A Modularization-based Approach to Finding All Justifications for OWL DL Entailments

Boontawee Suntisrivaraporn[1]and Guilin Qi[2] and Qiu Ji[2] and Peter Haase[2]

[1] Theoretical Computer Science, TU Dresden, Germany
meng@tcs.inf.tu-dresden.de
[2] AIFB Institute, University of Karlsruhe, Germany
{gqi,qiji,pha}@aifb.uni-karlsruhe.de

Abstract. Finding the justifications for an entailment (i.e., minimal sets of axioms responsible for it) is a prominent reasoning service in ontology engineering, as justifications facilitate important tasks like debugging inconsistencies or undesired subsumption. Though several algorithms for finding all justifications exist, issues concerning efficiency and scalability remain a challenge due to the sheer size of real-life ontologies. In this paper, we propose a novel method for finding all justifications in OWL DL ontologies by limiting the search space to smaller modules. To this end, we show that so-called locality-based modules cover all axioms in the justifications. We present empirical results that demonstrate an improvement of several orders of magnitude in efficiency and scalability of finding all justifications in OWL DL ontologies.

1 Introduction

Since the Web Ontology Language (OWL) has become a W3C standard, it is widely believed that ontologies play a prominent role in formal representation of knowledge on the Semantic Web. The main advantages of employing OWL in knowledge engineering are twofold. On the one hand, the well-defined semantics of Description Logic (DL), which is the logical underpinning of OWL, helps guarantee that everyone on the Web understands the described knowledge in a consistent way. On the other hand, reasoning services can be exploited to derive implicit knowledge from the one explicitly given. DL systems can, for example, identify unsatisfiable concepts and classify a given ontology, i.e., compute all the subsumption (subconcept–superconcept) relationships between the concepts defined in the ontology. These "standard" reasoning services have proved essential but not sufficient in engineering real-world ontologies. This is because building ontologies is an error-prone endeavor. Although most DL systems can detect an error (an unsatisfiable concept or undesired subsumption) in a given ontology, additional reasoning is needed in order to find its *justifications*, i.e., minimal subsets of the ontology that still have the error.

Several techniques for finding all justifications have been proposed in the literature in the past decade which can be categorized into glass-box approaches and black-box approaches.

Glass-box approaches require the decision (e.g., tableau) procedure to be modified, usually by adding labels to keep track of relevant axioms used during the computation [14, 12, 11, 1, 2]. Most of the work in this direction considers specific Description Logics, e.g., $\mathcal{ALC}$, and a specific type of entailment, e.g., concept unsatisfiability. In [14], Schlobach and Cornet proposed an extension to the tableau algorithm for $\mathcal{ALC}$ with unfoldable TBoxes. The extension uses labels to keep track of axioms used during the computation which directly corresponds to justifications. They also coined the name "axiom pinpointing" for the task of finding justifications for an entailment. Since glass-box approaches are based on modifying the internals of a DL reasoning algorithm, an extension has to be developed for each DL. Meyer et al. extended the idea to $\mathcal{ALC}$ with general concept inclusions (GCIs) [12], and Kalyanpur et al. extended it to the more expressive DL $\mathcal{SHIF}(\mathcal{D})$ [11] and $\mathcal{SHOIN}(\mathcal{D})$ [10] which underly the core of OWL. In [1], a general approach for extending a tableau-based algorithm to a pinpointing algorithm is proposed which can be used to find all justifications for a given entailment. Most previous work on glass-box methods considers tableau-based reasoning algorithm. An exception is the work by Baader et al. [2] which extends the polytime classification algorithm in order to compute justifications for a subsumption relation in the lightweight DL $\mathcal{EL}^+$, and also shows that axiom pinpointing is inherently hard, i.e., determining whether there is a justification within a given cardinality bound is NP-complete despite tractability of the underlying DL.

The other class of approaches to axiom pinpointing is known as *black-box*, where a DL reasoner is merely used to test specific entailment queries, and as such its internals need not be modified. With a naïve pruning algorithm, a justification can be computed by invoking the DL reasoner linear number of times [11, 2]. The naïve algorithm essentially sweeps through all the axioms in the ontology and tests if the entailment still holds in absence of each axiom. Since this approach is independent from reasoning algorithms, it can be easily implemented on top of any existing DL reasoners. The main disadvantage, however, is that it typically requires several calls to the DL reasoning services that are already computationally expensive. Therefore, several optimization techniques have very recently been proposed that help to reduce the number of calls to the DL reasoner and hence speed up the black-box approach. Examples include the 'sliding window' technique employed in the fast pruning algorithm [10], the 'binary-search' idea adapted to obtain a best-case logarithmic pruning algorithm [3], and the 'relevance-based selection function' that syntactically select relevant axioms from the ontology [9]. Based on a black-box pruning algorithm for computing a single justification, the hitting set tree (HST) algorithm [13, 10, 9] can be used to recursively compute all justifications.

Recently, ontology modularity and modularization have been studied extensively, with various applications ranging from ontology re-use and optimization of classical reasoning such as subsumption, as well as non-classical reasoning such as incremental classification [5] and axiom pinpointing [3]. Closely related to [9] is the modularization-based approach to axiom pinpointing where relevant axioms are precisely those axioms in the module [3]. In order to exploit modu-

Name	Syntax	Semantics
top	$\top$	$\Delta^{\mathcal{I}}$
concept name	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$
negation	$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
exists restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
at-least restriction	$\geq n\, s.C$	$\{x \in \Delta^{\mathcal{I}} \mid \sharp\{y : (x,y) \in s^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \geq n\}$
role name	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
inverse role	r^-	$\{(x,y) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (y,x) \in r^{\mathcal{I}}\}$
role hierarchy	$r \sqsubseteq s$	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
transitivity	$\mathsf{Trans}(r)$	$(x,y),(y,z) \in r^{\mathcal{I}}$ implies $(x,z) \in r^{\mathcal{I}}$
GCI	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

Table 1. Syntax and semantics of $\mathcal{SHOIQ}$ concepts and axioms.

larity in black-box axiom pinpointing, Baader and Suntisrivaraporn showed that the reachability-based module [16] covers all justifications for an entailment of interest in $\mathcal{EL}^+$ [3].

In the present paper, we combine the relevance-based techniques developed in [9] and the modularization-based techniques in [3] to effectively enhance the HST pinpointing algorithm. Since the results in [3] are w.r.t. reachability-based modules for $\mathcal{EL}^+$, we need to adopt the locality-based module [6] for $\mathcal{SHOIQ}$. Our main contributions in the present paper are twofold. In theory, we show that the minimal locality-based module is a *subsumption module* (first defined in [3]), i.e., it covers all justifications. As a consequence, it suffices to focus on axioms in the module when finding *all* justifications and when testing subsumption. In practice, we have implemented the approach using KAON2 as the black-box reasoner and evaluated it on realistic ontologies. Our empirical results demonstrate an improvement of several orders of magnitude in the efficiency and scalability of finding all justifications. The results thus render the black-box approach feasible for application-scale OWL DL ontologies.

