History Matters: Incremental Ontology Reasoning Using Modules

Bernardo Cuenca Grau[1], Christian Halaschek-Wiener[2], and Yevgeny Kazakov[1]

[1] The University of Manchester, School of Computer Science, Manchester, M13 9PL, UK
[2] Department of Computer Science, University of Maryland, College Park, MD 20740, USA

Abstract. The development of ontologies involves continuous but relatively small modifications. Existing ontology reasoners, however, do not take advantage of the similarities between different versions of an ontology. In this paper, we propose a technique for incremental reasoning—that is, reasoning that reuses information obtained from previous versions of an ontology—based on the notion of a module. Our technique does not depend on a particular reasoning calculus and thus can be used in combination with any reasoner. We have applied our results to incremental classification of OWL DL ontologies and found significant improvement over regular classification time on a set of real-world ontologies.

1 Introduction

The design and maintenance of OWL ontologies are highly complex tasks. The support of a reasoner is crucial for detecting modeling errors, which typically manifest themselves as concept unsatisfiability and unintended subsumption relationships.

The development of ontologies involves continuous but relatively small modifications. Even after a number of changes, an ontology and its previous version usually share most of their axioms. Unfortunately, when an ontology evolves, current reasoners do not take advantage of the similarities between the ontology and its previous version. That is, when reasoning over the latest version of an ontology, current reasoners do not reuse existing results already obtained for the previous one and repeat the whole reasoning process. For large and complex ontologies this may require a few minutes, or even a few hours. If the response of the reasoner is too slow, ontology engineers may end up not using the reasoner as often as they would wish. For ontology development and maintenance tasks it is important to detect possible errors as soon as possible; for such a purpose, the reasoner should be executed often and real time response from the reasoner becomes an important issue.

In this paper, we propose a technique for incremental ontology reasoning—that is, reasoning that reuses the results obtained from previous computations. Our technique is based on the notion of a *module* and can be applied to arbitrary queries against ontologies expressed in OWL DL. We focus on a particular kind of modules that exhibit a set of compelling properties and apply our method to incremental classification of OWL DL ontologies. Our techniques do not depend on a particular reasoner or reasoning method and could be easily implemented in any existing prover, such as Pellet, FaCT++, KAON2 or RACER. Our empirical results using Pellet[3] show substantial performance improvements over regular classification time.

2 Preliminaries

We introduce the syntax of the description logic $\mathcal{SHOIQ}$ [11], which provides the logical underpinning for OWL DL.

A $\mathcal{SHOIQ}$-signature is the disjoint union $\mathbf{S} = \mathbf{R} \uplus \mathbf{C} \uplus \mathbf{I}$ of sets of *atomic roles* (denoted by $R, S, \cdots$), *atomic concept* (denoted by $A, B, \cdots$) and *nominals* (denoted by $a, b, c, \cdots$). A $\mathcal{SHOIQ}$*-role* is either $R \in \mathbf{R}$ or an *inverse role* R^- with $R \in \mathbf{R}$. We denote by $\mathbf{Rol}$ the set of $\mathcal{SHOIQ}$-roles for the signature $\mathbf{S}$. The set $\mathbf{Con}$ of $\mathcal{SHOIQ}$-*concepts* for $\mathbf{S}$ is defined by the following grammar:

$$\mathbf{Con} ::= \bot \mid a \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid \exists R.C \mid \geqslant n\, S.C$$

where $a \in \mathbf{I}$, $A \in \mathbf{C}$, $C_{(i)} \in \mathbf{Con}$, $R, S \in \mathbf{Rol}$, with S a *simple* role,[4] and n a positive integer. We use the following abbreviations: $C \sqcup D$ stands for $\neg(\neg C \sqcap \neg D)$; $\top$ stands for $\neg\bot$; $\forall R.C$ stands for $\neg(\exists R.\neg C)$; and $\leqslant n\, S.C$ stands for $\neg(\geqslant n{+}1\, S.C)$.

A $\mathcal{SHOIQ}$ ontology $\mathcal{O}$ is a finite set of *role inclusion axioms* (RIs) $R_1 \sqsubseteq R_2$ with $R_i \in \mathbf{Rol}$, *transitivity axioms* $\mathsf{Trans}(R)$ with $R \in \mathbf{R}$ and *general concept inclusion axioms* (GCIs) $C_1 \sqsubseteq C_2$ with $C_i \in \mathbf{Con}$.[5] The concept definition $A \equiv C$ is an abbreviation for the two GCIs $A \sqsubseteq C$ and $C \sqsubseteq A$. The signature $\mathsf{Sig}(\alpha)$ of an axiom α is the union $\mathsf{RN}(\alpha) \cup \mathsf{CN}(\alpha) \cup \mathsf{Ind}(\alpha)$ of atomic roles, atomic concepts, and nominals that occur in α. The signature $\mathsf{Sig}(\mathcal{O})$ of an ontology $\mathcal{O}$ is defined analogously.

For the semantics of $\mathcal{SHOIQ}$, we refer the interested reader to [11].

3 The Challenge for Incremental Reasoning in Ontologies

Consider the medical ontology $\mathcal{O}^1$ given in Table 1, which consists of three concept definitions D1 – D3 and two inclusion axioms C1 – C2. For exposition, suppose that an ontology engineer in charge of this ontology notices that the definition D1 for the concept $\mathsf{Cystic\_Fibrosis}$ is incomplete and reformulates it by adding the new conjunct $\exists\mathsf{has\_Origin}.\mathsf{Genetic\_Origin}$. As a result, a new version $\mathcal{O}^2$ of the ontology is obtained. In order to ensure that no errors have been introduced by this change, the ontology engineer uses a reasoner to classify the new ontology $\mathcal{O}^2$.

Table 2 shows some subsumption relationships between atomic concepts in $\mathcal{O}^1$ and $\mathcal{O}^2$, which should be computed for classification. We can see that some of these subsumption relations have changed as a result of a modification in the ontology: axiom α_1 follows from the axioms D3, C2 and D1 in $\mathcal{O}^1$, but does not follow from $\mathcal{O}^2$ anymore since D1 has been modified; in contrast, the subsumption α_2, which did not follow from $\mathcal{O}^1$, is now a consequence of the modified D1, D2 and C1 in $\mathcal{O}^2$. Other subsumptions

[3]Pellet Homepage: `http://pellet.owldl.com`.

[4]See [11] for a precise definition of simple roles.

[5]Note that ABox assertions $a : C$ can be expressed in $\mathcal{SHOIQ}$ using GCIs $a \sqsubseteq C$.

