ReAD: Delegate OWL Reasoners for Ontology Classification with Atomic Decomposition[1]

Haoruo ZHAO [a], Bijan PARSIA [a] and Uli SATTLER [a]
[a] *University of Manchester, United Kingdom*
{ *haoruo.zhao, bijan.parsia, uli.sattler* } *@manchester.ac.uk*

Abstract. Classification is a key ontology reasoning task. Several highly-optimised OWL reasoners are designed for different fragments of OWL 2. Combining these delegate reasoners to classify one ontology gives potential benefits, but these may be offset by overheads or redundant subsumption tests. In this paper, we show that with the help of the atomic decomposition, a known ontology partition approach, these redundant subsumption tests can be avoided. We design and implement our classification algorithms and empirically evaluate them.

Keywords. OWL, Description Logic, Classification, Delegate Reasoner

1. Introduction

Classification is a core reasoning task for *Web Ontology Language (OWL)* 2 [1] *ontologies*. Several classification algorithms have been implemented in highly-optimized reasoners [2,3,4,5] for different fragments [6,7,8] of OWL 2. For OWL 2, these use a *traversal* algorithm which determines which *subsumptions* to test next, ideally avoiding most of them [9,10]. Ontology modularisation [11,12,13,14], especially *locality-based modules* [15] (logical approximations of *conservative extension*-based modules) and *Atomic Decomposition (AD)* [16] (an ontology partition algorithm), have already been used for optimizing the classification [17,18,19,20] by easing *subsumption test* (STs) or avoiding them. But checking STs with modules or AD rather than the whole ontology causes redundant STs. In this paper, we design and implement our algorithm with no redundant STs and also evaluate our algorithm with a corpus of *BioPortal* ontologies.

2. Background Knowledge

We assume the reader to be familiar with *Description Logics* (DLs), a family of knowledge representation languages underlying OWL 2 that are basically decidable fragments of First Order Logic [21], and only fix the notation used. In this paper, we use $\mathcal{O}$ for an

ontology and $\mathcal{M} \subseteq \mathcal{O}$ for a $\bot$-syntactic locality-based module.[2] A *signature* Σ is a set of concept names and role names. For an axiom, a module, or an ontology X, we use $\widetilde{X}$ to denote its signature.

2.1. Classification

To *classify* an ontology $\mathcal{O}$, we first check whether $\mathcal{O}$ is consistent. If it is, we check for all $A, B \in \widetilde{\mathcal{O}} \cap \mathbf{N_C}$, $A \neq B$, whether $\mathcal{O} \models^? A \sqsubseteq B, \mathcal{O} \models^? A \sqsubseteq \bot$ and $\mathcal{O} \models^? \top \sqsubseteq B$. Naively, this results in a quadratic number of entailment tests. To avoid these, DL reasoners employ a form of *traversal algorithm* to reduce the number of STs [9,10]. In practice, these are highly effective, almost always reducing the quadratic number of STs to at most $n * \log n$ tests [19]. In this paper, we concentrate on ST avoidance and assume all ontologies to be consistent.

2.2. Modules and AD

Throughout this paper, we concentrate on $\bot$-locality based modules [22]. These are subsets of an ontology that have certain properties which are important for their use in AD and, in turn, for classification optimization. Let $\mathcal{M} = \mathcal{M}(\Sigma, \mathcal{O})$ be such a $\bot$-locality based module of $\mathcal{O}$ for the signature Σ. Then $\mathcal{M}$

1. preserves all entailments of $\mathcal{O}$ over Σ, i.e., it covers Σ,
2. preserves all entailments of $\mathcal{O}$ over $\widetilde{\mathcal{M}}$, i.e., it is self-contained,
3. is unique, i.e., there is no other $\bot$-locality based module of $\mathcal{O}$ for the signature Σ, and
4. is subsumer-preserving, i.e., if A is a concept name in $\mathcal{M}$ and $\mathcal{O} \models A \sqsubseteq B$, then $\mathcal{M} \models A \sqsubseteq B$.

The *atomic decomposition (AD)* $\mathfrak{A}(\mathcal{O})$ [16] partitions an ontology into logically inseparable sets of axioms, so-called *atoms* $\mathfrak{a}$, and relates these atoms via a *dependency relation* $\succeq$. In the rest of paper, we use $\mathfrak{A}(\mathcal{O})$ to represent the $\bot$-AD because in [23], we find these to be mostly fine-grained than other ADs.

Definition 1. *[23] Given two axioms $\alpha, \beta \in \mathcal{O}$, we say $\alpha \sim_{\mathcal{O}} \beta$ if for all modules $\mathcal{M}$ of $\mathcal{O}$, we have $\alpha \in \mathcal{M}$ iff $\beta \in \mathcal{M}$. The atoms of an ontology $\mathcal{O}$ are the equivalence classes $\sim_{\mathcal{O}}$. Given two atoms $\mathfrak{a}$, $\mathfrak{b}$, we say that* $\mathfrak{a}$ is dependent on $\mathfrak{b}$*, written $\mathfrak{a} \succeq \mathfrak{b}$ if, for any module $\mathcal{M}$, $\mathfrak{a} \subseteq \mathcal{M}$ implies that $\mathfrak{b} \subseteq \mathcal{M}$.*

For an atom $\mathfrak{a} \in \mathfrak{A}(\mathcal{O})$, its *principal ideal* $\downarrow \mathfrak{a} = \{\alpha \in \mathfrak{b} \mid \mathfrak{b} \in \mathfrak{A}(\mathcal{O}), \mathfrak{a} \succeq \mathfrak{b}\}$ is the union of all atoms it depends on, and this has been shown to be a *genuine module*, i.e., it cannot be decomposed into a union of independent modules.