2 Preliminaries

In this section, we give formal definitions for $\mathcal{SHOIQ}$ ontologies, justifications and locality-based modules. Then, we introduce selection functions and the HST pinpointing algorithm.

Description logic and justifications

To make the paper self-contained, we first introduce the Description Logic (DL) $\mathcal{SHOIQ}$ [7] which is the underpinning DL formalism of the Web Ontology Language (OWL DL and OWL Lite).

Starting with disjoint sets of concept names CN, role names RN and individuals Ind, a $\mathcal{SHOIQ}$-role is either a role name $r \in \mathsf{RN}$ or an inverse role r^-

with $r \in \mathsf{RN}$. We denote by Rol the set of all $\mathcal{SHOIQ}$-roles. $\mathcal{SHOIQ}$-concepts can be built using the constructors shown in the upper part of Table 1, where $a \in \mathsf{Ind}, r, s \in \mathsf{Rol}$ with s a *simple role*[1], n is a positive integer, $A \in \mathsf{CN}$, and C, D are $\mathcal{SHOIQ}$-concepts.[2] We use the standard abbreviations: $\bot$ stands for $\neg\top$; $C \sqcup D$ stands for $\neg(\neg C \sqcap \neg D)$; $\forall r.C$ stands for $\neg(\exists r.\neg C)$; and $\leq ns.C$ stands for $\neg(\geq (n+1)s.C)$. We denote by Con the set of all $\mathcal{SHOIQ}$-concepts.

A $\mathcal{SHOIQ}$ ontology $\mathcal{O}$ is a finite set of *role hierarchy axioms* $r \sqsubseteq s$, *transitivity axioms* $\mathsf{Trans}(r)$, and a general concept inclusion axioms (GCIs) $C \sqsubseteq D$ with $r, s \in \mathsf{Rol}$ and $C, D \in \mathsf{Con}$.[3] We write $\mathsf{CN}(\mathcal{O})$, $\mathsf{RN}(\mathcal{O})$ and $\mathsf{Ind}(\mathcal{O})$ to denote, respectively, the set of concept names, role names and individuals occurring in the the ontology $\mathcal{O}$, and $\mathsf{Sig}(\mathcal{O})$ to denote the signature of $\mathcal{O}$, i.e., $\mathsf{CN}(\mathcal{O}) \cup \mathsf{RN}(\mathcal{O}) \cup \mathsf{Ind}(\mathcal{O})$. Similarly, $\mathsf{Sig}(r)$, $\mathsf{Sig}(C)$ and $\mathsf{Sig}(\alpha)$ are used to denote the signature of a role, a concept and an axiom, respectively.

The DL semantics is defined by means of interpretations $\mathcal{I}$ with a non-empty domain $\Delta^{\mathcal{I}}$ and a function $\cdot^{\mathcal{I}}$ that maps each concept $C \in \mathsf{Con}$ to a subset of the domain and each role $r \in \mathsf{Rol}$ to a binary relation over the domain. An interpretation $\mathcal{I}$ is a *model* of an ontology $\mathcal{O}$ ($\mathcal{I} \models \mathcal{O}$), if the conditions given in the semantics column of Table 1 are satisfied. The main types of entailments are concept satisfiability: C is *satisfiable* w.r.t. $\mathcal{O}$ if there exists a model $\mathcal{I}$ of $\mathcal{O}$ such that $C^{\mathcal{I}} \neq \emptyset$; and concept subsumption: C *is subsumed by* D w.r.t. $\mathcal{O}$ (written $\mathcal{O} \models C \sqsubseteq D$ or $C \sqsubseteq_{\mathcal{O}} D$) if, for every model $\mathcal{I}$ of $\mathcal{O}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Without loss of generality, we restrict attention to concept subsumption in what follows. Considering an example ontology depicted in Figure 1, all DL reasoners are able to detect that the subsumption $\mathcal{O}_{\mathsf{ex}} \models \sigma = (\mathsf{Endocarditis} \sqsubseteq \mathsf{HeartDisease})$ holds.

Definition 1 (Justification). *Let $\mathcal{O}$ be a $\mathcal{SHOIQ}$ ontology with an entailment σ (i.e., $\mathcal{O} \models \sigma$). A subset $J \subseteq \mathcal{O}$ is a* justification for σ in $\mathcal{O}$ *if $J \models \sigma$ and, for every $J' \subset J$, $J' \not\models \sigma$.*

Justifications for an entailment need not be unique. Moreover, given an ontology and an entailment, the number of justifications may be exponential in the size of the ontology. For the small example ontology $\mathcal{O}_{\mathsf{ex}}$ (see 1), it is not difficult to infer that there are precisely two justifications for σ: one consisting of axioms marked by $\bullet$, and the other by $\star$.

Modularization

We now introduce the notions of *syntactic locality* and *locality-based module*, which have been first introduced in [6]. Syntactic locality is used to define the notion of module for a signature, i.e., a subset of the ontology that preserves the meaning of names in the signature.

[1] A simple role is neither transitive nor a superrole of a transitive role.

[2] Concepts and roles in DL correspond to classes and properties in OWL, respectively.

[3] A concept definition $A \equiv C$ is an abbreviation of two GCIs $A \sqsubseteq C$ and $C \sqsubseteq A$, while ABox assertions $C(a)$ and $r(a, b)$ can be expressed as the GCIs $\{a\} \sqsubseteq C$ and $\{a\} \sqsubseteq \exists r.\{b\}$, respectively.

α_1	$\mathsf{Pericardium} \sqsubseteq \mathsf{Tissue} \sqcap \exists \mathsf{part\text{-}of}.\mathsf{Heart}$		
$\underline{\alpha_2}$	$\mathsf{Endocardium} \sqsubseteq \mathsf{Tissue} \sqcap \exists \mathsf{part\text{-}of}.\mathsf{HeartValve} \sqcap \exists \mathsf{part\text{-}of}.\mathsf{HeartWall}$	•	⋆
$\underline{\alpha_3}$	$\mathsf{HeartValve} \sqsubseteq \mathsf{BodyValve} \sqcap \exists \mathsf{part\text{-}of}.\mathsf{Heart}$	•	
$\underline{\alpha_4}$	$\mathsf{HeartWall} \sqsubseteq \mathsf{BodyWall} \sqcap \exists \mathsf{part\text{-}of}.\mathsf{Heart}$		⋆
α_5	$\mathsf{Pericarditis} \sqsubseteq \mathsf{Inflammation} \sqcap \exists \mathsf{has\text{-}loc}.\mathsf{Pericardium}$		
$\underline{\alpha_6}$	$\mathsf{Endocarditis} \sqsubseteq \mathsf{Inflammation} \sqcap \exists \mathsf{has\text{-}loc}.\mathsf{Endocardium}$	•	⋆
$\underline{\alpha_7}$	$\mathsf{Inflammation} \sqsubseteq \mathsf{Disease} \sqcap \exists \mathsf{acts\text{-}on}.\mathsf{Tissue}$	•	⋆
$\underline{\alpha_8}$	$\mathsf{Disease} \sqcap \exists \mathsf{has\text{-}loc}.\mathsf{Heart} \sqsubseteq \mathsf{HeartDisease}$	•	⋆
$\underline{\alpha_9}$	$\mathsf{part\text{-}of} \sqsubseteq \mathsf{has\text{-}loc}$	•	⋆
$\underline{\alpha_{10}}$	$\mathsf{Trans(has\text{-}loc)}$	•	⋆

Fig. 1. An example ontology $\mathcal{O}_{\mathsf{ex}}$; the minimal locality-based module $\mathcal{O}^{\mathsf{loc}}_{\mathsf{Endocarditis}}$; and the justifications for $\mathsf{Endocarditis} \sqsubseteq_{\mathcal{O}} \mathsf{HeartDisease}$.

