Indeed, the existence of a hyper-path h determines whether $\mathcal{O} \models C \sqsubseteq A$ in the following sense:

Theorem 2. *$\mathcal{O} \models C \sqsubseteq A$, $sig(C) \subseteq \Sigma$ iff there exists a hyper-path $h \in P^{\mathcal{O}}_{\Sigma,A}$ such that $\mathcal{O}_h \models C \sqsubseteq A$.*

Now, one could expect that $P^{\mathcal{O}}_{\Sigma,A}$ 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 illustrates 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 $\mathcal{O}_h$: $I(h) := \{C \sqsubseteq A \mid \mathcal{O}_h \models C \sqsubseteq A, sig(C) \in \Sigma\}$.

Example 3 (Example 1 and 2 cont'd)**.** *For terminology $\mathcal{O}_0$, we have $P^{\mathcal{O}_0}_{\Sigma_0,B_3} = \{h_1, h_2\}$. Note that β_2 is one of the axioms corresponding to $\{h_1, h_2\}$ but β_2 does not belong to the unique minimal module $\mathcal{M}_0$. The reason is that $I(h_1) \subset I(h_2)$, 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 $\mathcal{M}_0$.*

Example 4. *Assume $\mathcal{O}_1 = \{\beta_1 : B_2 \sqsubseteq B_1, \beta_2 : A \sqsubseteq B_1, \beta_3 : B_2 \sqsubseteq A\}$ and $\Sigma_1 = \{B_1, B_2\}$. Then*

$$P^{\mathcal{O}_1}_{\Sigma_1,B_1} = \{h_1 : [N_{B_2} \to N_{B_1}], h_2 : [N_{B_2} \to N_A, N_A \to N_{B_1}]\}.$$

We have $I(h_1) = I(h_2)$, meaning that h_1, h_2 provide the same logical information and thus a repetition. There are two minimal modules for Σ_1: $\mathcal{M}_1 = \{\beta_1\}$ and $\mathcal{M}_2 = \{\beta_2, \beta_3\}$ 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 $h_1, h_2 \in P^{\mathcal{O}}_{\Sigma,A}$, we say:

- h_1 is *redundant* if $I(h_1) \subset I(h')$ for some $h' \in P^{\mathcal{O}}_{\Sigma,A}$;
- h_1, h_2 are *equivalent* if $I(h_1) = I(h_2)$.

Now, to eliminate redundancy and repetition, we extract from $P^{\mathcal{O}}_{\Sigma,A}$ the set $\{\overline{h} \mid h \in P^{\mathcal{O}}_{\Sigma,A}$ is not redundant$\}$, where $\overline{h}$ denotes the class of all hyper-paths equivalent to h. Finally, our finite right invariant is defined by:

Theorem 3. *The map $\mathtt{Inv}^{\mathsf{r}}$ defined below is a right invariant:*

$$\mathtt{Inv}^{\mathsf{r}}(\mathcal{O}, \Sigma, A) = \{\overline{h} \mid h \in P^{\mathcal{O}}_{\Sigma,A} \text{ is not redundant}\}.$$

Clearly, $\mathtt{Inv}^{\mathsf{r}}(\mathcal{O}, \Sigma.A)$ is finite since $P^{\mathcal{O}}_{\Sigma,A}$ is a finite set.

Example 5 (Example 3 cont'd)**.** *Since $I(h_1) \subset I(h_2)$, the hyper-path h_1 is redundant and $\mathtt{Inv}^{\mathsf{r}}(\mathcal{O}_0, \Sigma_0, B_3) = \{\overline{h_2}\}$.*

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 $I(h_1) \subset I(h_2)$), thus indeed, an algorithm for checking $I(h_1) \subseteq I(h_2)$. In the general case, we can rely on an existing polynomial algorithm [7]. However, in the case of loop-free hyper-paths, we designed a specific straightforward algorithm. For details see supplementary materials.