3. The set of STs of an atom, a module and an atomic decomposition

In this section, we explain the foundations of using the AD for avoiding STs during classification. Using the AD, we identify a (hopefully small) set of subsumption tests

[2] We use module for it in the rest of this paper.

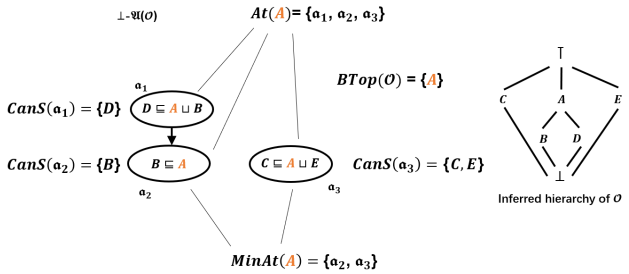


Figure 1. the example of different definitions of $\mathcal{O}$

that are sufficient for classification. Provided that the AD has a 'good' structure (no big atoms), this results in a low number of STs for a reasoner to carry out, with the potential to use delegate reasoners for these or parallelisation.

First, we fix some notation. Let A be a concept name. The set of *atoms* $\mathsf{Ats}(A)$ *of* A is defined as follows:

$$\mathsf{Ats}(A) := \{\mathfrak{a} \in \mathfrak{A}(\mathcal{O}) \mid A \in \widetilde{\mathfrak{a}}\}.$$

The set of *lowest atoms* $\mathsf{MinAts}(A)$ *of* A is defined as follows:

$$\mathsf{MinAts}(A) := \{\mathfrak{a} \in \mathsf{Ats}(A) \mid \text{ there is no } \mathfrak{b} \in \mathsf{Ats}(A) \text{ with } \mathfrak{a} \succ \mathfrak{b}\}.$$

For an atom $\mathfrak{a}$, the *candidate set of concept names to be tested* is defined as follows: $\mathsf{CanS}(\mathfrak{a})$ of $\mathfrak{a}$ is

$$\mathsf{CanS}(\mathfrak{a}) := \{A \mid \mathfrak{a} \in \mathsf{MinAts}(A) \text{ and } \#\mathsf{MinAts}(A) = 1\}.$$

Please note that $\mathsf{CanS}(\mathfrak{a})$ is a subset of the set of concept names of the atom, i.e., $\mathsf{CanS}(\mathfrak{a}) \subseteq \widetilde{\mathfrak{a}} \cap \mathbf{N_C}$.

The *set of concept names below top* $\mathsf{BTop}(\mathcal{O})$ is defined as follows:

$$\mathsf{BTop}(\mathcal{O}) := \{A \mid A \in \widetilde{\mathcal{O}} \cap \mathbf{N_C} \text{ and } \#\mathsf{MinAts}(A) > 1\}.$$

In Figure 1, these different definitions are shown in the example ontology.

Lemma 1. *For each concept name* $A \in \widetilde{\mathcal{O}}$*, we have*

1. $\widetilde{\mathcal{O}} \cap \mathbf{N_C} = \bigcup_{\mathfrak{a} \in \mathfrak{A}(\mathcal{O})} \mathsf{CanS}(\mathfrak{a}) \cup \mathsf{BTop}(\mathcal{O})$.
2. *if* $A \in \mathsf{BTop}(\mathcal{O})$*, there is no atom* $\mathfrak{a} \in \mathfrak{A}(\mathcal{O})$ *with* $A \in \mathsf{CanS}(\mathfrak{a})$.
3. *if* $A \in \mathsf{CanS}(\mathfrak{a})$*, there is no atom* $\mathfrak{b} \neq \mathfrak{a}$ *with* $A \in \mathsf{CanS}(\mathfrak{b})$.

Proof. (1). Since $\mathfrak{A}(\mathcal{O})$ partitions $\mathcal{O}$, we have that $\widetilde{\mathcal{O}} \cap \mathbf{N_C} = \bigcup_{\mathfrak{a} \in \mathfrak{A}(\mathcal{O})} \widetilde{\mathfrak{a}} \cap \mathbf{N_C}$, and thus for each concept name $A \in \widetilde{\mathcal{O}} \cap \mathbf{N_C}$, $\#\mathsf{MinAts}(A) > 0$. Let $A \in \widetilde{\mathcal{O}} \cap \mathbf{N_C}$. If $\mathsf{MinAts}(A) = 1$, we thus have some $\mathfrak{a} \in \mathfrak{A}(\mathcal{O})$ with $\mathfrak{a} \in \mathsf{MinAts}(A)$ and hence $A \in \mathsf{CanS}(\mathfrak{a})$. Otherwise, $\mathsf{MinAts}(A) > 1$ and $A \in \mathsf{BTop}(\mathcal{O})$. The "$\supseteq$" direction holds by definition of $\mathsf{CanS}(\mathfrak{a})$ and $\mathsf{BTop}(\mathcal{O})$.

(2). This follows immediately from the facts that $A \in \mathsf{BTop}(\mathcal{O})$ implies that $\#\mathsf{MinAts}(A) > 1$ and $A \in \mathsf{CanS}(\mathfrak{a})$ implies that $\#\mathsf{MinAts}(A) = 1$.