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

$$\begin{aligned}\mathsf{Con}^{\bot}(\mathbf{S}) &::= A^{\bot} \mid (\neg C^{\top}) \mid (C \sqcap C^{\bot}) \mid (\exists r^{\bot}.C) \mid (\exists r.C^{\bot}) \\ &\quad\mid (\geq n\, r^{\bot}.C) \mid (\geq n\, r.C^{\bot}) \\ \mathsf{Con}^{\top}(\mathbf{S}) &::= (\neg C^{\bot}) \mid (C_1^{\top} \sqcap C_2^{\top})\end{aligned}$$

where $A^{\bot} \notin \mathbf{S}$ *is a concept name,* C *is a* $\mathcal{SHOIQ}$*-concept,* $C^{\bot} \in \mathsf{Con}^{\bot}(\mathbf{S})$*,* $C_i^{\top} \in \mathsf{Con}^{\top}(\mathbf{S})$ *(for* $i = 1, 2$*), and* $\mathsf{Sig}(r^{\bot}) \not\subseteq \mathbf{S}$*.*

An axiom α *is* syntactically local w.r.t. $\mathbf{S}$ *if it is of one of the following forms: (i)* $r^{\bot} \sqsubseteq r$*, (ii)* $\mathsf{Trans}(\mathsf{r}^{\bot})$*, (iii)* $C^{\bot} \sqsubseteq C$ *or (iv)* $C \sqsubseteq C^{\top}$*. The set of all* $\mathcal{SHOIQ}$*-axioms that are syntactically local w.r.t.* $\mathbf{S}$ *is denoted by* $\mathsf{s\_local}(\mathbf{S})$*. A* $\mathcal{SHOIQ}$*-ontology* $\mathcal{O}$ *is* syntactically local w.r.t. $\mathbf{S}$ *if* $\mathcal{O} \subseteq \mathsf{s\_local}(\mathbf{S})$.

Intuitively, if an axiom α is syntactically local w.r.t. $\mathbf{S}$, its interpretation is *directly affected* by that of symbols in $\mathbf{S}$, in the sense that α is true in every interpretation $\mathcal{I}$ in which concept and role names from $\mathbf{S}$ are interpreted with the empty set. Based on this notion, locality-based modules can be defined as follows: Let $\mathcal{O}$ be a $\mathcal{SHOIQ}$ ontology, $\mathcal{O}' \subseteq \mathcal{O}$ a subset of it, and $\mathbf{S}$ a signature. Then, $\mathcal{O}'$ is a *locality-based module for* $\mathbf{S}$ *in* $\mathcal{O}$ if every axiom $\alpha \in \mathcal{O} \backslash \mathcal{O}'$ is syntactically local w.r.t. $\mathbf{S} \cup \mathsf{Sig}(\mathcal{O}')$. Given an ontology $\mathcal{O}$ and a signature $\mathbf{S}$, there always exists a unique, minimal locality-based module [4], denoted by $\mathcal{O}^{\mathsf{loc}}_{\mathbf{S}}$. In the example ontology, it can be easily verified that the underlined axioms are precisely those in $\mathcal{O}^{\mathsf{loc}}_{\{\mathsf{Endocarditis}\}}$.

The notion of strong subsumption module (first introduced in [3]) is essential for our modularization-based approach.

Definition 3 (Strong subsumption module). *Let $\mathcal{S} \subseteq \mathcal{O}$ be $\mathcal{SHOIQ}$ ontologies, and A a concept name. Then, $\mathcal{S}$ is a* subsumption module for A in $\mathcal{O}$ *if, for all $B \in \mathsf{CN}(\mathcal{O})$: $A \sqsubseteq_{\mathcal{O}} B$ iff $A \sqsubseteq_{\mathcal{S}} B$.*

A subsumption module $\mathcal{S}$ for A in $\mathcal{O}$ is called strong *if, for all $B \in \mathsf{CN}(\mathcal{O})$: $A \sqsubseteq_{\mathcal{O}} B$ implies that $J \subseteq \mathcal{S}$, for every justification J for $A \sqsubseteq B$ in $\mathcal{O}$.*

Observe that the *largest* such strong subsumption module is the whole ontology itself, and the *smallest* such module is precisely the union of all justifications J for $A \sqsubseteq B$ in $\mathcal{O}$, for all superconcept B of A. For our purpose, the minimal locality-based module is of interest since it is relative small (though not smallest) and cheap to compute (i.e., quadratic time).

Selection functions

We introduce the notion of selection function in a single ontology given in [8], which will be used in our algorithm to extract a subset of an ontology relevant to a subsumption to some degree. Though applied to arbitrary DL languages, we here restrict attention to $\mathcal{SHOIQ}$:

Definition 4 (Selection function). *Let $\mathcal{L}$ be the set of all $\mathcal{SHOIQ}$ axioms over a set of signature. Then, a selection function for $\mathcal{L}$ is a mapping $s_{\mathcal{L}} : \mathcal{P}(\mathcal{L}) \times \mathcal{L} \times \mathbb{N} \to \mathcal{P}(\mathcal{L})$ s.t. $s_{\mathcal{L}}(\mathcal{O}, \alpha, k) \subseteq \mathcal{O}$, where $\mathcal{P}(\mathcal{L})$ is the power set of $\mathcal{L}$.*

Intuitively, a selection function selects a subset of an ontology w.r.t. an axiom at step k. A specific selection function based on *syntactic relevance* is employed in our algorithm. We begin with defining *direct relevance* between two axioms.

Definition 5 (Direct relevance). *Two axioms α and β are* directly relevant *iff $\mathsf{Sig}(\alpha) \cap \mathsf{Sig}(\beta) \neq \emptyset$.*

The intuition is that two axioms are directly relevant if they share a common (concept or role) name. Another relevance relation is given in [15]. However, that relevance relation is tailored for *unfoldable* DL $\mathcal{ALC}$, and as such the selection function defined by it cannot be used to find all justifications in our setting, so we do not consider it here.