	Original Ontology $\mathcal{O}^1$:	**Modified Ontology $\mathcal{O}^2$:**
D1	$\mathsf{Cystic\_Fibrosis} \equiv \mathsf{Fibrosis} \sqcap \exists \mathsf{located\_In.Pancreas}$	$\mathsf{Cystic\_Fibrosis} \equiv \mathsf{Fibrosis} \sqcap \exists \mathsf{located\_In.Pancreas} \sqcap \boldsymbol{\exists \mathsf{has\_Origin.Genetic\_Origin}}$
D2	$\mathsf{Genetic\_Fibrosis} \equiv \mathsf{Fibrosis} \sqcap \exists \mathsf{has\_Origin.Genetic\_Origin}$	$\mathsf{Genetic\_Fibrosis} \equiv \mathsf{Fibrosis} \sqcap \exists \mathsf{has\_Origin.Genetic\_Origin}$
D3	$\mathsf{Pancreatic\_Fibrosis} \equiv \mathsf{Fibrosis} \sqcap \mathsf{Pancreatic\_Disorder}$	$\mathsf{Pancreatic\_Fibrosis} \equiv \mathsf{Fibrosis} \sqcap \mathsf{Pancreatic\_Disorder}$
C1	$\mathsf{Genetic\_Fibrosis} \sqsubseteq \mathsf{Genetic\_Disorder}$	$\mathsf{Genetic\_Fibrosis} \sqsubseteq \mathsf{Genetic\_Disorder}$
C2	$\mathsf{Pancreatic\_Disorder} \sqsubseteq \mathsf{Disorder} \sqcap \exists \mathsf{located\_In.Pancreas}$	$\mathsf{Pancreatic\_Disorder} \sqsubseteq \mathsf{Disorder} \sqcap \exists \mathsf{located\_In.Pancreas}$
$\boldsymbol{\Delta\mathcal{O}} = \mathbf{diff}(\boldsymbol{\mathcal{O}^1}, \boldsymbol{\mathcal{O}^2}) = (\boldsymbol{\Delta^-\mathcal{O}}, \boldsymbol{\Delta^+\mathcal{O}})$		
$\Delta^-\mathcal{O} =$ $\mathsf{Cystic\_Fibrosis} \equiv \mathsf{Fibrosis} \sqcap \exists \mathsf{located\_In.Pancreas}$		
$\Delta^+\mathcal{O} =$ $\mathsf{Cystic\_Fibrosis} \equiv \mathsf{Fibrosis} \sqcap \exists \mathsf{located\_In.Pancreas} \sqcap \boldsymbol{\exists \mathsf{has\_Origin.Genetic\_Origin}}$		

Table 1. Evolution of a Bio-Medical Ontology $\mathcal{O}$

$\boldsymbol{\alpha}$	**Axiom:**	$\boldsymbol{\mathcal{O}^1 \overset{?}{\models} \alpha}$**,**	**follows from:**	$\boldsymbol{\mathcal{O}^2 \overset{?}{\models} \alpha}$**,**	**follows from:**
α_1	$\mathsf{Pancreatic\_Fibrosis} \sqsubseteq \mathsf{Cystic\_Fibrosis}$	Yes	D3, C2, **D1**	No	—
α_2	$\mathsf{Cystic\_Fibrosis} \sqsubseteq \mathsf{Genetic\_Disorder}$	No	—	Yes	**D1**, D2, C1
α_3	$\mathsf{Pancreatic\_Fibrosis} \sqsubseteq \mathsf{Disorder}$	Yes	D3, C2	Yes	D3, C2
α_4	$\mathsf{Genetic\_Fibrosis} \sqsubseteq \mathsf{Cystic\_Fibrosis}$	No	—	No	—

Table 2. Subsumption Relations Before and After the Change

such as α_3 and α_4 did not change: α_3 is a consequence of axioms D3 and C2 which have not been modified; α_4 follows neither from $\mathcal{O}^1$ nor from $\mathcal{O}^2$.

It is reasonable to expect that small changes in ontologies will not affect many subsumption relations. That is, the number of subsumptions that change their entailment status w.r.t. the ontology, like, say, α_1 or α_2 in Table 2, is probably small compared to the number of subsumptions that do not, like α_3 or α_4. If so, then many (possibly expensive) re-computations can be avoided by reusing the subsumption relations computed for the previous version of the ontology. In order to realize this idea, one has to identify which subsumptions could be affected by a change and which are not.

Suppose we know that a subsumption α holds in $\mathcal{O}^1$. Then we can guarantee that α still holds in $\mathcal{O}^2$ provided the axioms from which α follows in $\mathcal{O}^1$ have not been modified. For example, in Table 2, the subsumption α_3 is a consequence of axioms D3 and C2, both of which have not been modified in $\mathcal{O}^2$. Hence, we can conclude that α_3 holds in $\mathcal{O}^2$ without performing reasoning over $\mathcal{O}^2$. In contrast, this test is not applicable for the subsumption α_1, since α_1 is a consequence of axioms D3, C2 and D1 in $\mathcal{O}^1$, and D1 has been modified in $\mathcal{O}^2$. In this case, the status of α_1 in $\mathcal{O}^2$ has to be computed by other means, e.g. using a reasoner. Thus, the status of every subsumption relation α that holds in $\mathcal{O}^1$ requires re-computation for $\mathcal{O}^2$ only if in every justification for α (every minimal subset of $\mathcal{O}^1$ which implies α) some axiom has been modified. This approach is reminiscent of the way Truth Maintenance Systems (TMS) maintain logical dependencies between axioms [6, 3]. The notion of justification for an axiom has also been used for pinpointing the axioms responsible for errors in ontologies, such as unsatisfiable concepts and unintended subsumptions [14, 13].

The situation is principally different in the case of subsumptions α that do *not* hold in $\mathcal{O}^1$. In this case, if to follow the previous approach, one has to keep track of "evidences" for *non-entailments* of subsumptions in ontologies and verify if at least one such "evidence" for α in $\mathcal{O}^1$ can be reused in $\mathcal{O}^2$. Here, the "evidence" might be, for example, a (part of a) counter-model for α in $\mathcal{O}^1$ that is constructed by tableau-based procedures. Such techniques based on *model caching* have been recently proposed in the context incremental reasoning [8]. These techniques, however, have only been applied so far to additions and deletions of ABox assertions, since changes in general axioms often require considerable modifications of the models. Moreover, such techniques require close interaction with the model construction routine of the tableau reasoner, which precludes their use in arbitrary "off-the-shelf" reasoners without considerable modifications. In particular, these techniques cannot be directly used in reasoners like KAON2, which are not tableaux-based.