(3). Let $A \in \mathsf{CanS}(\mathfrak{a})$. By definition, $\mathfrak{a} \in \mathsf{MinAts}(A)$. Assume there was some $\mathfrak{b} \neq \mathfrak{a}$ with $A \in \mathsf{CanS}(\mathfrak{b})$; this would mean $\mathfrak{b} \in \mathsf{MinAts}(A)$, contradicting $\#\mathsf{MinAts}(A) = 1$. □

Theorem 2. *Given an ontology $\mathcal{O}$ and a concept name $A \in \mathsf{BTop}(\mathcal{O})$, we have*

1. $\mathcal{M}(\{A\}, \mathcal{O}) = \emptyset$,
2. $\mathcal{O} \not\models A \sqsubseteq \bot$, *and*
3. *there is no concept name $B \neq A$, $B \in \widetilde{\mathcal{O}}$ with $\mathcal{O} \models A \sqsubseteq B$.*

Proof. (1). Let $A \in \mathsf{BTop}(\mathcal{O})$. Hence $\#\mathsf{MinAts}(A) > 1$, and thus there are two atoms $\mathfrak{a}, \mathfrak{b} \in \mathsf{MinAts}(A)$. By definition of $\mathsf{MinAts}(A)$, $\mathfrak{a} \not> \mathfrak{b}$ and $\mathfrak{b} \not> \mathfrak{a}$. Now assume $\mathcal{M}(\{A\}, \mathcal{O}) \neq \emptyset$; hence there is some $\mathfrak{c}$ with $\mathcal{M}(\{A\}, \mathcal{O}) = \downarrow \mathfrak{c}$. Since $\bot$-locality based modules are monotonic, $\mathcal{M}(\{A\}, \mathcal{O}) \subseteq \downarrow \mathfrak{a}$ and $\mathcal{M}(\{A\}, \mathcal{O}) \subseteq \downarrow \mathfrak{b}$ which, together with $\mathfrak{a}, \mathfrak{b} \in \mathsf{MinAts}(A)$ and $A \in \widetilde{\mathfrak{c}}$, contradicts the minimality condition in the definition of $\mathsf{MinAts}(A)$.

(2). This is a direct consequence of (1) $\mathcal{M}(\{A\}, \mathcal{O}) = \emptyset$: $\bot$-locality based modules capture deductive (and model) conservativity, hence $\mathcal{M}(\{A\}, \mathcal{O}) = \emptyset$ implies that $\mathcal{O}$ cannot entail $A \sqsubseteq \bot$.

(3). This is also a direct consequence of (1) and the fact that $\bot$-locality based modules are closed under subsumers [22]. □

Next, we use the AD to identify a (hopefully small) set of STs that are sufficient for classification.

Definition 2. *The* set of STs $\mathsf{Subs}(\mathfrak{a})$ of an atom $\mathfrak{a}$ *is defined as follows:*

$$\begin{aligned}\mathsf{Subs}(\mathfrak{a}) := \{(A, B) \mid A \in \mathsf{CanS}(\mathfrak{a}),\ B \in \widetilde{\downarrow \mathfrak{a}},\ \textit{and}\ A \neq B\} \cup \\ \{(A, \bot) \mid A \in \mathsf{CanS}(\mathfrak{a})\} \cup \\ \{(\top, B) \mid B \in \mathsf{CanS}(\mathfrak{a})\}.\end{aligned}$$

Given a module $\mathcal{M} = \mathfrak{a}_1 \cup \mathfrak{a}_2 \ldots \cup \mathfrak{a}_n$, the set of STs $\mathsf{Subs}(\mathcal{M})$ of $\mathcal{M}$ *is defined as follows:*

$$\mathsf{Subs}(\mathcal{M}) := \bigcup_{i=1,\ldots,n} \mathsf{Subs}(\mathfrak{a}_i).$$

Analogously, $\mathsf{Subs}(\mathcal{O}) := \bigcup_{\mathfrak{a} \in \mathfrak{A}(\mathcal{O})} \mathsf{Subs}(\mathfrak{a})$.

The following theorem states a set of STs that tests are sufficient for classification, and thus also which can be avoided/are redundant.

Theorem 3. *For $A, B \in \widetilde{\mathcal{O}} \cap \boldsymbol{N_C}$ with $A \neq B$, $A \neq \bot$. If $\mathcal{O} \models A \sqsubseteq B$ then there is exactly one $\mathfrak{a}$ with $(A, B) \in \mathsf{Subs}(\mathfrak{a})$.*

Proof. Let A, B be as described in Theorem 3 and let $\mathcal{O} \models A \sqsubseteq B$. Then $A \notin \mathsf{BTop}(\mathcal{O})$ by Theorem 2 (3). By Lemma 1 (1), there is an atom $\mathfrak{a}$ with $A \in \mathsf{CanS}(\mathfrak{a})$. By definition, $\mathsf{CanS}(\mathfrak{a}) \subseteq \widetilde{\mathfrak{a}} \subseteq \widetilde{\downarrow \mathfrak{a}}$, and thus $A \in \widetilde{\downarrow \mathfrak{a}}$ and, by Section 2.2, $\downarrow \mathfrak{a}$ is a module. By Section 2.2, we have $B \in \widetilde{\downarrow \mathfrak{a}}$. By definition of $\mathsf{Subs}(\cdot)$, $(A, B) \in \mathsf{Subs}(\mathfrak{a})$. By Lemma 1 (3), we know there is no another atom $\mathfrak{a}$ is unique. □

In [23], we have shown that the AD of many ontologies results in many small atoms with a rather shallow and wide dependency relation. As a consequence, we should be able to exploit the AD and the insights captured in Theorem 3 to avoid almost all subsumption tests in a novel, AD-informed alternative to well-known enhanced traversal algorithms [9,10].