Based on the notion of direct relevance, we can define the notion of relevance between an axiom and an ontology.

Definition 6. *An axiom α is relevant to an ontology $\mathcal{O}$ iff there exists an axiom β in $\mathcal{O}$ such that α and β are directly relevant.*

We introduce the relevance-based selection function which can be used to find all the axioms in an ontology that are relevant to an axiom to some degree.

Definition 7 (Relevance-based selection function). *Let $\mathcal{O}$ be an ontology, α be an axiom and k be an integer. The* relevance-based selection function, *written s_{rel}, is defined inductively as follows:*
$s_{rel}(\mathcal{O}, \alpha, 0) = \emptyset$
$s_{rel}(\mathcal{O}, \alpha, 1) = \{\beta \in \mathcal{O} : \alpha \text{ and } \beta \text{ are directly relevant}\}$
$s_{rel}(\mathcal{O}, \alpha, k) = \{\beta \in \mathcal{O} : \beta \text{ is directly relevant to } s_{rel}(\mathcal{O}, \alpha, k-1)\}$*, where* $k > 1$.

We call $s_{rel}(\mathcal{O}, \alpha, k)$ the k-relevant subset of $\mathcal{O}$ w.r.t. α. For convenience, we define $s_k(\mathcal{O}, \alpha) = s_{rel}(\mathcal{O}, \alpha, k) \setminus s_{rel}(\mathcal{O}, \alpha, k-1)$ for $k \geq 1$.

Hitting set tree (HST) algorithm

We briefly introduce some notions regarding Reiter's Hitting Set Tree algorithm given in [13] which will be used in our algorithm to find all justifications. We follow the reformulated notions in Reiter's theory in [10]. Given a *universal set* U, and a set $S = \{s_1, ..., s_n\}$ of subsets of U which are *conflict sets*, i.e. subsets of the system components responsible for the error. A *hitting set* T for S is a subset of U such that $s_i \cap T \neq \emptyset$ for all $1 \leq i \leq n$. A *minimal hitting set* T for S is a hitting set such that no $T' \subset T$ is a hitting set for S. A hitting set T is cardinality-minimal if there is no other hitting set T' such that $|T'| < |T|$. Reiter's algorithm is used to calculate minimal hitting sets for a collection $S = \{s_1, ..., s_n\}$ of sets by constructing a labeled tree, called a Hitting Set Tree (HST). In a HST, each node is labeled with a set $s_i \in S$, and each edge is labeled with an element in $\cup_{s_i \in S} s_i$. For each node n in a HST, let $H(n)$ be the set of edge labels on the path from the root of the HST to n. Then the label for n is any set $s \in S$ such that $s \cap H(n) = \emptyset$, if such a set exists. Suppose s is the label of a node n, then for each $\sigma \in s$, n has a successor n_σ connected to n by an edge with σ in its label. If the label of n is the empty set, then we have that $H(n)$ is a hitting set of S. In the case of finding justifications, the universal set corresponds to the ontology and a conflict set corresponds to a justification [10].

3 Justification Coverage in Locality-based Modules

This section presents the main technical contribution of the paper that lays the foundation of our modularization-based algorithm. We show that a locality-based module for $\mathbf{S} = \{A\}$ in $\mathcal{O}$ is a strong subsumption module for A in $\mathcal{O}$.

Proposition 1. *Let* $\mathbf{S}$ *be a signature, and* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ *an interpretation such that* $x^\mathcal{I} = \emptyset$ *for all (concept and role) names* $x \notin \mathbf{S}$*. Then,* $(C^\perp)^\mathcal{I} = \emptyset$ *for every concept* $C^\perp \in \mathsf{Con}^\perp(\mathbf{S})$*, and* $(C^\top)^\mathcal{I} = \Delta^\mathcal{I}$ *for every concept* $C^\top \in \mathsf{Con}^\top(\mathbf{S})$*.*

The proof is an easy induction on the structure of the concepts $C^\perp$ and $C^\top$. Intuitively, every concept in $\mathsf{Con}^\top(\mathbf{S})$ ($\mathsf{Con}^\perp(\mathbf{S})$, resp.) behaves as if it were the top concept (the bottom concept, resp.) in any interpretation $\mathcal{I}$ with $x^\mathcal{I} = \emptyset$ for all $x \notin \mathbf{S}$. It follows that syntactically local axioms of the form $C^\perp \sqsubseteq C$ and $C \sqsubseteq C^\top$ are vacuously satisfied by such an interpretation $\mathcal{I}$. This property of syntactically local axioms is used to prove the following lemma.

Lemma 1. *Let* $\mathcal{O}$ *be a* $\mathcal{SHOIQ}$ *ontology,* A, B *concept names in* $\mathsf{Sig}(\mathcal{O})$ *such that* $A \sqsubseteq_\mathcal{O} B$*,* $\mathcal{O}^{\mathsf{loc}}_A$ *a locality-based module for* $\{A\}$ *in* $\mathcal{O}$*. If* $A \sqsubseteq_\mathcal{S} B$ *for an* $\mathcal{S} \subseteq \mathcal{O}$ *such that* $\mathcal{S} \nsubseteq \mathcal{O}^{\mathsf{loc}}_A$*, then* $A \sqsubseteq_{\mathcal{S}'} B$ *with* $\mathcal{S}' = \mathcal{S} \cap \mathcal{O}^{\mathsf{loc}}_A$*.*

Proof. We show the contraposition by assuming that $A \not\sqsubseteq_{\mathcal{S}'} B$ and then demonstrating that $A \not\sqsubseteq_\mathcal{S} B$. Since $A \not\sqsubseteq_{\mathcal{S}'} B$, there must be a model $\mathcal{I}'$ of $\mathcal{S}'$ and an individual $w \in \Delta^{\mathcal{I}'}$ such that $w \in A^{\mathcal{I}'} \backslash B^{\mathcal{I}'}$. Construct a new interpretation $\mathcal{I}$ based on $\mathcal{I}'$ by setting $x^\mathcal{I} := \emptyset$ for all symbols (role or concept names) $x \in \mathsf{Sig}(\mathcal{O}) \backslash \mathsf{Sig}(\mathcal{O}^{\mathsf{loc}}_A)$. Obviously, $w \in A^\mathcal{I}$ since $\mathcal{I}$ does not change the interpretation of $A \in \mathsf{Sig}(\mathcal{O}^{\mathsf{loc}}_A)$. There are two possibilities for B: either $B^\mathcal{I} = B^{\mathcal{I}'}$ or $B^\mathcal{I} = \emptyset$. In either case, we have that $w \notin B^\mathcal{I}$.