3.3 Building a finite left invariant

The left invariant is constructed along the same lines as the right case. The main difference is that our left invariant is based on a graph $\mathcal{G}^{\mathcal{O}}$ (instead of a hyper-graph) and on *clusters* (instead of hyper-paths).

The graph $\mathcal{G}^{\mathcal{O}} = (\mathcal{N}, \mathcal{E})$ associated with $\mathcal{O}$ consists of a node set $\mathcal{N} := \{N_A \mid A \in \mathsf{N_C}\}$ and an edge set

$$\mathcal{E} := \{N_A \to N_B \mid \mathcal{O} \models A \sqsubseteq B\}$$
$$\cup \{N_A \xrightarrow{r} N_B \mid A \equiv \exists r.B \in \mathcal{O} \text{ or } A \sqsubseteq \exists r.B \in \mathcal{O}\}.$$

One may consider to construct a finite left invariant using paths in $\mathcal{G}^{\mathcal{O}}$ as in Section 3.2. However, the paths in $\mathcal{G}^{\mathcal{O}}$ are not sufficient for instance to capture the entailment $\mathcal{O} \models A \sqsubseteq \exists r.(B \sqcap C)$. Indeed, a path like $[N_A \to N_{A_1}, N_{A_1} \xrightarrow{r} N_{A_2}, N_{A_2} \xrightarrow{s} N_B]$ in $\mathcal{G}^{\mathcal{O}}$ corresponds to $\mathcal{O} \models A \sqsubseteq \exists r.\exists s.B$, 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 Σ, is a union of paths, which do not contain a loop twice[2], from N_A to some N_B, $B \in \Sigma$ in $\mathcal{G}^{\mathcal{O}}$. We say t is *loop-free* if it does not contain loops.

Let us define $T^{\mathcal{O}}_{\Sigma,A}$ as the collection of all clusters from A to Σ. Then $T^{\mathcal{O}}_{\Sigma,A}$ 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 $\mathcal{O}_0$, $\mathcal{G}^{\mathcal{O}_0}$ 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_{B_3} to some node in Σ and three loop-free clusters $t_1 = p_1$, $t_2 = p_2$, $t_3 = p_1 \cup p_2$.*

[2]For example, $[N_1 \to N_2, N_2 \to N_3, N_3 \to N_2, N_2 \to N_3, N_3 \to N_2]$ contains the loop $[N_2 \to N_3, N_3 \to N_2]$ twice.
[3]The upper bound is $(m+2)!$ if $\mathcal{G}^{\mathcal{O}}$ has m edges since a path with length $m+1$ contains a loop, and thus are at most $m!$ such loops.

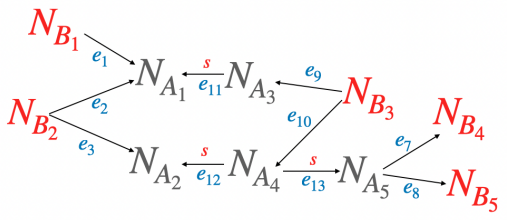


Figure 2: $\mathcal{G}^{O_0}$, concepts and roles in Σ_0 are red.

Each cluster t is a sub-graph of $\mathcal{G}^{\mathcal{O}}$ and can be associated with an ontology:

$$\mathcal{O}_t := \{A \sqsubseteq B \mid N_A \rightarrow N_B \in t\} \cup \{A \sqsubseteq \exists r.B \mid N_A \xrightarrow{r} N_B \in t\}.$$

The following result is similar to Theorem 2:

Theorem 4. *$\mathcal{O} \models A \sqsubseteq C$, $sig(C) \subseteq \Sigma$ iff there exists a cluster $t \in T^{\mathcal{O}}_{\Sigma,A}$ such that $\mathcal{O}_t \models A \sqsubseteq C$.*

To define a finite left invariant, we need to filter out repetitive and redundant clusters in $T^{\mathcal{O}}_{\Sigma,A}$. Let us denote $I(t) = \{A \sqsubseteq C \mid \mathcal{O}_t \models A \sqsubseteq C, sig(C) \subseteq \Sigma\}$ and say:

- $t_1 \in T^{\mathcal{O}}_{\Sigma,A}$ is redundant if there exists $t' \in T^{\mathcal{O}}_{\Sigma,A}$ such that $I(t_1) \subset I(t')$,
- $t_1, t_2 \in T^{\mathcal{O}}_{\Sigma,A}$ are equivalent if $I(t_1) = I(t_2)$.