4. How the set of satisfiability tests of an AD interacts with delegate reasoners

Assume we have, for $1 \leq i \leq n$ modules $\mathcal{M}_i \subseteq \mathcal{O}$ that are in a specific description logic $\mathcal{L}$ for which we have a specialised, optimised reasoner.[3] Based on our observations in Section 3, we can partition our subsumption tests as follows:

$$\mathsf{Subs}(\mathcal{O}) = \bigcup_{\substack{\mathfrak{a}_6 \in \mathfrak{A}(\mathcal{O}),\\ \mathfrak{a}_6 \not\subseteq \mathcal{M}_1 \cup \ldots \cup \mathcal{M}_n}} \mathsf{Subs}(\mathfrak{a}_6) \cup \ldots \cup \bigcup_{\substack{\mathfrak{a}_7 \in \mathfrak{A}(\mathcal{O}),\\ \mathfrak{a}_7 \subseteq \mathcal{M}_1 \cup \ldots \cup \mathcal{M}_n}} \mathsf{Subs}(\mathfrak{a}_7)$$

If we have one reasoner optimised for $\mathcal{L}$, the modules $\mathcal{M}_1$, $\mathcal{M}_2 ... \mathcal{M}_n$ can be classified by this reasoner. Based on Section 2.2, if we classify the union of $\mathcal{L}$ modules $\mathcal{M}_1 \cup \ldots \cup \mathcal{M}_n$, these satisfiability tests $\bigcup_{\mathfrak{a}_7 \in \mathfrak{A}(\mathcal{O}), \mathfrak{a}_7 \subseteq \mathcal{M}_1 \cup \ldots \cup \mathcal{M}_n} \mathsf{Subs}(\mathfrak{a}_7)$ are checked. Then we just have $\bigcup_{\mathfrak{a}_6 \in \mathfrak{A}(\mathcal{O}), \mathfrak{a}_6 \not\subseteq \mathcal{M}_1 \cup \ldots \cup \mathcal{M}_n} \mathsf{Subs}(\mathfrak{a}_6)$ waiting to be checked. Next, we will explain it with examples.

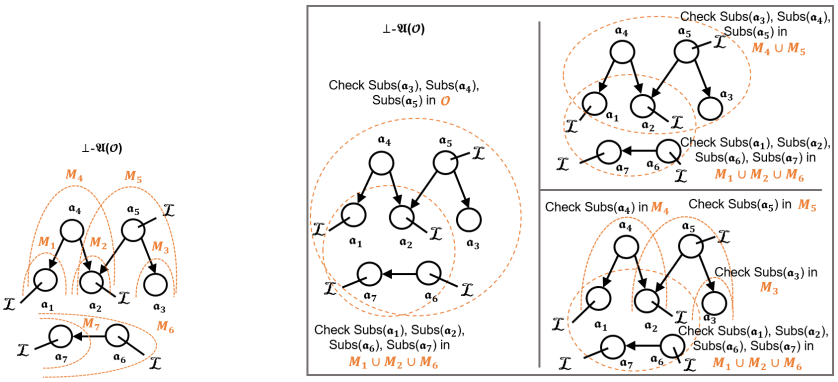
Assume we have one ontology shown in Figure 2(a) with 7 atoms and these axioms in atoms $\mathfrak{a}_1$, $\mathfrak{a}_2$, $\mathfrak{a}_5$, $\mathfrak{a}_6$, $\mathfrak{a}_7$ are in $\mathcal{L}$. We also have four modules $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_6$, $\mathcal{M}_7$ in $\mathcal{L}$. Here $\mathcal{M}_5$ is not in $\mathcal{L}$ because the atom $\mathfrak{a}_3 \subseteq \mathcal{M}_5$ is not in $\mathcal{L}$.

Now we can use a specific reasoner for $\mathcal{L}$ to classify a set of axioms $\mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_6$. Because our ontology is monotonic, classify $\mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_6$ means these subsumption tests $\mathsf{Subs}(\mathfrak{a}_1)$, $\mathsf{Subs}(\mathfrak{a}_2)$, $\mathsf{Subs}(\mathfrak{a}_6)$, $\mathsf{Subs}(\mathfrak{a}_7)$ are checked. We still have subsumption tests $\mathsf{Subs}(\mathfrak{a}_3)$, $\mathsf{Subs}(\mathfrak{a}_4)$, $\mathsf{Subs}(\mathfrak{a}_5)$ remaining. Now we can either check these tests in the ontology shown in Figure 2(b) left part or check them in several modules contain atoms $\mathfrak{a}_3$, $\mathfrak{a}_4$, $\mathfrak{a}_5$. For example, we can check $\mathsf{Subs}(\mathfrak{a}_3)$ in $\mathcal{M}_3$, check $\mathsf{Subs}(\mathfrak{a}_4)$ in $\mathcal{M}_4$ and check $\mathsf{Subs}(\mathfrak{a}_5)$ in $\mathcal{M}_5$ shown in Figure 2(b) right part. Similarly, we can also check $\mathsf{Subs}(\mathfrak{a}_3)$, $\mathsf{Subs}(\mathfrak{a}_4)$, $\mathsf{Subs}(\mathfrak{a}_5)$ in $\mathcal{M}_4 \cup \mathcal{M}_5$.