We stress that the challenge for incremental ontology reasoning is mainly to maintain non-subsumptions since, in typical ontologies, almost 99% of subsumption relations between atomic concepts do not hold. In other words, the case of axiom α_4 in Figure 2 is likely to be the most one after a change in an ontology.[6]

In this paper we propose an alternative approach for incremental reasoning based on the module-extraction techniques introduced in [2]. Our technique can be used to keep track of "evidences" for *both* subsumptions and non-subsumptions modulo arbitrary changes in ontologies, and works in combination with any DL-reasoner providing for standard reasoning services.

4 Modules and Syntactic Locality

In this section we define the notion of a module [2], which underlies our technique for incremental reasoning. We also outline the algorithm proposed in [2] for extracting a particular kind of modules, called *locality-based* modules.

Definition 1 (Module for an Axiom and a Signature). *Let $\mathcal{O}$ be an ontology and $\mathcal{O}_1 \subseteq \mathcal{O}$ is a (possibly empty) subset of axioms in $\mathcal{O}$. We say that $\mathcal{O}_1$ is a* module for for an axiom *α in $\mathcal{O}$ (or short, an α*-module in *$\mathcal{O}$) if: $\mathcal{O}_1 \models \alpha$ iff $\mathcal{O} \models \alpha$.*

We say that $\mathcal{O}_1$ is a module for a signature $\mathbf{S}$ *if for every axiom α with* $\mathsf{Sig}(\alpha) \subseteq \mathbf{S}$, *we have that $\mathcal{O}_1$ is a module for α in $\mathcal{O}$.*

Intuitively, a module for an axiom α in an ontology $\mathcal{O}$ is a subset $\mathcal{O}_1$ of $\mathcal{O}$ which contains the axioms that are "relevant" for α in $\mathcal{O}$, in the sense that $\mathcal{O}$ implies α if and only if $\mathcal{O}_1$ implies α. In case $\mathcal{O}$ implies α, then every module $\mathcal{O}_1$ for α should contain

[6]In Section 6 we provide empirical evidences confirming our conjectures

at least one justification for α (that is, a minimal set of axioms which imply α). In case $\mathcal{O}$ does not imply α (that is, there are no justifications for α), $\mathcal{O}_1$ can be any subset of $\mathcal{O}$. Hence, knowing all the justifications for α in $\mathcal{O}$ is sufficient for identifying all modules for α in $\mathcal{O}$.

The notion of module *for a signature* has been introduced in [2]. Intuitively, a module for a signature is a subset of the ontology that is a module for every axiom constructed over this signature. An algorithm for extracting modules based on a notion of syntactic locality was proposed in [2], and it was empirically verified that this algorithm extracts reasonably small modules in existing ontologies.

Definition 2 (Syntactic Locality for $\mathcal{SHOIQ}$). *Let* $\mathbf{S}$ *be a signature. The following grammar recursively defines two sets of concepts* $\mathbf{Con}^{\emptyset}(\mathbf{S})$ *and* $\mathbf{Con}^{\Delta}(\mathbf{S})$ *for* $\mathbf{S}$*:*

$$\begin{aligned}\mathbf{Con}^{\emptyset}(\mathbf{S}) &::= A^{\emptyset} \mid (\neg C^{\Delta}) \mid (C^{\emptyset} \sqcap C) \mid (C \sqcap C^{\emptyset}) \\ &\quad\mid (\exists R^{\emptyset}.C) \mid (\exists R.C^{\emptyset}) \mid ({\geqslant}\, n\, R^{\emptyset}.C) \mid ({\geqslant}\, n\, R.C^{\emptyset})\,. \\ \mathbf{Con}^{\Delta}(\mathbf{S}) &::= (\neg C^{\emptyset}) \mid (C^{\Delta}_1 \sqcap C^{\Delta}_2)\,.\end{aligned}$$

where $A^{\emptyset} \notin \mathbf{S}$ *is an atomic concept,* $R^{\emptyset}$ *is (possibly inverse of) an atomic role* $r^{\emptyset} \notin \mathbf{S}$*,* C *is any concept,* R *is any role, and* $C^{\emptyset} \in \mathbf{Con}^{\emptyset}(\mathbf{S})$*,* $C^{\Delta}_{(i)} \in \mathbf{Con}^{\Delta}(\mathbf{S})$*,* $i = 1, 2$*.*

An axiom α *is* local w.r.t. $\mathbf{S}$ *if it is of one of the following forms:* (1) $R^{\emptyset} \sqsubseteq R$*, or* (2) $\mathsf{Trans}(R^{\emptyset})$*, or* (3) $C^{\emptyset} \sqsubseteq C$ *or* (4) $C \sqsubseteq C^{\Delta}$*.*[7]

Intuitively, an axiom α is syntactically local w.r.t. $\mathbf{S}$ if, by simple syntactical simplifications, one can demonstrate that α is true in every interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ in which concept and atomic roles not from $\mathbf{S}$ are interpreted with the empty set. For example, the axiom D2 from Table 1 is local w.r.t. $\mathbf{S} = \{\mathsf{Fibrosis}, \mathsf{has\_Origin}\}$: if we interpret the remaining symbols in this axiom with the empty set, we obtain a model of the axiom, independently of the interpretation of the symbols in $\mathbf{S}$.

$$\overbrace{\mathsf{Genetic\_Fibrosis}}^{\emptyset} \equiv \underline{\mathsf{Fibrosis}} \sqcap \underbrace{\exists \underline{\mathsf{has\_Origin}}.\overbrace{\mathsf{Genetic\_Origin}}^{\emptyset}}_{\emptyset}$$

If an ontology $\mathcal{O}$ can be partitioned as $\mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_s$ such that every axiom in $\mathcal{O}_s$ is syntactically local w.r.t. $\mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_1)$, then $\mathcal{O}_1$ is a module for $\mathbf{S}$ in $\mathcal{O}$ [2]. Algorithm 1 extracts a module $\mathcal{O}_1$ for a signature $\mathbf{S}$ from an ontology $\mathcal{O}$ using this property. The procedure first initializes $\mathcal{O}_1$ to the empty set and then iteratively moves to $\mathcal{O}_1$ those axioms α from $\mathcal{O}$ that are not local w.r.t. $\mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_1)$ until all such axioms have been moved. We assume that $\mathrm{s\_local}(\alpha, \mathbf{S})$ tests for syntactic locality of an axiom α w.r.t. signature $\mathbf{S}$ according to Definition 2. In Table 3 we provide a trace of Algorithm 1 for the input ontology $\mathcal{O}^1$ in Table 1 and signature $\mathbf{S} = \{\mathsf{Pancreatic\_Fibrosis}\}$.

Proposition 1 (Correctness of Algorithm 1 (see [2]) for details).
Given an $\mathcal{SHOIQ}$ *ontology* $\mathcal{O}$ *and a signature* $\mathbf{S}$*, Algorithm 1 terminates in polynomial time in the size of* $\mathcal{O}$ *and returns a module* $\mathcal{O}_1$ *for* $\mathbf{S}$ *in* $\mathcal{O}$*.*

[7]Recall that $\forall R.C$, $({\leqslant}\, n\, R.C)$ and $C_1 \sqcup C_2$ are expressed using the other constructors, so they can be used in local axioms as well.