Then, our finite left invariant is defined as follows:

Theorem 5. *The map $\mathtt{Inv}^{\mathrm{l}}$ defined below is a left invariant.*

$$\mathtt{Inv}^{\mathrm{l}}(\mathcal{O}, \Sigma, A) = \{\bar{t} \mid t \in T^{\mathcal{O}}_{\Sigma,A} \text{ is not redundant}\}.$$

To compute $\mathtt{Inv}^{\mathrm{l}}(\mathcal{O}, \Sigma, A)$, as in Section 3.2, we need to determine whether $I(t_1) \subseteq I(t_2)$. The general case can be solved by a polynomial algorithm proposed in [7]. As for hyper-paths, we designed a specific straightforward algorithm for loop-free clusters (for details see supplementary materials).

Example 7. *Continuing Example 6, since $I(t_1) \subset I(t_3)$ and $I(t_2) \subset I(t_3)$, the clusters t_1 and t_2 are redundant and $\mathtt{Inv}^{\mathrm{l}}(\mathcal{O}, \Sigma_0, B_3) = \{\bar{t}_3\}$.*

In conclusion of this section, thanks to Corollary 1, we are able to exhibit a finite invariant $\mathtt{Inv}$ for $\mathcal{O}, \Sigma$ using the finite right and left invariant above.

4 COMPUTING MINIMAL MODULES

Given a terminology $\mathcal{O}$ and a signature Σ, in this section, we compute the minimal modules for $\mathcal{O}$ and Σ using a SAT-based approach as in PULi [11]. Thus minimal modules are computed in three steps:

Step 1. The right invariants $\mathtt{Inv}^{\mathrm{r}}(\mathcal{O}, \Sigma, A)$ and left invariants $\mathtt{Inv}^{\mathrm{l}}(\mathcal{O}, \Sigma, A)$ are computed for each $A \in \Sigma$ which, by Corollary 1, provides us with the invariant $\mathtt{Inv}(\mathcal{O}, \Sigma)$.

Step 2. We build a set C_Σ of Horn clauses to encode the derivations of elements in $\mathtt{Inv}(\mathcal{O}, \Sigma)$ from $\mathcal{O}$. These clauses are given below:

(1) We add the clause $\wedge_{A \in \Sigma}(l^r_A \wedge l^l_A) \rightarrow l_\Sigma$, where l_Σ captures a minimal module for Σ and $l^r_A \wedge l^l_A$ a tuple

$$(A, \mathtt{Inv}^{\mathrm{r}}(\mathcal{O}, \Sigma, A), \mathtt{Inv}^{\mathrm{l}}(\mathcal{O}, \Sigma, A))$$

of the invariant;

(2) We add two clauses

$$\wedge_{\bar{h} \in \mathtt{Inv}^{\mathrm{r}}(\mathcal{O},\Sigma,A)}(l_{\bar{h}}) \rightarrow l^r_A, \wedge_{\bar{t} \in \mathtt{Inv}^{\mathrm{l}}(\mathcal{O},\Sigma,A)}(l_{\bar{t}}) \rightarrow l^l_A$$

for each $A \in \Sigma$;

(3) We add the clause $\wedge_{e \in h}(l_e) \rightarrow l_{\bar{h}}$ (resp. $\wedge_{e \in t}(l_e) \rightarrow l_{\bar{t}}$), for each non-redundant $h \in P^{\mathcal{O}}_{\Sigma,A}$ (resp. $t \in T^{\mathcal{O}}_{\Sigma,A}$) and $A \in \Sigma$;