Here a union of modules sometimes is not a module. For example, if we have $\mathfrak{a}_1 = \{A \sqsubseteq B\}, \mathfrak{a}_2 = \{C \sqsubseteq D\}, \mathfrak{a}_3 = \{B \sqcap D \sqsubseteq E \sqcup F\}$ and $\mathcal{L} = \mathcal{EL}$ in our example ontology shown in Figure 2(a). Now the union of modules $\mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_6$ is not a module. Because now B, D are all in the signature of this module. Based on Section 2.2, we put the axioms which entails the subsumption relation between $B \sqcap D$ and its subsumers into the module. So if we want to fix $\mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_6$ to a module, we put $B \sqcap D \sqsubseteq E \sqcup F$ into the module. Then the module is not even in $\mathcal{EL}$. It means if we want a $\mathcal{L}$ module which is a superset of the union of $\mathcal{L}$ modules, we will have not only additional axioms in this module but also this $\mathcal{L}$ module is sometimes not in $\mathcal{L}$. The evaluation of this phenomenon is shown in Section 6.2.

In this paper, we have $\mathcal{L} = \mathcal{EL}^{++}$ [6] and we use ELK [5] for classifying the union of $\mathcal{EL}^{++}$ modules $\mathcal{M}_1 \cup \ldots \cup \mathcal{M}_n$.

[3] It is straightforward to extend this to more than one DL and more than one specialised, optimised reasoner.



(a) an AD with seven atoms in different Description Logics

(b) Different Choice for left subsumption tests

Figure 2. a example for dealing with Subs($\mathcal{O}$)

5. How the set of satisfiability tests of an AD interacts with classification algorithm

In this section, we introduce our the classification algorithm ReAD as sketched in Algorithm 1. Firstly, we compute the AD and get the union of $\mathcal{EL}^{++}$ modules $\mathcal{T}$. Then we use a reasoner specific for $\mathcal{EL}^{++}$ to classify $\mathcal{T}$ and store the subsumption relation to hierarchy $\mathcal{H}$. Then we use another reasoner for OWL DL to finish the rest of STs in left atoms. In order to do that, we modify the classification algorithm for the OWL DL reasoner. In this paper, we pick HermiT as the OWL DL reasoner. The modified classification algorithm of HermiT is shown in Algorithm 2.

Algorithm 1 AD-aware Classification

Require: an ontology $\mathcal{O}$
1: initialize a hierarchy $\mathcal{H}$
2: compute the $\bot$-$\mathfrak{A}(\mathcal{O})$
3: compute a set of atoms SetAtoms = $\{\mathfrak{a} \mid \mathfrak{a} \in \bot\text{-}\mathfrak{A}(\mathcal{O})$ and $\downarrow\mathfrak{a}$ is in $\mathcal{EL}^{++}\}$
4: compute a set of axioms $\mathcal{T} = \bigcup_{\mathfrak{a} \in \text{SetAtoms}} \mathfrak{a}$
5: Classify($\mathcal{T}$) {use a reasoner specific for $\mathcal{T}$}
6: add $A \sqsubseteq B$ into $\mathcal{H}$ if $\mathcal{T} \models A \sqsubseteq B$ and $A \neq B$
7: RemainingAtoms = $\bot$-$\mathfrak{A}(\mathcal{O})$ \ SetAtoms
8: CheckRemainingSTs(RemainingAtoms, $\bot$-$\mathfrak{A}(\mathcal{O})$, $\mathcal{H}$) {use a modified reasoner }
9: **return** $\mathcal{H}$

In Algorithm 2 we modify the classification algorithm in HermiT in order to make it AD-aware (recall that we assume that our ontology has already been tested for consistency). If $A = \top$, we will not check $\mathcal{T}' \not\models A \sqsubseteq \bot$. For every ST, HermiT uses a well-organized transitive closure and free results from ST to implement Prune() in order to avoid STs. More details about Prune() are shown in [10].

Algorithm 2 CheckRemainingSTs

Require: a set of atoms RemainingAtoms, $\bot$-$\mathfrak{A}(\mathcal{O})$, a hierarchy $\mathcal{H}$

TopAtoms = $\{\mathfrak{a} \mid \mathfrak{a} \in$ RemainingAtoms and there is no atom $\mathfrak{b} \in$ RemainingAtoms with $\mathfrak{b} \succ \mathfrak{a}\}$
compute a set of axioms $\mathcal{T}' = \bigcup_{\mathfrak{a} \in \mathsf{TopAtoms}} \downarrow \mathfrak{a}$
pre-processing
compute $\mathsf{CanS}(\mathfrak{a})$ for $\mathfrak{a} \in$ RemainingAtoms and $\mathsf{BTop}(\mathcal{O})$
for every concept name $A \in \mathsf{BTop}(\mathcal{O})$ **do**
  add $\bot \sqsubseteq A$ and $A \sqsubseteq \top$ to $\mathcal{H}$.
end for
unclassifiedCon = $\emptyset$
for every concept name $A \in \mathsf{CanS}(\mathfrak{a})$ with $\mathfrak{a} \in$ RemainingAtoms or $A = \top$ **do**
  if $\mathcal{T}' \not\models A \sqsubseteq \bot$ **then**
    for every concept name $B \in \widetilde{\downarrow \mathfrak{a}}$, and $A \neq B$ **do**
      if $A \sqsubseteq B \notin \mathcal{H}$ **then**
        Check $\mathcal{O} \models^{?} A \sqsubseteq B$. If yes, add $A \sqsubseteq B$ to $\mathcal{H}$
        Prune()
      end if
    end for
  else
    add $A \sqsubseteq \bot$ to $\mathcal{H}$
  end if
end for
return the hierarchy