Algorithm 1 extract_module($\mathcal{O}, \mathbf{S}$)

```
Input:
    $\mathcal{O}$: ontology
    $\mathbf{S}$: signature
Output:
    $\mathcal{O}_1$: a module for $\mathbf{S}$ in $\mathcal{O}$
 1: $\mathcal{O}_1 \leftarrow \emptyset \quad \mathcal{O}_2 \leftarrow \mathcal{O}$
 2: while not empty($\mathcal{O}_2$) do
 3:     $\alpha \leftarrow$ select_axiom($\mathcal{O}_2$)
 4:     if s_local($\alpha$, $\mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_1)$) then
 5:         $\mathcal{O}_2 \leftarrow \mathcal{O}_2 \setminus \{\alpha\}$
 6:     else
 7:         $\mathcal{O}_1 \leftarrow \mathcal{O}_1 \cup \{\alpha\}$
 8:         $\mathcal{O}_2 \leftarrow \mathcal{O} \setminus \mathcal{O}_1$
 9:     end if
10: end while
11: return $\mathcal{O}_1$
```

A sample trace for the Algorithm 1 for $\mathcal{O} = \mathcal{O}^1$ from Table 1 and $\mathbf{S} = \{\mathsf{Pancreatic\_Fibrosis}\}$:

	$\mathcal{O}_1$	$\mathcal{O}_2$	New $X \in \mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_1)$	α	loc?
1	–	D1, D2, D3, C1, C2	Pancreatic_Fibrosis	D3	No
2	D3	D1, D2, C1, C2	Fibrosis, Pancreatic_Disorder	D1	Yes
3	D3	D2, C1, C2	–	D2	Yes
4	D3	C1, C2	–	C1	Yes
5	D3	C2	–	C2	No
6	D3, C2	D1, D2, C1,	Disorder, located_In, Pancreas	D1	Yes
7	D3, C2	D2, C1	–	D2	Yes
8	D3, C2	C1	–	C1	Yes
9	D3, C2	–	–	–	

Table 3. An algorithm for extracting syntactic locality-based modules from ontologies.

In order to extract a module for an axiom α in $\mathcal{O}$ it is sufficient to run Algorithm 1 for $\mathbf{S} = \mathsf{Sig}(\alpha)$. However, when α is a subsumption between atomic concepts, $\top$ or $\bot$, it suffices to extract a module only for $\mathbf{S} = \mathsf{Sig}(X)$, as given in the following proposition.

Proposition 2 (see [2] for details). *Let $\mathcal{O}$ be a $\mathcal{SHOIQ}$ ontology, $X, Y \in \mathsf{CN}(\mathcal{O}) \cup \{\top\} \cup \{\bot\}$, and $\mathcal{O}_X$ the output of Algorithm 1 for input $\mathcal{O}$ and $\mathbf{S} = \mathsf{Sig}(X)$. Then $\mathcal{O}_X$ is a module in $\mathcal{O}$ for $\alpha = (X \sqsubseteq Y)$.*

Finally, we point out that the modules extracted using Algorithm 1 are not necessary minimal ones. That is, if $\mathcal{O} \models \alpha$, the computed module for α might be a strict superset of a justification for α in $\mathcal{O}$, and if $\mathcal{O} \not\models \alpha$ then the module for $\mathsf{Sig}(\alpha)$ might not necessarily be the empty set. In fact, if α is not a tautology, computing a minimal module for α in $\mathcal{O}$ is at least as hard as checking whether $\mathcal{O} \not\models \alpha$ since $\mathcal{O} \models \alpha$ iff the minimal module for α is empty. The last problem is computationally expensive for many ontology languages, including OWL DL. The advantage of the module-extraction algorithm described in this section is that, on the one hand, it runs in polynomial and, on the other hand, it still generates reasonably small modules.

5 Incremental Classification Using Locality-Based Modules

In this section we show how to use then notion of module for incremental reasoning over ontologies. First, we outline the general idea behind using modules for incrementally maintaining (non)entailment of axioms and then describe an algorithm for incremental classification of ontologies using locality-based modules, as described in Section 4.

The following proposition, which is a simple consequence of Definition 1, provides the basic property underlying incremental reasoning using modules:

Proposition 3. *Let $\mathcal{O}^1$, $\mathcal{O}^2$ be ontologies, α an axiom, and $\mathcal{O}^1_\alpha$, $\mathcal{O}^2_\alpha$ respectively modules for α in $\mathcal{O}^1$ and $\mathcal{O}^2$. Then:*

1. *If $\mathcal{O}^1 \models \alpha$ and $\mathcal{O}^1_\alpha \subseteq \mathcal{O}^2$, then $\mathcal{O}^2 \models \alpha$*
2. *If $\mathcal{O}^1 \not\models \alpha$ and $\mathcal{O}^2_\alpha \subseteq \mathcal{O}^1$, then $\mathcal{O}^2 \not\models \alpha$*

Proposition 3 suggests that, in order to test if the entailment of an axiom α has not been affected by a change $\mathcal{O}^1 \Rightarrow \mathcal{O}^2$, it is sufficient to compute, depending on whether $\mathcal{O}^1 \models \alpha$ or $\mathcal{O}^1 \not\models \alpha$, a module $\mathcal{O}^1_\alpha$ for α in $\mathcal{O}^1$, or a module $\mathcal{O}^2_\alpha$ for α in $\mathcal{O}^2$ respectively. If the change does not involve any of the axioms in the module, then the status of the entialment of α also does not change. The converse of this is not necessarily true: even if the corresponding module has been modified, the status of α might still remain unaffected. For example, the axiom $\alpha = (\mathsf{Cystic\_Fibrosis} \sqsubseteq \mathsf{Fibrosis})$ follows from D1 both before and after the change, even though D1 has been modified. In such a case, the status of α w.r.t. $\mathcal{O}^2$ should be verified using the reasoner. The use of modules, however, is also valuable in this situation: instead of checking if α follows from $\mathcal{O}^2$, one could equivalently check if α follows from the (hopefully much smaller) module $\mathcal{O}^2_\alpha$.

Therefore, the use of modules provides two compelling advantages for incremental reasoning: first, the computation of a given query may be avoided and the answer can be simply reused from a previous test; second, even if the query needs to be performed, the use of modules allows for filtering out irrelevant axioms and reduces the search space.