(4) For each edge $e \in h$ (resp. $e \in t$), where $h \in P^{\mathcal{O}}_{\Sigma,A}$ (resp. $t \in T^{\mathcal{O}}_{\Sigma,A}$) is non-redundant and $A \in \Sigma$:

(a) if $e = N_A \rightarrow N_B$, then we add clause $\wedge_{\beta \in J}(l_\beta) \rightarrow l_e$ for each justification J of $A \sqsubseteq B$;

(b) if $e = N_A \xrightarrow{r} N_B$ or $\{N_{A_1}, \cdots, N_{A_n}\} \rightarrow N_B$, then we add the clause $l_\beta \rightarrow l_e$, where $\beta \in \mathcal{O}$ is the axiom corresponding to e.

The **answer literals** are defined as the literals l_β such that $\beta \in \mathcal{O}$.

Step 3. We apply *resolution* over $C_\Sigma \cup \{l_\Sigma \rightarrow \emptyset\}$. Then let us consider $\mathtt{M}_\Sigma$, the collection of resulting minimal[4] clauses composed of answer literals only. From $\mathtt{M}_\Sigma$, all minimal modules for Σ are extracted easily as stated in Theorem 6.

Theorem 6. *A subset $\mathcal{M} = \{\beta_1, \cdots, \beta_k\} \subseteq \mathcal{O}$ is a minimal module for Σ with respect to $\mathcal{O}$ iff $\wedge^k_{i=1}(l_{\beta_i}) \rightarrow \emptyset \in \mathtt{M}_\Sigma$.*

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 $\mathtt{Inv}^{\mathrm{r}}(\mathcal{O}_0, \Sigma_0, B_3) = \{\bar{h}_2\}$, $\mathtt{Inv}^{\mathrm{l}}(\mathcal{O}_0, \Sigma_0, B_3) = \{\bar{t}_3\}$. Thus,*

$$\mathtt{Inv}(\mathcal{O}_0, \Sigma_0) = \{(B_3, \{\bar{h}_2\}, \{\bar{t}_3\})\} \cup \{(B_k, \emptyset, \emptyset) \mid k = 1, 2, 4, 5\}$$

The set of clauses C_{Σ_0} is shown in Table 1. We obtain

$$\mathtt{M}_{\Sigma_0} = \{l_{\beta_1} \wedge l_{\beta_3} \wedge l_{\beta_4} \wedge l_{\beta_5} \wedge l_{\beta_6} \wedge l_{\beta_7} \rightarrow \emptyset\}$$

by resolution over $C_{\Sigma_0} \cup \{l_{\Sigma_0} \rightarrow \emptyset\}$. Therefore, the only minimal module for Σ_0 is $\mathcal{O}_0 \setminus \{\beta_2\}$.

Approximation. Now, let $\mathcal{O}^{ap} = \{\beta \in \mathcal{O} \mid l_\beta \text{ appears in } C_\Sigma\}$, then $\mathcal{O}^{ap}$ contains all the minimal modules of Σ and thus $\mathcal{D}_\Sigma(\mathcal{O}^{ap}, \mathcal{O}) = \emptyset$. Next, we regard $\mathcal{O}^{ap}$ as an approximation of minimal modules. Indeed, $\mathcal{O}^{ap}$ 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 $P^{\mathcal{O}}_{\Sigma,A}$ (resp. $T^{\mathcal{O}}_{\Sigma,A}$) with respect to the size of $\mathcal{H}^{\mathcal{O}}$ (resp. $\mathcal{G}^{\mathcal{O}}$). In order to reduce the computation cost, we can restrict the search space to hyper-paths (resp. clusters) having particular forms.