It remains to show that $\mathcal{I}$ is a model of $\mathcal{S}$, i.e., satisfies every axiom $\alpha = (\alpha_L \sqsubseteq \alpha_R)$ in $\mathcal{S}$. We make a case distinction as follows:

- $\alpha \in \mathcal{O}_A^{\mathsf{loc}}$. It follows that $\alpha \in \mathcal{S}'$, and thus $\mathcal{I}' \models \alpha$. By construction, both $\mathcal{I}$ and $\mathcal{I}'$ agree on the interpretation of symbols in $\mathsf{Sig}(\mathcal{O}_A^{\mathsf{loc}})$ and thus $\mathsf{Sig}(\alpha)$. Hence, $\mathcal{I} \models \alpha$ as required.
- $\alpha \notin \mathcal{O}_A^{\mathsf{loc}}$. By definition of locality-based modules, α is syntactically local w.r.t. $\mathbf{S} = \mathsf{Sig}(\mathcal{O}_A^{\mathsf{loc}}) \cup \{A\}$. Then, there are four possibilities for α:
  - $\alpha = r^\perp \sqsubseteq r$. First, assume that $r^\perp$ is a role name. Then, $r^\perp \notin \mathbf{S}$ and thus $r^\perp \in \mathsf{Sig}(\mathcal{O}) \backslash \mathsf{Sig}(\mathcal{O}_A^{\mathsf{loc}})$. By construction of $\mathcal{I}$, $(r^\perp)^\mathcal{I} = \emptyset$. Otherwise, $r^\perp$ is an inverse role s^-. Then, $s \in \mathsf{Sig}(r^\perp) \not\subseteq \mathbf{S}$. It follows that $s \in \mathsf{Sig}(\mathcal{O}) \backslash \mathsf{Sig}(\mathcal{O}_A^{\mathsf{loc}})$, and thus $(r^\perp)^\mathcal{I} = s^\mathcal{I} = \emptyset$. In both cases, $\mathcal{I} \models \alpha$ as required.
  - $\alpha = \mathsf{Trans}(r^\perp)$. Analogous to the first case.
  - $\alpha = C^\perp \sqsubseteq C$. By Proposition 1, $(C^\perp)^\mathcal{I} = \emptyset$. Hence, $\mathcal{I} \models \alpha$.
  - $\alpha = C \sqsubseteq C^\top$. By Proposition 1, $(C^\top)^\mathcal{I} = \Delta^\mathcal{I}$. Hence, $\mathcal{I} \models \alpha$.

Since $\mathcal{I}$ is a model of $\mathcal{S}$ such that $w \in A^\mathcal{I} \backslash B^\mathcal{I}$, we have $A \not\sqsubseteq_\mathcal{S} B$, contradicting the premise of the lemma. ❏

Now, we are ready to establish the required property of the modules:

Theorem 1 ($\mathcal{O}_A^{\mathsf{loc}}$ is a strong subsumption module). *Let $\mathcal{O}$ be a $\mathcal{SHOIQ}$ ontology and A a concept name. Then $\mathcal{O}_A^{\mathsf{loc}}$ is a strong subsumption module for A in $\mathcal{O}$.*

Proof. The fact that $\mathcal{O}_A^{\mathsf{loc}}$ is a subsumption module has been shown in [4]. It remains to show that it is strong, i.e., every justification $J \subseteq \mathcal{O}$ for $A \sqsubseteq_\mathcal{O} B$ is contained in $\mathcal{O}_A^{\mathsf{loc}}$, for every concept name $B \in \mathsf{CN}(\mathcal{O})$

Assume to the contrary that there is a concept name B and a justification J for $A \sqsubseteq_\mathcal{O} B$ that is not contained in $\mathcal{O}_A^{\mathsf{loc}}$. By Lemma 1, the strict subset $J' = J \cap \mathcal{O}_A^{\mathsf{loc}}$ of J is such that $A \sqsubseteq_{J'} B$. Obviously, J is not minimal and hence cannot be a justification for $A \sqsubseteq_\mathcal{O} B$, contradicting the initial assumption. ❏

Intuitively, the (minimal) locality-based module for $\mathbf{S} = \{A\}$ in a $\mathcal{SHOIQ}$-ontology $\mathcal{O}$ contains *all* the relevant axioms for any subsumption $\sigma = (A \sqsubseteq_\mathcal{O} B)$, in the sense that all responsible axioms for σ are included. In other words, in order to find all justifications for a certain entailment in an OWL ontology, it is sufficient to consider only axioms in the locality-based module. Since the *minimal* locality-based modules are relatively very small (see, e.g., [6, 16]), our modularization-based approach proves promising. The empirical results on real-life ontologies are described in Section 5.

4 Our Modularization-based Algorithm

In this section, we propose a new algorithm for finding all justifications based on the relevance-based algorithm and the modularization extraction algorithm. Before we describe our algorithm, we need to recap the relevance-based algorithm given in [9].

```
Algorithm 1: REL_ALL_JUSTS($A \sqsubseteq B, \mathcal{O}, s$)
   Data: An ontology $\mathcal{O}$, a subsumption $A \sqsubseteq B$ and a selection function $s$.
   Result: All justifications $\mathcal{J}$
1  begin
2      Globals : $\mathcal{J} \leftarrow \emptyset$;
3      $\mathcal{O}' \leftarrow HS \leftarrow HS_{local} \leftarrow \emptyset$; $k \leftarrow 1$;
4      $\mathcal{S} \leftarrow s(\mathcal{O}, A \sqsubseteq B, k)$;
5      while $\mathcal{S} \neq \emptyset$ do
6          $\mathcal{O}' \leftarrow \mathcal{O}' \cup \mathcal{S}$;
7          if $HS_{local} \neq \emptyset$ then
8              for $P \in HS_{local}$ do            /* Get global hitting sets */
9                  if $\mathcal{O} \setminus P \not\models A \sqsubseteq B$ then
10                     $HS \leftarrow HS \cup \{P\}$;
11             $HS_{local} \leftarrow HS_{local} \setminus HS$;
12             if $(HS_{local} = \emptyset)$ then
13                 return $\mathcal{J}$                     /* Early termination */;
14             $HS_{temp} \leftarrow HS_{local}$;
15             for $P \in HS_{temp}$ do             /* Expand hitting set tree */
16                 $(\mathcal{J}', HS'_{local}) \leftarrow$ EXPAND_HST($A \sqsubseteq B, \mathcal{O}' \setminus P$);
17                 $\mathcal{J} \leftarrow \mathcal{J} \cup \mathcal{J}'$;
18                 $HS_{local} \leftarrow HS_{local} \cup \{P \cup P' | P' \in HS'_{local}\} \setminus \{P\}$;
19         else if $\mathcal{O}' \models A \sqsubseteq B$ then
20             $(\mathcal{J}, HS_{local}) \leftarrow$ EXPAND_HST($A \sqsubseteq B, \mathcal{O}'$);
21         $k \leftarrow k + 1$;
22         $\mathcal{S} \leftarrow s_k(\mathcal{O}, A \sqsubseteq B)$;
23     return $\mathcal{J}$
24 end
```

The relevance-based algorithm (Algorithm 1) receives an ontology $\mathcal{O}$, a subsumption $A \sqsubseteq B$ of $\mathcal{O}$ and a selection function s, and outputs the set of all justifications $\mathcal{J}$. We sketch the basic idea of the algorithm and refer to [9] for details of the algorithm. First of all, we find the first k such that $A \sqsubseteq B$ is inferred by the k-relevant subset $\mathcal{O}'$ of $\mathcal{O}$, i.e., the "if" condition in line 19 is satisfied. We then call Algorithm 2 to find a set of justifications for $A \sqsubseteq B$ in $\mathcal{O}'$ and a set of *local hitting sets*, where a local hitting set is a hitting set for all justifications in the selected sub-ontology, i.e., O' in line 20. We then add to the sub-ontology obtained in the previous iteration those axioms that are directly relevant this sub-ontology. For those local hitting sets that are not hitting sets of all justifications in the entire ontology $\mathcal{O}$, we call Algorithm 2 to further expand them, and so on.