Note that the sizes of modules $\mathcal{O}^1_\alpha$ and $\mathcal{O}^2_\alpha$ have a direct impact on the quality of the incremental entailment test for α. The smaller the modules, the more likely it is that they do not contain the modified axioms. Nevertheless, as pointed out in Section 4, computing a smallest possible module is computationally expensive: it is at least as hard as just checking whether $\mathcal{O}^1 \models \alpha$ (respectively $\mathcal{O}^2 \models \alpha$). Thus, there is a trade-off between the complexity of computing a module on the one hand, and its usefulness for incremental reasoning on the other hand. Intuitively, the smaller the module, the more useful and the harder it is to compute. We demonstrate empirically that Algorithm 1 computes small enough modules to be useful for incremental reasoning.

In the remainder of this section we apply the general idea for incremental reasoning sketched above for incremental classification of ontologies using the module-extraction procedure given by Algorithm 1. Classification of an ontology $\mathcal{O}$ amounts to computing *subsumption relations* $X \sqsubseteq Y$ where X and Y range over all atomic concepts from $\mathcal{O}$, $\bot$, and $\top$. The relations are non-trivial when $X \in \mathsf{CN}(\mathcal{O}) \cup \{\top\}$ and $Y \in \mathsf{CN}(\mathcal{O}) \cup \{\bot\}$. As shown in Proposition 2, in order to check incrementally a subsumption relation $\alpha = (X \sqsubseteq Y)$, it is sufficient to keep track of the modules $\mathcal{O}_X$ for $\mathsf{Sig}(X)$ in $\mathcal{O}$.

Consider the ontologies $\mathcal{O}^1$ and $\mathcal{O}^2$ in Table 1 and the axioms α_1–α_4 in Table 2. Each of these axioms is of the form $\alpha = (X \sqsubseteq Y)$, with X and Y atomic concepts. Table 4 provides the locality-based modules for α_1–α_4 in $\mathcal{O}^1$ and in $\mathcal{O}^2$ computed using Algorithm 1. Note that the modules are not minimal: in our case, they are strict supersets of the actual minimal modules from Table 2 where the additional axioms are underlined. The modules for axioms α_1–α_3 have been changed, whereas the module for the axiom α_4 has remained unchanged. Hence, the sufficient test for preservation

α	**Axiom** $X \sqsubseteq Y$:	$\mathcal{O}^1_X$	$\mathcal{O}^2_X$
α_1	Pancreatic_Fibrosis	D3,C2,**D1**	
	$\sqsubseteq$ Cystic_Fibrosis		$\underline{\text{D3}}$,$\underline{\text{C2}}$
α_2	Cystic_Fibrosis	$\underline{\textbf{D1}}$	
	$\sqsubseteq$ Genetic_Disorder		**D1**,D2,C1
α_3	Pancreatic_Fibrosis	D3,C2,$\underline{\textbf{D1}}$	
	$\sqsubseteq$ Disorder		D3,C2
α_4	Genetic_Fibrosis	$\underline{\text{C1}}$	
	$\sqsubseteq$ Cystic_Fibrosis		$\underline{\text{C1}}$

X	$\mathcal{O}^1_X$	$\mathcal{O}^2_X$
Cystic_Fibrosis	**D1**	**D1**,D2,C1
Fibrosis	$\emptyset$	$\emptyset$
Pancreas	$\emptyset$	$\emptyset$
Genetic_Fibrosis	C1	C1
Genetic_Origin	$\emptyset$	$\emptyset$
Pancreatic_Fibrosis	D3,C2,**D1**	D3,C2
Pancreatic_Disorder	C2	C2
Genetic_Disorder	C1	C1
Disorder	C2	C2
$\top$	$\emptyset$	$\emptyset$

Table 4. Modules For Subsumptions and Concept Names in Ontologies from Table 1

of (non)subsumptions using modules gave us only one "false positive" for subsumption α_3, where the subsumption relation did not change, but the modules have been modified.

The right part of Table 4 provides the full picture on the modules and their changes for our example ontology from Table 1. The only modules that have been changed are the ones for X = Cystic_Fibrosis and X = Pancreatic_Fibrosis, where for the first module axiom D1 has been changed, and for in the second module axiom D1 has been removed. Applying Proposition 2 and Proposition 3 we can conclude that every subsumption that dissapears as a the result of the change should be either of the form $\alpha = (\mathsf{Cystic\_Fibrosis} \sqsubseteq Y)$ or $\alpha = (\mathsf{Pancreatic\_Fibrosis} \sqsubseteq Y)$, and every subsumption that can appear should be of the form $\alpha = (\mathsf{Cystic\_Fibrosis} \sqsubseteq Y)$.

Algorithm 2 outlines an incremental classification procedure based on the ideas just discussed. Given an ontology $\mathcal{O}^1$ and a change $\Delta\mathcal{O} = (\Delta^-\mathcal{O}, \Delta^+\mathcal{O})$ consisting of the sets of removed and added axioms, the algorithm computes the subsumption partial order $\sqsubseteq_2$ for the resulting ontology $\mathcal{O}^2 = (\mathcal{O}^1 \setminus \Delta^-\mathcal{O}) \cup \Delta^+\mathcal{O}$ by reusing the one $\sqsubseteq_1$ already computed for $\mathcal{O}^1$. In order to perform this operation, the algorithm internally maintains the modules $\mathcal{O}^1_X$ and $\mathcal{O}^2_X$ for every atomic concept or the top concept X. We will show that mantaining these additional modules does not involve a significant overhead in practice. The algorithm consists of the following phases:

1. *Process the new symbols (lines 2–6):* The modules $\mathcal{O}^1_X$ and the subsumption partial order $\sqsubseteq_1$ for $\mathcal{O}^1$ are extended for every newly introduced atomic concept A. The module for A, about which nothing has been said yet, is equivalent to the module for the empty signature–that, is the module for $\top$. Thus, we have: (i) $\mathcal{O}^1_A = \mathcal{O}^1_\top$, (ii) $\mathcal{O}^1 \models A \sqsubseteq Y$ iff $\mathcal{O}^1 \models \top \sqsubseteq Y$, and (iii) $\mathcal{O}^1 \models X \sqsubseteq A$ iff $\mathcal{O}^1 \models X \sqsubseteq \bot$.
2. *Identifying the affected modules (lines 7–19):* The sets M^- and M^+ contain those $X \in \mathsf{CN}(\mathcal{O}^1) \cup \{\top\}$ for which the corresponding modules must be modified by removing and/or adding axioms. If α removed from $\mathcal{O}^1$ is non-local w.r.t. $\mathsf{Sig}(\mathcal{O}^1_X)$ then at least α should be removed from $\mathcal{O}^1_X$. If α is added to $\mathcal{O}^1$ and is non-local w.r.t. $\mathsf{Sig}(\mathcal{O}^1_X)$, then the module $\mathcal{O}^1_X$ needs to be extended at least with α.
3. *Computing new modules and subsumptions (lines 20–34):* The affected modules found in the previous phase are re-extracted and those that are not are just copied