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

Table 1: Clause set C_{Σ_0}

1.	$\wedge_{i=1}^{5}(l^r_{B_i} \wedge l^l_{B_i}) \rightarrow l_\Sigma$;
2.	$l^-_{h_2} \rightarrow l^r_{B_3}$, $l^-_{t_3} \rightarrow l^l_{B_3}$, $\emptyset \rightarrow l^r_{B_k}$, $\emptyset \rightarrow l^r_{B_k}, k \in \{1,2,4,5\}$;
3.	$l_{e_2} \wedge l_{e_4} \wedge l_{e_3} \wedge l_{e_5} \wedge l_{e_6} \rightarrow l^-_{h_2}$, $l_{e_{10}} \wedge l_{e_{13}} \wedge l_{e_7} \wedge l_{e_8} \rightarrow l^-_{t_3}$;
4. (a)	$e_2 : l_{\beta_1} \rightarrow l_{e_2}, e_3 : l_{\beta_1} \rightarrow l_{e_3}$, $e_7 : l_{\beta_7} \rightarrow l_{e_7}, e_8 : l_{\beta_7} \rightarrow l_{e_8}$.
(b)	$e_4 : l_{\beta_3} \rightarrow l_{e_4}, e_5 : l_{\beta_4} \rightarrow l_{e_5}$, $e_{11} : l_{\beta_3} \rightarrow l_{e_{11}}, e_{12} : l_{\beta_4} \rightarrow l_{e_{12}}$, $e_{13} : l_{\beta_6} \rightarrow l_{e_{13}}, e_6 : l_{\beta_5} \rightarrow l_{e_6}$.

(1) Hyper-paths. We say a hyper-path h over $\mathcal{H}^{\mathcal{O}}$ is **compact** when it does not contain sub-paths of either form:

(1) $[\{N_{B_1}, \cdots, N_{B_n}\} \rightarrow N_B, N_B \rightarrow N_A]$, where $B \notin \Sigma$,

(2) $[N_{B_1} \xrightarrow{r} N_B, N_B \rightarrow N_A]$, where $B \notin \Sigma$.

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

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

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

(2) Clusters. We say that a cluster $t = p_1 \cup p_2 \cup \cdots \cup p_n$ in $T^{\mathcal{O}}_{\Sigma,A}$ is **compact** if either $n=1$ or the paths $p_1, \cdots, p_n$ have a common prefix path $[e_1, \cdots, e_k]$ 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 or $\exists r.C_1$, $\mathcal{O} \models A \sqsubseteq C$ iff there exists a compact cluster $t \in T^{\mathcal{O}}_{\Sigma,A}$ such that $\mathcal{O}_t \models A \sqsubseteq C$.*

In proposition 3, we only consider concepts of the form B or $\exists r.C_1$, because any concept C can be written as $B_1 \sqcap \cdots \sqcap B_n \sqcap \exists r_1.C_1 \sqcap \cdots \sqcap \exists r_m.C_m$, where $B_1, \cdots, B_n$ are atomic concepts. Thus, we know that $\mathcal{O} \models A \sqsubseteq C$ iff $\mathcal{O} \models A \sqsubseteq B_i$, $\mathcal{O} \models A \sqsubseteq \exists r_j.C_j$ for any $1 \leq i \leq n$, $1 \leq j \leq m$.

Therefore, by Proposition 3, we can require all clusters in $T^{\mathcal{O}}_{\Sigma,A}$ 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 $\mathcal{EL}$-fragment of two prominent biomedical ontologies: Snomed CT (version Jan 2016)[5], a terminology with 317891 axioms, and NCI (version 16.03d)[6], a terminology having 165341 axioms. All the experiments are run on a machine with an Intel Xeon Core 4 Duo CPU 2.50 GHz with 64 GiB of RAM.