To compute a single justification in Algorithm 2, we invoke a sub-procedure SINGLE_JUST($\sigma, \mathcal{O}$) which is a black-box pinpointing algorithm optimized either by the sliding window technique in [10] or by binary search technique in [3].

The correctness of Algorithm 1 follows from Theorem 1 in [9].

Algorithm 2: EXPAND_HST($A \sqsubseteq B, \mathcal{O}$)

```
Data: An ontology $\mathcal{O}$ and a subsumption $A \sqsubseteq B$ of $\mathcal{O}$
Result: A set of justifications $\mathcal{J}$ for $A \sqsubseteq B$ in $\mathcal{O}$ and a set of hitting sets
begin
    $HS \leftarrow HS_1 \leftarrow \emptyset$
    $J \leftarrow$ SINGLE_JUST($A \sqsubseteq B, \mathcal{O}$)
    $\mathcal{J} \leftarrow \mathcal{J} \cup \{J\}$
    for $\alpha \in J$ do                              /* Create all possible branches. */
        $HS_1 \leftarrow HS_1 \cup \{\{\alpha\}\}$
    while true do
        $HS_2 \leftarrow \emptyset$
        for ($P \in HS_1$) do
            if $\mathcal{O} \setminus P \not\models A \sqsubseteq B$ then
                $HS \leftarrow HS \cup \{P\}$
            else
                $HS_2 \leftarrow HS_2 \cup \{P\}$        /* Branches need to be expanded */
        if ($HS_1 = \emptyset$) or ($HS_2 = \emptyset$) then
            return ($\mathcal{J}, HS$)
        $HS_1 \leftarrow \emptyset$
        for $P \in HS_2$ do
            $J \leftarrow$ SINGLE_JUST($A \sqsubseteq B, \mathcal{O} \setminus P$)
            $\mathcal{J} \leftarrow \mathcal{J} \cup \{J\}$
            for $\alpha \in J$ do
                $HS_1 \leftarrow HS_1 \cup \{P \cup \{\alpha\}\}$
end
```

Theorem 2. *Given an ontology $\mathcal{O}$, a subsumption $A \sqsubseteq B$ of $\mathcal{O}$ and a relevance-based selection function s_{rel}, $\mathcal{J}$ returned by Algorithm 1 is the set of all justifications for $A \sqsubseteq B$.*

Based on the algorithms introduced above, we propose our novel algorithm for computing all the justification. The idea of our algorithm is straightforward: to find all justifications for a subsumption $A \sqsubseteq B$ in $\mathcal{O}$, we first extract the locality-based module $\mathcal{O}_A^{\mathsf{loc}}$ for $\mathbf{S} = \{A\}$ in $\mathcal{O}$ and then apply Algorithm 1. The method is outlined in Algorithm 3, where EXTRACT_MODULE implements the locality-based extraction algorithm in [4], and s_{rel} is the relevance-based selection function. The correctness of the algorithm can be seen by Theorem 1 and Theorem 2. We illustrate the effectiveness of our algorithm by means of an example:

Example 1. Consider an ontology $\mathcal{O}$ that contains the following axioms:

$$\alpha_{1i} : A_{1i} \sqsubseteq P_{1i} \sqcap Q_{1i} \sqcap Z, \quad \alpha_{2i} : P_{1i} \sqsubseteq A_{2i} \sqcap Z, \quad \alpha_{3i} : Q_{1i} \sqsubseteq A_{2i} \sqcap Z$$
$$\alpha_{4i} : A_{2i} \sqsubseteq P_{2i} \sqcap Q_{2i} \sqcap Z, \quad \alpha_{5i} : P_{2i} \sqsubseteq A_{3i} \sqcap Z, \quad \alpha_{6i} : Q_{2i} \sqsubseteq A_{3i} \sqcap Z,$$

Algorithm 3: MODULE_ALL_JUSTS($A \sqsubseteq B, \mathcal{O}$)

```
Data: An ontology $\mathcal{O}$ and a subsumption $A \sqsubseteq B$
Result: All justifications $\mathcal{J}$
begin
    $\mathcal{O}^{\mathsf{loc}}_A \leftarrow$ EXTRACT_MODULE($\mathcal{O}, A$)
    return REL_ALL_JUSTS($A \sqsubseteq B, \mathcal{O}^{\mathsf{loc}}_A, s_{rel}$)
end
```

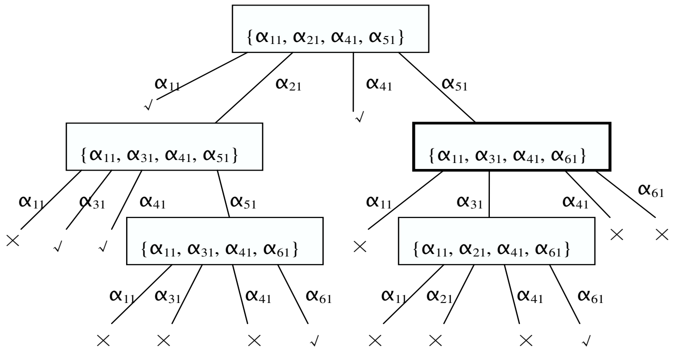


Fig. 2. Finding all justifications by HST algorithm on the locality-based module. Each rectangle represents a justification, and the bold rectangle indicates a justification reuse. '×' means early path termination, while '$\surd$' means a hitting set is found.

for $1 \leq i \leq 10000$. Obviously, $\mathcal{O}$ comprises 60 000 axioms and entails the subsumption $\sigma = (A_{11} \sqsubseteq A_{31})$. While such an ontology clearly is not a realistic ontology, it well demonstrates the need and potential of search space reduction. If algorithm REL_ALL_JUSTS is applied directly to this ontology, one cannot expect an acceptable performance when finding all justifications. This is because: (i) SINGLE_JUST($\sigma, \mathcal{O}$) has to prune a very large set, and (ii) each subsumption test is w.r.t. the entire ontology $\mathcal{O}$ since all the axioms $\mathcal{O}$ share a common concept Z. In our modularization-based approach, however, we first extract the locality-based module $\mathcal{O}^{\mathsf{loc}}_{A_{11}}$ for $\mathbf{S} = \{A_{11}\}$ in $\mathcal{O}$, and then apply REL_ALL_JUSTS to $\mathcal{O}^{\mathsf{loc}}_{A_{11}}$ instead of $\mathcal{O}$. Since the module contains only 6 axioms, i.e., $\mathcal{O}^{\mathsf{loc}}_{A_{11}} = \{\alpha_{11}, \alpha_{21}, \alpha_{31}, \alpha_{41}, \alpha_{51}, \alpha_{61}\}$, both points above can be achieved in much less time.