Algorithm 2 inc_classify($\mathcal{O}^1, \Delta\mathcal{O}, \sqsubseteq_1, X \to \mathcal{O}^1_X$)

Input:

$\mathcal{O}^1$: an ontology
$\Delta\mathcal{O} = (\Delta^-\mathcal{O}, \Delta^+\mathcal{O})$: removed / added axioms
$\sqsubseteq_1$: subsumption relations in $\mathcal{O}^1$
$X \to \mathcal{O}^1_X$: a module for every $X \in \mathsf{CN}(\mathcal{O}^1) \cup \{\top\}$

Output:

$\mathcal{O}^2$: the result of applying the change $\Delta\mathcal{O}$ to $\mathcal{O}^1$
$\sqsubseteq_2$: subsumption relations in $\mathcal{O}^2$
$X \to \mathcal{O}^2_X$: a module for every $X \in \mathsf{CN}(\mathcal{O}^2) \cup \{\top\}$

```
1: $\mathcal{O}^2 \leftarrow (\mathcal{O}^1 \setminus \Delta^-\mathcal{O}) \cup \Delta^+\mathcal{O}$
2: for each $A \in \mathsf{CN}(\mathcal{O}^2) \setminus \mathsf{CN}(\mathcal{O}^1)$ do
3:     $\mathcal{O}^1_A \leftarrow \mathcal{O}^1_\top$
4:     for each $\top \sqsubseteq_1 Y$ do $A \sqsubseteq_1 Y \leftarrow$ true
5:     for each $X \sqsubseteq_1 \bot$ do $X \sqsubseteq_1 A \leftarrow$ true
6: end for
7: $\mathrm{M}^- \leftarrow \emptyset \quad \mathrm{M}^+ \leftarrow \emptyset$
8: for each $X \in \mathsf{CN}(\mathcal{O}^2) \cup \{\top\}$ do
9:     for each $\alpha \in \Delta^-\mathcal{O}$ do
10:        if not s_local($\alpha, \mathsf{Sig}(\mathcal{O}^1_X)$) then
11:            $\mathrm{M}^- \leftarrow \mathrm{M}^- \cup \{X\}$
12:        end if
13:    end for
14:    for each $\alpha \in \Delta^+\mathcal{O}$ do
15:        if not s_local($\alpha, \mathsf{Sig}(\mathcal{O}^1_X)$) then
16:            $\mathrm{M}^+ \leftarrow \mathrm{M}^+ \cup \{X\}$
17:        end if
18:    end for
19: end for
20: for each $X \in \mathsf{CN}(\mathcal{O}^2) \cup \{\top\}$ do
21:    if $X \in \mathrm{M}^- \cup \mathrm{M}^+$ then
22:        $\mathcal{O}^2_X \leftarrow$ extract_module($\mathsf{Sig}(X), \mathcal{O}^2$)
23:    else
24:        $\mathcal{O}^2_X \leftarrow \mathcal{O}^1_X$
25:    end if
26:    for each $Y \in \mathsf{CN}(\mathcal{O}^2) \cup \{\bot\}$ do
27:        if ($X \in \mathrm{M}^-$ and $X \sqsubseteq_1 Y$) or
28:          ($X \in \mathrm{M}^+$ and $X \not\sqsubseteq_1 Y$) then
29:            $X \sqsubseteq_2 Y \leftarrow$ test($\mathcal{O}^2_X \models X \sqsubseteq Y$)
30:        else
31:            $X \sqsubseteq_2 Y \leftarrow X \sqsubseteq_1 Y$
32:        end if
33:    end for
34: end for
35: return $\mathcal{O}^2, \sqsubseteq_2, X \to \mathcal{O}^2_X$
```

(lines 21–25). Then, every subsumption $X \sqsubseteq Y$, using Proposition 3, is either re-computed against the module $\mathcal{O}^2_X$, or is reused from $\mathcal{O}^1$ (lines 26–33).

In Algorithm 5, the procedure extract_module($\mathbf{S}$, $\mathcal{O}$) refers to Algorithm 1 in Section 4. The procedure test($\mathcal{O} \models X \sqsubseteq Y$) uses a reasoner to check if $\mathcal{O}$ entails the subsumption $X \sqsubseteq Y$. The correctness of the algorithm is easy to prove using Proposition 2 and Proposition 3.

It is worth emphasizing that, in our algorithm, the reasoner is only used as a black box to answer subsumption queries; this provides two important advantages: on the one hand, the internals of the reasoner need not be modified and, on the other hand, *any* sound and complete reasoner for OWL DL can be plugged in, independently of the reasoning technique it is based on (e.g. tableaux or resolution).

To conclude, we illustrate the execution of Algorithm 5 on the ontologies $\mathcal{O}^1, \mathcal{O}^2$ in Table 1, where the sets $\Delta^{-}\mathcal{O}$ and $\Delta^{+}\mathcal{O}$ of removed and added axioms for our example are given in the lower part of Table 1. In our case, $\mathcal{O}^2$ doesn't introduce new atomic concepts w.r.t. $\mathcal{O}^1$. Thus, Phase 1 in Algorithm 2 can be skipped. The sets $\mathrm{M}^-, \mathrm{M}^+$ computed in Phase 2 are as follows: $\mathrm{M}^- = \{\mathsf{Cystic\_Fibrosis}, \mathsf{Pancreatic\_Fibrosis}\}$ and $\mathrm{M}^+ = \{\mathsf{Cystic\_Fibrosis}\}$ since the axiom in $\Delta^{-}\mathcal{O}$ (see Table 1) is not sytactically local w.r.t. the signature of the module in $\mathcal{O}^1$ for Cystic_Fibrosis and Pancreatic_Fibrosis; analogously, the axiom in $\Delta^{+}\mathcal{O}$ is non-local w.r.t. the signature of the module in $\mathcal{O}^2$ for Cystic_Fibrosis. In Phase 3, the modules for Cystic_Fibrosis and Pancreatic_Fibrosis are re-computed. In the former module, the algorithm recomputes only the subsumption relations between Cystic_Fibrosis and Pancreatic_Fibrosis and their subsumers in $\mathcal{O}^1$; in the latter one, the only the subsumption relations between the non-subsumers of Cystic_Fibrosis in $\mathcal{O}^1$ are computed.

6 Empirical Evaluation

We have implemented Algorithm 2 and used the OWL reasoner Pellet for evaluation. Our implementation is, however, independent from Pellet, and our results intend to determine the usefulness of our approach for optimizing any reasoner. Our system implements a slightly more simplistic procedure than the one in Algorithm 2; in particular, once the affected modules have been identified, our implementation simply reclassifies the union of these modules using Pellet to determine the new subsumption relations, instead of using the procedure described in lines 20–34 of Algorithm 2.