[5]https://www.snomed.org/

[6]http://evs.nci.nih.gov/ftp1/NCI_Thesaurus

Table 3: Successful rate

Successful rate(%)	$\Sigma^{snt}_{50,10}$	$\Sigma^{snt}_{100,10}$	$\Sigma^{nci}_{50,10}$	$\Sigma^{nci}_{100,10}$
Zooms	57.1	32.5	79.0	57.3
GIMM	**84.2** (+27.1)	**78.5** (+46)	**91.8** (+12.8)	**74.3** (+17)

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

Time(s)	*GIMM*	*Zooms*
$\Sigma^{snt}_{50,10}$	7.57 / 5.13 / 6.13 / 6.12	558.76 / 34.85 / 186.04 / 143.53
$\Sigma^{snt}_{100,10}$	9.82 / 5.19 / 6.24 / 6.18	563.24 / 71.38 / 302.24 / 294.19
$\Sigma^{nci}_{50,10}$	29.54 / 2.03 / 2.56 / 2.43	560.89 / 7.81 / 91.08 / 60.42
$\Sigma^{nci}_{100,10}$	15.50 / 2.04 / 2.63 / 2.46	576.35 / 15.49 / 145.32 / 105.92

For each terminology, we run the experiments over 2 sets $\Sigma_{n,m}$ of 1000 signatures randomly generated, each one containing n concepts and m roles. We tested $\Sigma^{snt}_{50,10}, \Sigma^{snt}_{100,10}$ for Snomed CT and $\Sigma^{nci}_{50,10}, \Sigma^{nci}_{100,10}$ for NCI.

Next, we say a signature Σ is *loop-free* if all hyper-paths (resp. clusters) in the right (resp. left) invariant of (Σ, A), $A \in \Sigma$ are loop-free. A signature is *trivial* if its minimal module is empty. *GIMM* computes minimal modules for loop-free signatures. Table 2 (Column 2) shows the distribution of signatures among "with loop", "trivial", and "loop-free and non trivial". Then it provides the maximal, minimal, average number (Column 3) and size (Column 4) of the minimal modules with respect to the signatures.

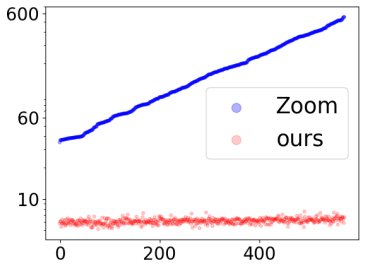
Table 2: Running GIMM on the signature sets: statistics

	Signature (loop/trivial/ non-trivial)	Minimal modules	
		Number (max/min/mean)	Size (max/min/mean)
$\Sigma^{snt}_{50,10}$	0 / 53 / 947	935325 / 1 / 5046.82	112 / 0 / 72.59
$\Sigma^{snt}_{100,10}$	0 / 3 / 997	322597 / 1/ 1330.87	134 / 0 / 86.04
$\Sigma^{nci}_{50,10}$	25 / 21 / 954	10947030 / 1 / 63929.41	110 / 0 / 47.42
$\Sigma^{nci}_{100,10}$	47 / 4 / 949	4334784 / 1 / 129078	144 / 0 / 54.10

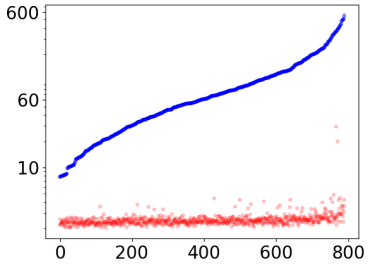
Time cost: GIMM vs. Zooms. First, we compare *GIMM* with the state-of-the-art algorithm [6], called *Zooms*. For each signature, we set the run-time limit to 600s. For simplicity, we regard signatures with loop as timed out samples for *GIMM*. *Zooms* is actually unable to solve any of these signatures.

As shown in Table 3, the successful rate of *GIMM* is between +12.8% and +46% 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 $\Sigma^{nci}_{50,10}$. 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.