6. Evaluation

In this section, we report on the empirical evaluation of our algorithm. In particular, we answer the following research questions:

1. Compared to the size of the whole ontology, what is the size of the union of $\mathcal{EL}^{++}$ modules? This will help us to understand the potential benefits of using ELK.
2. How many $\mathcal{EL}^{++}$ modules are in the union of modules? Are these modules a $\mathcal{EL}^{++}$ module? This will help us to understand potential overlap.
3. Compared to the classification time and the number of STs of HermiT, what is the classification time of ReAD and how many STs does ReAD run?

6.1. Experiment Setting

Corpus In our experiment, we use the snapshot of the NCBO BioPortal ontology repository[4] by [24], which contains 438 ontologies. Firstly, we remove ABox axioms for these 438 ontologies since we want to know how the classification algorithm behaves on the TBox axioms. Then we remove those ontologies that have only ABox axioms or are not in OWL 2 DL. We also remove those ontologies for which we cannot compute an AD or which HermiT cannot handle. To concentrate on STs and traversal algorithms, we also

[4] `https://bioportal.bioontology.org`

Table 1. A summary of 98 ontologies. The 50th (median), 90th, 95th, 99th, 100th (maximum) percentiles are shown for the size (i.e. number of axioms) and the length (i.e., sum of length of axioms) of ontologies.

	Mean	**StdDev**	**P50**	**P90**	**P95**	**P99**	**P100**
Size	7,154	23,219	911	13,223	26,540	117,375	167,022
Length	19,897	72,366	2,266	27,633	53,188	444,713	511,147

remove *deterministic* ontologies using a test provided by HermiT.[5] This leaves us with a corpus of 98 ontologies; the number of their TBox axioms and the length[6] of these ontologies is summarised in Table 1.

Implementation The implementation of ReAD is based on the OWL API [25] Version 3.4.3, especially on the implementation of the AD[7] that is part of the OWL API, namely the one available via Maven Central (maven.org) with an `artifactId` of `owlapi-tools`. We modify the code of reasoner HermiT in version 1.3.8[8] and use the reasoner ELK version 0.4.2 as a delegate reasoner. We also use the code from MORe[9] for checking $\mathcal{EL}^{++}$ axioms in used for ELK. All experiments have been performed on Intel(R) Core(TM) i7-6700HQ CPU 2.60GHz RAM 8GB, running Xms1G Xmx8G. Time is measured in CPU time.

6.2. The Size and Details of the Union of $\mathcal{L}$ modules

In this section, we answer our research Questions 1 and 2. Using Algorithm 1, we implement and compute the union of $\mathcal{EL}^{++}$ modules for these ontologies; in the following, we call this union *the $\mathcal{EL}^{++}$-part* of an ontology. We have 34 ontologies that do not contain any $\mathcal{EL}^{++}$ modules. Figure 3 shows scatter plots of the size of TBox axioms (each point) and the union of $\mathcal{EL}^{++}$ modules (each x-mark) of each ontology (with the same x-axis value) among 64 ontologies which contain at least one $\mathcal{EL}^{++}$ module in our corpus. We find that there are many ontologies in our corpus that have a large $\mathcal{EL}^{++}$-part. These ontologies also distribute evenly among different sizes.

To answer our research question 2, we consider how the number of (subset-) maximal $\mathcal{EL}^{++}$ modules in the $\mathcal{EL}^{++}$-parts varies across the ontologies in our corpus in Table 2. Among our 64 ontologies, only one ontology has only one such maximal $\mathcal{EL}^{++}$ module. In Table 2, we find most of ontologies have a large number of maximal $\mathcal{EL}^{++}$ genuine modules (average 1,701).; half of our ontologies (32) have more than 69 maximal $\mathcal{EL}^{++}$ genuine modules and 99% ontologies have 9,662 maximal $\mathcal{EL}^{++}$ genuine modules.

As discussed at the end of Section 4, a union of $\mathcal{EL}^{++}$ modules is not necessarily an $\mathcal{EL}^{++}$ module. We tested, for our 64 ontologies, whether the $\mathcal{EL}^{++}$-part is a module. It turns out that this is the case a surprisingly large proportion of our ontologies, i.e., 80% (13/64), which could be exploited further for classification.

[5] HermiT does not use a traversal algorithm for these.

[6] The length in this chapter is defined in [23] Page 24.

[7] AD implementation now is only in OWL API version 5. In our implementation we compile one to OWL API version 3

[8] The code of this version can be found in `http://www.hermit-reasoner.com`

[9] http://owlapi.sourceforge.net/

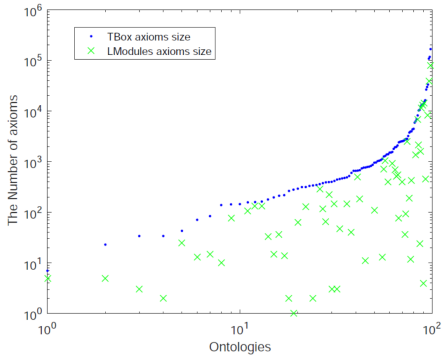


Figure 3. The size of the ontologies and their $\mathcal{L}$-part in our corpus.