As a test suite, we have selected a set of well-known ontologies that are currently being developed. NCI[8], and the Gene Ontology[9] are expressed in a simple fragment of OWL DL. In contrast, GALEN[10], and NASA's SWEET ontology[11] are written in a more expressive language. Table 5 includes their expressivity, number of atomic concepts and axioms, total classification time in Pellet, and the percentage of possible subsumption relations that actually hold between atomic concepts. Note that for large ontologies,

[8]http://www.mindswap.org/2003/CancerOntology/nciOncology.owl
[9]http://www.geneontology.org
[10]http://www.openclinical.org/prj_galen.html
[11]http://sweet.jpl.nasa.gov/ontology/

over 99% of subsumpton relations do not hold. Table 5 also shows the average time to extract the modules for all atomic concepts, as well as the average and maximum size of these modules (in terms of the number of axioms). Even if the initial module extraction may introduce overhead, we argue that this "startup-cost" is bearable since the set of all modules needs only be computed once. We observe that, in general, the modules are very small relative to the size of the ontology.

Ontology	Logic	♯ Concept Names	♯ Axioms	Class. Time (s.)	% Subs	Init. Mod. Extract (s.)	Mod. Size (Avg/Max)	Non-Loc. Axioms
SWEET	$\mathcal{SHOIF}$	1400	2573	3.6	0.37	1.05	76 / 420	28
Galen	$\mathcal{SHF}$	2749	4529	15.7	0.37	4.8	75 / 530	0
GO	$\mathcal{EL}$	22357	34980	63	0.04	69.6	17.6 / 161	0
NCI	$\mathcal{EL}$	27772	46940	41.1	0.03	76.5	28.9 / 436	0

Table 5. Test suite ontologies.

We have performed the following experiment for each ontology: for various numbers *n*, we have **1)** removed *n* random axioms; **2)** classified the resulting ontology using Pellet; then, we have repeated the following two steps 50 times: **3)** extracted the minimal locality-based module for each atomic concept, **4)** removed an additional *n* axioms, added back the previously removed *n* axioms, and reclassified the ontology using our incremental algorithm. Our goal is to simulate the ontology evolution process where n axioms are changed (which can be viewed as a simultaneous deletion and addition); all results have been gathered during step **4)** of the experiment. We considered different types of axioms, namely concept definitions, GCIs and role axioms.

	n	1: ♯ Mod. Affected (Av / Mx)	2: ♯ Axioms in Aff. Mod. (Av / Mx)	3: Update Aff. Mod. (Av / Mx)	4: Re-class. Aff. Mod. (Av / Mx)	5. Total Time (Av / Mx)	6: ♯ New (Non)Sub. (Av / Mx)	7: ♯ Mod. (Non)Sub (Av / Mx)
NCI	2	67 / 936	545 / 3025	.81 / 5.4	.21 / 1.2	1.03 / 6.77	54 / 1268	17 / 348
SWEET	2	36.9 / 300	281 / 857	.097 / .929	.182 / 1.4	.280 / 2.3	39 / 686	20.1 / 255
Galen	2	134 / 1045	1003 / 2907	.833 / 3.6	2.8 / 13	3.6 / 16.5	111 / 1594	17 / 158
GO	1	39.2 / 1513	127 / 1896	.24 / 1.4	.05 / .47	.29 / 1.5	69 / 2964	33 / 1499
GO	2	46 / 891	216 / 1383	.5 / 2.8	.07 / .43	.57 / 3.2	51 / 1079	26 / 775
GO	4	97 / 1339	474 / 3021	1.4 / 10.1	.25 / 3.2	1.7 / 13.4	94 / 1291	44 / 1034

Table 6. Results for varying update sizes for class and role axioms. Time in seconds.

Table 6 summarizes the results of the experiments for $n = 2$. Columns 1 and 2 detail the number of affected modules and their total size respectively. It can be observed that, in general, only a very small number of the modules are affected for a given update. Column 3 provides the total time to locate and re-extract the affected modules; Column 4 shows the reclassification time for all the affected modules after they have been re-extracted. In all cases, the average time is significantly smaller than standard re-classification. It can be observed that, in the case of Galen, the maximum time to classify the affected modules actually takes longer than classifying the entire ontology. While unexpected, this is likely caused as traditional classification optimizations (e.g., model merging, top-bottom search, etc.) are not as effective, due to affected modules containing a subset of the original axioms; therefore, additional subsumption checks have to be performed. We note, however, that on average this does not occur. Column 5 presents the total time to update the modules, load them into the reasoner, and reclassify them; it can be seen that this outperforms reclassifying from scratch. For future work, we plan to more tightly integrate the approach into Pellet, as this will avoid the additional overhead attributed to loading the affected modules into the reasoner for classification. Column 6 shows the number of new subsumption and non-subsumption relations (i.e., the sum) for each ontology, and column 7 provides the average number of modules which have a new subsumption or non-subsumption after a change. The number of new (non) subsumptions is very small, which supports our initial hypothesis that changes do not typically affect a large portion of the original ontology. In the case of SWEET, the ratio of modules with new (non)subsumptions is relatively high when compared to the average number of modules affected; specifically in these cases, almost 50% of the affected modules actually contains a new subsumption/non-subsumption relation after the update. This empirically demonstrates that locality-based modules can be very effective for maintaining (non)subsumptions relations as the underlying ontology changes. Finally, the last two rows of the Table show the results for $n = 1, 2, 4$ in the case of the Gene Ontology. These results suggests that incremental classification time may grow linearly with the number of modified axioms; similar behavior can be observed for the remaining ontologies.

	n	1: ♯ Mod. Affected (Av / Mx)	2: ♯ Axioms in Aff. Mod. (Av / Mx)	3: Update Aff. Mod. (Av / Mx)	4: Re-class. Aff. Mod. (Av / Mx)	5. Total Time (Av / Mx)	6: ♯ New (Non)Sub. (Av / Mx)	7: ♯ Mod. (Non)Sub (Av / Mx)
NCI	2	2274 / 10217	12161 / 29091	25.7 / 60.4	10.4 / 30.8	36.2 / 91.3	0 / 0	0 / 0
SWEET	2	116 / 296	411 / 956	.42 / .93	.6 / 1.4	1.03 / 2.33	.56 / 28	.28 / 14
Galen	2	524 / 1906	1813 / 3780	2.1 / 4.7	6.5 / 15.6	8.6 / 20.4	3.3 / 82	2.5 / 37

Table 7. Results for varying update sizes for role axiom changes only. Time in seconds.