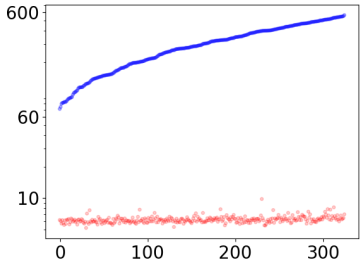
One reason *GIMM* runs so fast is that the size of our invariants were usually small. On average, we have 0.95 non-redundant hyper-paths and 11.51 non-redundant clusters for each signature. For more than half of the cases, the set of non-redundant hyper-paths is empty. But in some rare cases, there exists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



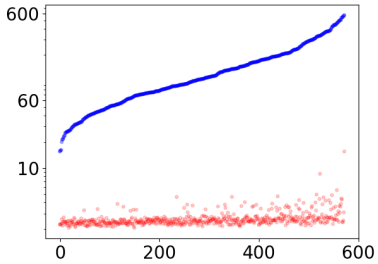
(a) $\Sigma^{snt}_{50,10}$



(b) $\Sigma^{nci}_{50,10}$



(c) $\Sigma^{snt}_{100,10}$



(d) $\Sigma^{nci}_{100,10}$

Figure 3: x: Signatures, y: time cost (s). In each sub-figure, we arrange the order of signatures according to their time cost in *Zooms*

For the signatures solved by both algorithms, *GIMM* spent most of the time (96.8% on average) on computing our invariant. Besides, *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 $\mathcal{H}^{\mathcal{O}}$ is always much bigger than the graph $\mathcal{G}^{\mathcal{O}}$. Thus extracting hyper-paths over $\mathcal{H}^{\mathcal{O}}$ is more difficult than extracting clusters over $\mathcal{G}^{\mathcal{O}}$, even if there are much more non-redundant clusters than hyper-paths on average.

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

Size	$\#\lvert O^{ap}\rvert$	*#locality-based module*
$\Sigma^{snt}_{50,10}$	1581 / 0 / 237.11 / 67.5	10182 / 3426 / 6539.11 / 6506.5
$\Sigma^{snt}_{100,10}$	2537 / 0 / 480.75 / 154	18311 / 8515 / 13080.30 / 13033
$\Sigma^{nci}_{50,10}$	285 / 0 / 58.94 / 51	7746 / 256 / 5542.04 / 6874
$\Sigma^{nci}_{100,10}$	332 / 0 / 101.59 / 90	8571 / 1261 / 7337.73 / 7465

Time(s)	*GIMM*	*locality-based module (OWL API)*
$\Sigma^{snt}_{50,10}$	34.71 / 5.13 / 8.22 / 6.37	13.23 / 4.08 / 7.48/ 7.88
$\Sigma^{snt}_{100,10}$	259.39 / 5.19 / 10.24 / 6.79	12.01 / 4.44 / 7.83 / 8.19
$\Sigma^{nci}_{50,10}$	4.42 / 2.03 / 2.49 / 2.44	3.99 / 1.72 / 2.85 / 2.93
$\Sigma^{nci}_{100,10}$	5.17 / 2.04 / 2.63 / 2.52	3.92 / 2.10 / 2.96 / 2.96

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 $\Sigma^{nci}_{100,10}$ 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 $\Sigma^{snt}_{100,10}$ 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 $\Sigma^{snt}_{100,10}$).

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 $\mathcal{EL}$-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 $\mathcal{O}_1, \mathcal{O}_2$ requiring $\mathcal{O}_1 \subseteq \mathcal{O}_2$. Third, we will investigate how to generalize our method to $\mathcal{EL}$ ontologies or even more expressive languages.

REFERENCES

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

Debugging $\mathcal{EL}^+$ Ontologies. In *International Conference on Theory and Applications of Satisfiability Testing*. Springer, 521–530.

[2] M Fareed Arif, Carlos Mencía, and Joao Marques-Silva. 2015. Efficient axiom pinpointing with EL2MCS. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*. Springer, 225–233.

[3] M Fareed Arif, Carlos Mencía, and Joao Marques-Silva. 2015. Efficient MUS enumeration of Horn formulae with applications to axiom pinpointing. In *International Conference on Theory and Applications of Satisfiability Testing*. Springer, 324–342.