Table 2. A summary of 64 ontologies. The 50th (median), 90th, 95th, 99th, 100th (maximum) percentiles are shown for the number of $\mathcal{L}$ modules of ontologies.

	Mean	**StdDev**	**P50**	**P90**	**P95**	**P99**	**P100**
The number of $\mathcal{EL}^{++}$ modules	1,701	8,311	69	1,363	5,055	9,662	66,050

6.3. The Classification Time and CTs number during Classification

In these experiments, we use HermiT version 1.3.8 to compare classification times with. First, we classify our 64 ontologies 5 times to understand the variation of classification time. Then we use ReAD to classify the 64 ontologies and record classification time data as a combination of ELK classification time and modified HermiT classification time.[10] Inspired by [26,27], we split our classification data into three bins: (1) less than 1,000ms; (2) more than 1,000ms and less than 10,000ms; (3) more than 10,000ms. Then we discuss and compare the results for each bin.

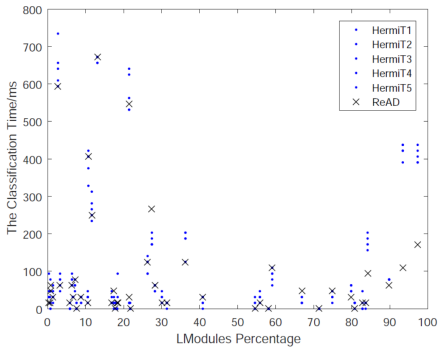
We also want to know how the $\mathcal{EL}^{++}$ modules size affect the our classification time. So we define $\mathcal{EL}^{++}$ModulesPercentage as:

$$\mathcal{EL}^{++}ModulesPercentage = \frac{\#UnionOf\mathcal{EL}^{++}Modules}{\#TBoxAxioms}$$

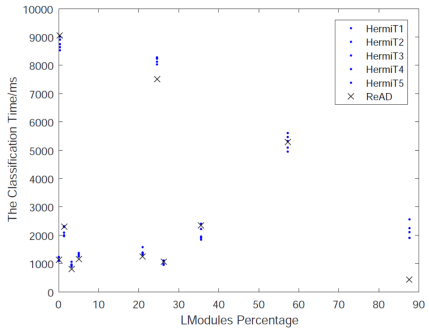
The classification times measured in CPU Time (ms) versus $\mathcal{EL}^{++}$ Modules percentage among 64 ontologies for HermiT and ReAD are shown in Figure 4. It shows scatter plots of the HermiT classification time (each point) and ReAD classification time (each x-mark) of each ontology (with the same x-axis value) among 64 ontologies. We find that for these ontologies which $\mathcal{EL}^{++}$ModulesPercentage is more than 50% trends to have a classification time improvement. We also have some ontologies which have lower than 50% $\mathcal{EL}^{++}$ModulesPercentage but also have a significant improvement, e.g. NCIT (47%) from around 75s to 47s, PHAGE (36%) from around 594s to 436s.

We also count the STs number of HermiT and modified HermiT in ReAD shown in Table 3. We find the number of STs decreases as we expected in theory.

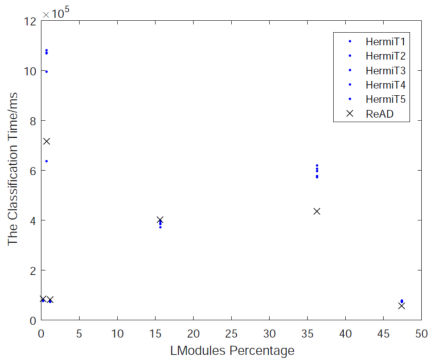
[10]This excludes the time used for computation of AD.



(a) Classification Time less than 1,000ms for HermiT and ReAD



(b) Classification Time less than 10,000ms for HermiT and ReAD



(c) Classification Time more than 10,000ms for HermiT and ReAD

Figure 4. Classification Time measured in CPU Time (ms) versus $\mathcal{EL}^{++}$ Modules percentage among 64 ontologies for HermiT and ReAD

Table 3. A summary of 64 ontologies. The 50th (median), 90th, 95th, 99th, 100th (maximum) percentiles are shown for the STs number in HermiT and modified HermiT of ontologies.

	Mean	**StdDev**	**P50**	**P90**	**P95**	**P99**	**P100**
#STs in HermiT	387	834	64	849	2,359	4,094	4,130
#STs in modified HermiT	178	528	11	481	660	2,730	3,465

7. Summary and Future Work

In this paper, we design and implement our approach for avoiding redundant STs during classification with the help of AD. We show how our algorithm interacts with the classification algorithm in HermiT. In the future, we also want to explore how our algorithm interacts with Enhanced Traversal algorithm. In Section 4, there are several choices for checking the remaining STs. In this paper, we use the union of modules of each atom $\mathfrak{a}$ if we have $\mathsf{Subs}(\mathfrak{a})$ in our remaining STs for our experiment evaluation. We can also check these STs in the whole ontology or check each $\mathsf{Subs}(\mathfrak{a})$ in its genuine module $\downarrow\mathfrak{a}$.