Table 7 considers the particular case of changes to role axioms only[12]. As shown in Table 7, for SWEET the results are comparable to those presented in Table 6. For NCI and Galen, changes in role axioms do have a more substantial impact.

The particular case of changes to concept axioms only is provided in Table 8[13]. It can be observed that the are much better than those when only role axiom changes are performed. These results confirm that role axioms may cause larger effects than changes in concept definitions.

[12]GO has not been included in Table 7 as it only contains one role axiom

[13]Again GO has not been included in Table 8 as it only contains one role axiom

	n	1: ♯ Mod. Affected (Av / Mx)	2: ♯ Axioms in Aff. Mod. (Av / Mx)	3: Update Aff. Mod. (Av / Mx)	4: Re-class. Aff. Mod. (Av / Mx)	5. Total Time (Av / Mx)	6: ♯ New (Non)Sub. (Av / Mx)	7: ♯ Mod. (Non)Sub (Av / Mx)
NCI	2	33 / 847	396 / 5387	.59 / 8.7	.15 / 2.6	.75 / 11.4	67 / 2228	20 / 610
SWEET	2	15.2 / 243	276 / 800	.02 / .07	.07 / .65	.095 / .732	31 / 553	12 / 241
Galen	2	131 / 1463	913 / 3397	.84 / 4.5	2.6 / 15.4	3.4 / 19.5	69 / 4323	42 / 1178

Table 8. Results for varying update sizes for concept axiom changes only. Time in seconds.

7 Related Work

While there has been substantial work on optimizing reasoning services for description logics (see [10] for an overview), the topic of reasoning through evolving DL knowledge bases remains relatively unaddressed. Notable exceptions include [7–9, 12]; these papers, however, investigate the problem of incremental reasoning using model-caching techniques in application scenarios that involve changes *only* in the ABox.

There has been substantial work on incremental query and view maintenance in databases (e.g., [1, 15, 16]) and rule-based systems (e.g., Datalog [4, 5]). While related, our work addresses a more expressive formalism; further, traditionally in database systems the problem of incremental maintenance is considered with respect to data (corresponding to DL ABoxes) and not with respect to the database schema (corresponding to DL TBoxes). Our technique, however, focuses on schema reasoning.

There has additionally been extensive work in Truth Maintenance Systems (TMSs) for logical theories (e.g., [3, 6]). As pointed out in Section 3, a justification-based approach would be advantageous for incremental classification only if the number of positive subsumptions was larger than the number of non-subsumptions; that is, if most of the formulas the justifications keep track of were provable. This is, however, not the case, as typically there are far more non-subsumptions than subsumptions. Additionally, a TMS system designed to support non-subsumptions (e.g., by caching models) would most likely be impractical due to the potentially large size of these models and substantial modifications likely to be caused by changes in general axioms; however, in our approach, maintaining locality-based modules introduces limited overhead. Finally, the representation language in practical TMSs is mostly propositional logic, whereas we focus on much more expressive languages.

8 Conclusion

We have proposed a general technique for incremental reasoning under arbitrary changes in an ontology. We have used locality-based modules due to their compelling properties and applied our method to incremental classification of OWL DL ontologies.

For ontology development, it is desirable to re-classify the ontology after a small number of changes. In this scenario, our results are very promising. Incremental classification using modules is nearly real-time for almost all ontologies and therefore the reasoner could be working transparently to the user in the background without slowing down the editing of the ontology. There are, however, some disadvantages of our approach. First, there are cases where a change which does not affect the concept hierarchy, affects a large number of modules; second, for complex ontologies including nominals, such as the Wine ontology, the modules can be large; third classifying a (large enough) fragment might be more expensive than classifying the whole ontology. In most cases, however, our incremental approach provides a substantial speed-up w.r.t. regular classification. For future work, we are planning to exploit modules for incremental ABox reasoning tasks, such as query answering.

Acknowledgements: This work is supported by the EU Project TONES (Thinking ONtologieES) ref:IST-007603 and by the EPSRC Project REOL (Reasoning in Expressive Ontology Languages) ref:EP/C537211/1. We would like to acknowledge Evren Sirin and Mike Smith for their contributions to the empirical evaluation presented here.

References

1. J. A. Blakeley, P.-A. Larson, and F. W. Tompa. Efficiently updating materialized views. In *Proc. of SIGMOD '86: ACM SIGMOD International Conference on Management of Data*, pages 61–71, 1986.
2. B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Just the right amount: Extracting modules from ontologies. In *Proc. of WWW2007*, 2007.
3. J. de Kleer. An assumption-based TMS. *Artif. Intell.*, 28(2):127–162, 1986.
4. G. Dong, J. Su, and R. W. Topor. Nonrecursive incremental evaluation of datalog queries. *Annals of Mathematics and Artificial Intelligence*, 14(2-4), 1995.
5. G. Dong and R. W. Topor. Incremental evaluation of datalog queries. In *Proc. of the 4th Int. Conference on Database Theory*, 1992.
6. J. Doyle. A truth maintenance system. *Readings in nonmonotonic reasoning*, pages 259–279, 1987.
7. V. Haarslev and R. Möller. Incremental query answering for implementing document retrieval services. In *Proc. of DL-2003*, pages 85–94, 2003.
8. C. Halaschek-Wiener and J. Hendler. Toward expressive syndication on the web. In *Proc. of the 16th Int. World Wide Web Conference (WWW 2007)*, 2007.
9. C. Halaschek-Wiener, B. Parsia, and E. Sirin. Description logic reasoning with syntactic updates. In *Proc. of ODBase2006*, 2006.
10. I. Horrocks. Implementation and optimisation techniques. In *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 313–355. Cambridge University Press, 2003.
11. I. Horrocks and U. Sattler. A tableaux decision procedure for SHOIQ. In *Proc. of IJCAI 2005*, pages 448–453. Professional Book Center, 2005.
12. B. Parsia, C. Halaschek-Wiener, and E. Sirin. Towards incremental reasoning through updates in OWL-DL. Reasoning on the Web - Workshop at WWW-2006, 2006.
13. B. Parsia, E. Sirin, and A. Kalyanpur. Debugging OWL ontologies. In *WWW*, pages 633–640, 2005.
14. S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proc. of IJCAI, 2003*, pages 355–362. Morgan Kaufmann, 2003.
15. M. Stonebraker. Implementation of integrity constraints and views by query modification. In *SIGMOD '75: Proc. of the 1975 ACM SIGMOD international conference on Management of data*, pages 65–78, New York, NY, USA, 1975.
16. D. B. Terry, D. Goldberg, D. Nichols, and B. M. Oki. Continuous queries over append-only databases. In *Proc. of the Intl. Conf. on Management of Data*, 1992.