[4] Giorgio Ausiello and Luigi Laura. 2017. Directed hypergraphs: Introduction and fundamental algorithms—a survey. *Theoretical Computer Science* 658 (2017), 293–306.

[5] Franz Baader, Rafael Penaloza, and Boontawee Suntisrivaraporn. 2007. Pinpointing in the Description Logic EL+. In *Annual Conference on Artificial Intelligence*. Springer, 52–67.

[6] Jieying Chen, Michel Ludwig, Yue Ma, and Dirk Walther. 2017. Zooming in on Ontologies: Minimal Modules and Best Excerpts. In *16th International Semantic Web Conference, Proceedings, Part I*. Springer, 173–189. https://doi.org/10.1007/978-3-319-68288-4_11

[7] Andreas Ecke, Michel Ludwig, and Dirk Walther. 2013. The Concept Difference for EL-Terminologies using Hypergraphs.. In *DChanges*. Citeseer.

[8] William Gatens, Boris Konev, and Frank Wolter. 2014. Lower and Upper Approximations for Depleting Modules of Description Logic Ontologies.. In *European Conference on Artificial Intelligence*. 345–350.

[9] B Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. 2008. Modular reuse of ontologies: Theory and practice. *Journal of Artificial Intelligence Research* 31 (2008), 273–318.

[10] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Ulrike Sattler, Thomas Schneider, and Rafael Berlanga. 2008. Safe and economic re-use of ontologies: A logic-based methodology and tool support. In *European Semantic Web Conference*. Springer, 185–199.

[11] Yevgeny Kazakov and Peter Skočovskỳ. 2018. Enumerating justifications using resolution. In *International Joint Conference on Automated Reasoning*. Springer, 609–626.

[12] Boris Konev, Michel Ludwig, Dirk Walther, and Frank Wolter. 2012. The Logical Difference for the Lightweight Description Logic EL. *J. Artif. Intell. Res.* 44 (2012), 633–708.

[13] Boris Konev, Michel Ludwig, Dirk Walther, and Frank Wolter. 2012. The logical difference for the lightweight description logic EL. *Journal of Artificial Intelligence Research* 44 (2012), 633–708.

[14] Boris Konev, Carsten Lutz, Dirk Walther, and Frank Wolter. 2008. Semantic Modularity and Module Extraction in Description Logics.. In *European Conference on Artificial Intelligence*. 55–59.

[15] Roman Kontchakov, Frank Wolter, and Michael Zakharyaschev. 2010. Logic-based ontology comparison and module extraction, with an application to DL-Lite. *Artificial Intelligence* 174, 15 (2010), 1093–1141.

[16] Patrick Koopmann and Jieying Chen. 2020. Deductive Module Extraction for Expressive Description Logics. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, Christian Bessiere (Ed.). 1636–1643. https://doi.org/10.24963/ijcai.2020/227

[17] Patrick Koopmann and Renate A. Schmidt. 2013. Forgetting Concept and Role Symbols in ALCH-Ontologies. In *Logic for Programming, Artificial Intelligence, and Reasoning - 19th International Conference*. Springer, 552–567. https://doi.org/10.1007/978-3-642-45221-5_37

[18] Michel Ludwig and Dirk Walther. 2014. The Logical Difference for ELHr-Terminologies using Hypergraphs. In *Euro-pean Conference on Artificial Intelligence*. IOS Press, 555–560.

[19] Carsten Lutz and Frank Wolter. 2011. Foundations for uniform interpolation and forgetting in expressive description logics. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

[20] Norbert Manthey, Rafael Peñaloza, and Sebastian Rudolph. 2016. Efficient Axiom Pinpointing in EL using SAT Technology.. In *Description Logics*.

[21] Rafael Penaloza and Barış Sertkaya. 2017. Understanding the complexity of axiom pinpointing in lightweight description logics. *Artificial Intelligence* 250 (2017), 80–104.

[22] Ulrike Sattler, Thomas Schneider, and Michael Zakharyaschev. 2009. Which kind of module should I extract? *Description Logics* 477 (2009), 78.