Figure 2 illustrates the process of finding all justifications by means of expanding a hitting set tree (HST). To begin with, a justification $\{\alpha_{11}, \alpha_{21}, \alpha_{41}, \alpha_{51}\}$ is computed by SINGLE_JUST($\sigma, \mathcal{O}^{\mathsf{loc}}_{A_{11}}$), which is taken as the root of the tree. Since $\mathcal{O}^{\mathsf{loc}}_{A_{11}}$ dispensed with α_{11} does not entail σ, $\{\alpha_{11}\}$ is a hitting set. On the other hand, $\mathcal{O}' = \mathcal{O}^{\mathsf{loc}}_{A_{11}} \backslash \{\alpha_{21}\}$ still entails σ, and thus another justification can be computed by calling SINGLE_JUST($\sigma, \mathcal{O}'$). The process continues to expand HST until it finds all other justifications for σ: $\{\alpha_{11}, \alpha_{31}, \alpha_{41}, \alpha_{51}\}$, $\{\alpha_{11}, \alpha_{31}, \alpha_{41}, \alpha_{61}\}$, $\{\alpha_{11}, \alpha_{21}, \alpha_{41}, \alpha_{61}\}$. Observe that the node following the branch $\{\alpha_{51}\}$ is a result of the optimization 'justification reuse.'

Ontologies	♯Axioms	♯Concepts	♯Roles	Module size		Extraction time
				Average	Maximum	(sec)
Galen	4 529	2 748	413	75	530	6
Go	28 897	20 465	1	16	125	40
Nci	46 940	27 652	70	29	436	65

Table 2. Benchmark ontologies and their characteristics.

5 Empirical Results

Our algorithm has been realized by using KAON2[4] as the black-box reasoner. Of course, the method (like other black-box approaches) can be applied to any other reasoner, e.g., RacerPro[5] and FaCT++[6]. To fairly compare with the pinpointing algorithm in [10], we re-implemented it with KAON2 API (henceforth referred to as ALL_JUSTS algorithm). The experiments have been performed on a Linux server with an Intel(R) CPU Xeon(TM) 3.2GHz running Sun's Java 1.5.0 with allotted 2GB heap space.

Benchmark ontologies used in our experiments are the Galen Medical Knowledge Base[7], the Gene Ontology (Go)[8] and the US National Cancer Institute thesaurus (Nci)[9]. The three biomedical ontologies are well-known to both the life science and Semantic Web communities since they are employed in real-world applications and often used as benchmarks for testing DL reasoners. Both Go and Nci are formulated in the lightweight DL $\mathcal{EL}$, while Galen uses expressivity of the more complex DL $\mathcal{SHF}$. Some information concerning the size and characteristics of the benchmark ontologies are given in the left part of Table 2.

Modularization reveals structures and dependencies of concepts in the ontologies as argued in [4, 16]. We extract the (minimal) locality-based module for $\mathbf{S} = \{A\}$ in $\mathcal{O}$, for every benchmark ontology $\mathcal{O}$ and each concept name $A \in \mathsf{CN}(\mathcal{O})$. The size of the modules and the time required to extract them are shown in the last three columns of Table 2. Observe that the modules in Galen are larger than those in the other two ontologies although the ontology itself is smaller. This suggests that Galen is more complex in the sense that more axioms in it are non-local (thus relevant) according to Definition 2.

In the experiments, we consider three concept names in $\mathsf{CN}(\mathcal{O})$ for each benchmark ontology $\mathcal{O}$ such that one of them has the largest locality-based module[10]. For the sake of brevity, we denote by $\mathsf{subs}(\mathcal{O})$ the set of all tested subsumptions $A \sqsubseteq B$ in $\mathcal{O}$, with A one of the three concept names mentioned above and B an

[4] http://kaon2.semanticweb.org/

[5] http://www.racer-systems.com/

[6] http://owl.man.ac.uk/factplusplus/

[7] http://www.openclinical.org/prj_galen.html

[8] http://www.geneontology.org

[9] http://www.mindswap.org/2003/CancerOntology/nciOntology.owl

[10] The concept name with largest module is hand-picked in order to cover hard cases in our experiments, while the other two are randomly selected.

inferred subsumer of A. For each $\mathcal{O}$ of our benchmark ontologies, we compute *all* justifications for σ in $\mathcal{O}$, where $\sigma \in \mathsf{subs}(\mathcal{O})$. In order to compare with the other existing approaches, we perform the following for each σ and $\mathcal{O}$ to compute all justifications:

1. $\mathsf{ALL\_JUSTS}(\sigma, \mathcal{O})$ (i.e., the algorithm in [10]).
2. $\mathsf{REL\_ALL\_JUSTS}(\sigma, \mathcal{O}, s_{rel})$;
3. $\mathsf{MODULE\_ALL\_JUSTS}(\sigma, \mathcal{O})$;

The justification results by MODULE_ALL_JUSTS are shown in Table 3, where the ontology marked with $\star$ means that some run does not terminate within the two hour time-out. Precisely, there are three subsumptions in Go and one in Nci, for which the computation took more than two hours. The statistics given on the right hand side of the table does not take into account these subsumptions.

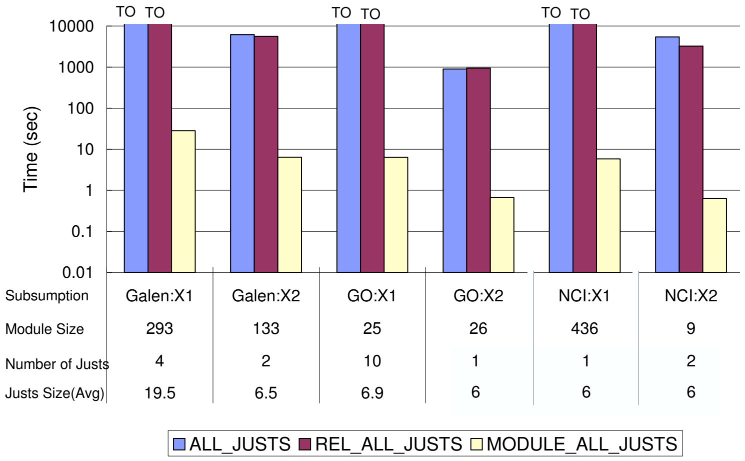
Ontologies	♯Subsumptions	♯Justifications		Justification size	
	$\lvert\mathsf{subs}(\mathcal{O})\rvert$	Average	Maximum	Average	Maximum
Galen	69	1.5	4	9.7	24
Go*	53	3.2	11	5.3	9
Nci*	23	1.6	8	5.4	9

Table 3. Justification results using the modularization-based approach.