How to decide this is also related to easing the STs. In [28] Chapter 7, the author designs, categorizes and organizes how to make a choice of the set of axioms for STs. Using these strategies for checking left STs will tell us more about easification hypothesis: Using module than the whole ontology checking STs will make the STs easier and faster or not?

Since AD gives benefit for optimizing Classification. Optimizing the computation AD gives benefit. In [29], the author recommend that we can store the AD as a specific format for ontology. When we classify the ontology, we can use this pre-computed AD rather than computing AD every time.

References

[1] Grau BC, Horrocks I, Motik B, Parsia B, Patel-Schneider P, Sattler U. OWL 2: The next step for OWL. Journal of Web Semantics. 2008;6(4):309–322.

[2] Glimm B, Horrocks I, Motik B, Stoilos G, Wang Z. HermiT: an OWL 2 reasoner. Journal of Automated Reasoning. 2014;53(3):245–269.

[3] Steigmiller A, Liebig T, Glimm B. Konclude: system description. Journal of Web Semantics. 2014;27:78–85.

[4] Sirin E, Parsia B, Grau BC, Kalyanpur A, Katz Y. Pellet: A practical owl-dl reasoner. Journal of Web Semantics. 2007;5(2):51–53.

[5] Kazakov Y, Krötzsch M, Simančík F. The incredible elk. Journal of automated reasoning. 2014;53(1):1–61.

[6] Baader F, Brandt S, Lutz C. Pushing the $\mathcal{EL}$ envelope further. In: OWLED; 2008. .

[7] Horrocks I, Kutz O, Sattler U. The Even More Irresistible $\mathcal{SROIQ}$. In: KR-06; 2006. p. 57–67.

[8] Hustadt U, Motik B, Sattler U. Data complexity of reasoning in very expressive description logics. In: IJCAI; 2005. p. 466–471.

[9] Baader F, Hollunder B, Nebel B, Profitlich HJ, Franconi E. An empirical analysis of optimization techniques for terminological representation systems. Applied Intelligence. 1994;4(2):109–132.

[10] Glimm B, Horrocks I, Motik B, Shearer R, Stoilos G. A novel approach to ontology classification. Journal of Web Semantics. 2012;14:84–101.

[11] Grau BC, Parsia B, Sirin E, Kalyanpur A. Modularity and Web Ontologies. In: KR; 2006. p. 198–209.

[12] Romero AA, Kaminski M, Grau BC, Horrocks I. Module extraction in expressive ontology languages via datalog reasoning. Journal of Artificial Intelligence Research. 2016;55:499–564.

[13] Lutz C, Walther D, Wolter F. Conservative Extensions in Expressive Description Logics. In: IJCAI; 2007. p. 453–458.

[14] Ghilardi S, Lutz C, Wolter F. Did I Damage My Ontology? A Case for Conservative Extensions in Description Logics. In: KR. AAAI Press; 2006. p. 187–197.

[15] Cuenca Grau B, Horrocks I, Kazakov Y, Sattler U. A Logical Framework for Modularity of Ontologies. In: IJCAI; 2007. p. 298–303.

[16] Del Vescovo C, Parsia B, Sattler U, Schneider T. The Modular Structure of an Ontology: Atomic Decomposition. In: IJCAI; 2011. p. 2232–2237.

[17] Tsarkov D, Palmisano I. Chainsaw: a Metareasoner for Large Ontologies. In: ORE; 2012. .

[18] Romero AA, Grau BC, Horrocks I. MORe: Modular combination of OWL reasoners for ontology classification. In: International Semantic Web Conference. Springer; 2012. p. 1–16.

[19] Matentzoglu N, Parsia B, Sattler U. OWL reasoning: Subsumption test hardness and modularity. Journal of automated reasoning. 2018;60(4):385–419.

[20] Zhao H, Parsia B, Sattler U. Avoiding Subsumption Tests During Classification Using the Atomic Decomposition. In: DL-19. vol. 573; 2019. .

[21] Baader F, Horrocks I, Lutz C, Sattler U, editors. An Introduction to Description Logic. Cambridge University Press; 2017.

[22] Cuenca Grau B, Horrocks I, Kazakov Y, Sattler U. Modular reuse of ontologies: Theory and practice. Journal of Artificial Intelligence Research. 2008;31(1):273–318.

[23] Del Vescovo C, Horridge M, Parsia B, Sattler U, Schneider T, Zhao H. Modular Structures and Atomic Decomposition in Ontologies (to be published). Journal of Artificial Intelligence Research.

[24] Matentzoglu N, Parsia B. BioPortal Snapshot 30 March 2017 (data set). Zenodo; 2017. http://doi.org/10.5281/zenodo.439510.
[25] Horridge M, Bechhofer S. The owl api: A java api for owl ontologies. Semantic web. 2011;2(1):11–21.
[26] Goncalves JR. Impact analysis in description logic ontologies. The University of Manchester; 2014.
[27] Gonçalves RS, Parsia B, Sattler U. Performance heterogeneity and approximate reasoning in description logic ontologies. In: International Semantic Web Conference. Springer; 2012. p. 82–98.
[28] Matentzoglu NA. Module-based classification of OWL ontologies. University of Manchester; 2016.
[29] Del Vescovo C, Gessler DD, Klinov P, Parsia B, Sattler U, Schneider T, et al. Decomposition and modular structure of bioportal ontologies. In: International Semantic Web Conference. Springer; 2011. p. 130–145.