To visualize the time performances of the three algorithms, we randomly selected two subsumptions σ_1 and σ_2 from $\mathsf{subs}(\mathcal{O})$ for each ontology $\mathcal{O}$ and compared their computation time required by the three algorithms. These subsumptions are shown as follows:

Galen:σ_1	AcuteErosionOfStomach $\sqsubseteq$ GastricPathology
Galen:σ_2	AppendicularArtery $\sqsubseteq$ PhysicalStructure
Go:σ_1	GO_0000024 $\sqsubseteq$ GO_0007582
Go:σ_2	GO_0000027 $\sqsubseteq$ GO_0044238
Nci:σ_1	CD97_Antigen $\sqsubseteq$ Protein
Nci:σ_2	APC_8024 $\sqsubseteq$ Drugs_and_Chemicals

The chart in Figure 3 depicts the overall computation time required for each algorithm to find all justifications for each tested subsumption. Unlike the time results reported in [10], which excluded the time for satisfiability checking, we report here the overall computation time, i.e. the total time of the algorithm including the time needed by the black-box reasoner for the standard reasoning tasks. Observe that both ALL_JUSTS and REL_ALL_JUSTS did not yield results within the time-out of two hours on three out of six tested subsumptions (marked by "TO" on the chart). Comparing these two algorithms (without modularization), REL_ALL_JUSTS performs noticeably better than ALL_JUSTS in most cases. For instance, on the subsumptions Galen:σ_2 and Nci:σ_2, REL_ALL_JUSTS outperforms ALL_JUSTS by about 10 and 20 minutes, respectively. On the subsumption Go:σ_2, both algorithms show a similar performance, i.e., time difference is less



Subsumption	Galen:X1	Galen:X2	GO:X1	GO:X2	NCI:X1	NCI:X2
Module Size	293	133	25	26	436	9
Number of Justs	4	2	10	1	1	2
Justs Size(Avg)	19.5	6.5	6.9	6	6	6

Fig. 3. The time performance of three algorithms for finding all justifications.

than a minute. More explanations on the comparison between these two algorithms can be found in [9].

Interestingly, MODULE_ALL_JUSTS outperforms all the other algorithms on all subsumptions, and the improvement is tremendous as can be seen in all cases in the chart. This empirically confirms our initial conjecture that, given the strongness property (in the sense of Definition 3) and the small size (see Table 2 and [6, 16]) of locality-based modules, our optimization should be highly effective. As an example, MODULE_ALL_JUSTS took *only* 0.6 seconds to find all the justifications for NCI:σ_2, while REL_ALL_JUSTS needed 3 242 seconds. In this case, the locality-based module for APC_8024 in NCI consists of 9 axioms, whereas the whole ontology has some tens of thousands of axioms. Although the selection function used in REL_ALL_JUSTS also prunes the search space by considering only "k-directly relevant" axioms (see Definition 7) when HST algorithm is executed, several irrelevant axioms (in the sense of syntactic locality) are still considered.

6 Conclusion

In this paper, we proposed a novel approach for finding all justifications for an entailment in OWL DL. The approach is based on the computation of minimal locality-based modules. We first showed that locality-based modules always cover all axioms in all justifications and exploited this property to limit the search space when finding all justifications. Then, we presented a modularization-based pinpointing algorithm that is based on relevance-based techniques and a hitting set tree algorithm. Finally, we reported on several promising empirical results that demonstrate an improvement of several orders of magnitude in efficiency and scalability of finding all justifications in OWL DL ontologies. Our work is based on locality-based modules. As future work, we shall investigate different kinds of modules and selection functions that hopefully produce even more relevant axioms for pinpointing.

Acknowledgements: This work was partially supported by the DFG project under grant BA1122/11-1 and the EU under the IST project NeOn (IST-2006-027595) http://www.neon-project.org.

References

1. Franz Baader and Rafael Peñaloza. Axiom pinpointing in general tableaux. In *Proc. of TABLEAUX'07*, pages 11–27, 2007.
2. Franz Baader, Rafael Peñaloza, and Boontawee Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}^+$. In *Proc. of KI'07*, pages 52–67, 2007.
3. Franz Baader and Boontawee Suntisrivaraporn. Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}^+$. In *Proceedings of KR-MED'08: Representing and Sharing Knowledge Using SNOMED*, 2008.
4. Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. Modular reuse of ontologies: Theory and practice. *J. of Artificial Intelligence Research (JAIR)*, 31:273–318, 2008.
5. B. Cuenca Grau, C. Halaschek-Wiener, and Y. Kazakov. History matters: Incremental ontology reasoning using modules. In *Proceedings of ISWC*, Busan, South Korea, 2007. Springer.
6. B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Just the right amount: Extracting modules from ontologies. In *Proc. of WWW'07*, pages 717–726, Banff, Canada, 2007. ACM.
7. Ian Horrocks and Ulrike Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proc. of IJCAI'05*, pages 448–453, 2005.
8. Zhisheng Huang, Frank van Harmelen, and Annette ten Teije. Reasoning with inconsistent ontologies. In *Proc. of IJCAI'05*, pages 254–259. 2005.
9. Qiu Ji, Guilin Qi, and Peter Haase. A relevance-based algorithm for finding justifications of DL entailments. In *Technical report, University of Karlsruhe*, http://www.aifb.uni-karlsruhe.de/WBS/gqi/papers/RelAlg.pdf, 2008.
10. Aditya Kalyanpur, Bijan Parsia, Matthew Horridge, and Evren Sirin. Finding all justifications of OWL DL entailments. In *Proc. of ISWC/ASWC'07*, pages 267–280, 2007.
11. Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and James Hendler. Debugging unsatisfiable classes in OWL ontologies. *Journal of Web Semantics*, 3(4):268–293, 2005.
12. Thomas Meyer, Kevin Lee, and Richard Booth. Knowledge integration for description logics. In *Proc. of AAAI'05*, pages 645–650. AAAI Press, 2005.
13. Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.
14. Stefan Schlobach and Ronald Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proc. of IJCAI'03*, pages 355–362. 2003.
15. Stefan Schlobach, Zhisheng Huang, Ronald Cornet, and Frank van Harmelen. Debugging incoherent terminologies. *J. Autom. Reasoning*, 39(3):317–349, 2007.
16. Boontawee Suntisrivaraporn. Module extraction and incremental classification: A pragmatic approach for $\mathcal{EL}^+$ ontologies. In *Proc. of ESWC'08*, pages 230–244, 2008.