RESEARCH ARTICLE

ScienceAsia 39 (2013): **79–90**

doi: 10.2306/scienceasia1513-1874.2013.39.079

Finding all justifications in SNOMED CT

Boontawee Suntisrivaraporn

School of Information, Computer and Communication Technology,
Sirindhorn International Institute of Technology, Thammasat University, Thailand

e-mail: sun@siit.tu.ac.th

Received 24 May 2012
Accepted 16 Jan 2013

ABSTRACT: SNOMED CT is a large-scale medical ontology which is developed using a variant of the inexpressive Description Logic $\mathcal{EL}$, a logic-based knowledge representation formalism and a logical underpinning of OWL 2 EL Profile. Description Logic reasoning can be used to compute subsumption relationships between SNOMED CT concepts and to pinpoint the reasons why a certain subsumption holds by finding justifications (sets of axioms responsible for this relationship). This helps the ontology developers to understand such a relationship and to debug it if needed. This article describes an extension of the method of finding one justification to one that finds all justifications for a given subsumption, introduces a SNOMED CT-specific optimization, and presents empirical evaluation results. Our extensive experiments on SNOMED CT show that (i) the proposed modularization-based approach makes it practicable to find all justifications in most cases in SNOMED CT; (ii) the first ten justifications, if any, for a subsumption can be computed in an acceptable runtime, and can be displayed in an incremental manner, i.e., the ontology developers may inspect the first justification while the reasoner continues to find more; and (iii) there is a high degree of commonality among justifications for a subsumption of interest.

KEYWORDS: description logic, OWL 2 EL, axiom pinpointing, automated reasoning

INTRODUCTION

The Systematized Nomenclature of Medicine, Clinical Terms (SNOMED CT) is a comprehensive clinical and medical ontology that covers a wide range of concepts in the domain, including anatomy, diseases, pharmaceutical products, clinical findings and medical procedures[1,2]. Resulting from merging SNOMED Reference Terminology[3] with Clinical Terms Version 3[4], SNOMED CT comprises almost four hundred thousand logical statements (henceforth, called axioms). The medical ontology has so far been modelled with the help of a small extension of the Description Logic $\mathcal{EL}$, for which automated reasoning of classification has proved useful in the generation of SNOMED CT in "normal form"[5] for distribution purposes.

Description Logics (DLs)[6] are a family of logic-based knowledge representation formalisms, which can be used to develop ontologies (i.e., knowledge bases) in a formally well-founded way. This is true both for expressive DLs, which are the logical basis of the Web Ontology Language OWL 2 (see http://www.w3.org/TR/owl2-overview/), and for lightweight DLs of the $\mathcal{EL}$ family[7], which are used in the design of large-scale medical ontologies such as SNOMED CT[1] and form one of the W3C-recommended tractable OWL profiles, OWL 2 EL[8]. One of the main advantages of employing a logic-based ontology language is that reasoning services can be used to derive implicit knowledge from the one explicitly represented. DL systems can, for example, classify a given ontology, i.e., compute all the subclass/superclass relationships between the concepts defined in the ontology and arrange these relationships as a hierarchical graph. The advantage of using a lightweight DL of the $\mathcal{EL}$ family is that classification is tractable, i.e., $\mathcal{EL}$ reasoners such as CEL[9] can compute the subclass hierarchy of a given ontology in polynomial time.

As with writing large programs, building large-scale ontologies is an error-prone try. Classification can help to alert the developer or user of an ontology to the existence of errors. A case in point for SNOMED CT is that the procedure concept 'amputation of finger' is an inferred subtype of 'amputation of hand.'[10,11] Another example has a procedure concept wrongly subsumed by a disease concept: 'Lichtenstein repair of inguinal hernia' is subsumed by 'inguinal hernia.'[12] Both examples are clearly unintended and can be uncovered by almost any DL classification reasoner. However, given an unintended subsumption relationship in a large ontology like SNOMED CT, it is not always easy to find the erroneous axioms responsible for it by hand. To overcome this problem, the DL community has thus far invested quite some work on automating this process. Given a subclass relationship or another questionable consequence, the aim is to find "a justification" or "all justifications" for that consequence, where a justification is a minimal subset of the ontology that has this consequence. Most of the work on axiom pinpointing in DLs was concerned with rather expressive DLs (see, e.g., Refs. 13–17). The first paper that concentrated on finding justifications in the $\mathcal{EL}$ family of DLs was Ref. 18. In addition to providing complexity results for finding justifications, it introduces a "pragmatic" algorithm for computing one justification, which is based on a modified version of the classification algorithm used by the CEL reasoner[9,19]. Although that approach worked quite well for mid-size ontologies (see the experiments on a variant of the GALEN medical ontology described in Ref. 18), it was not efficient enough to deal with large-scale ontologies like SNOMED CT. The first positive result on finding justifications in SNOMED CT was reported in Ref. 20, in which the authors proposed a highly efficient optimization technique based on the computation of a module of the ontology. However, the implemented algorithm in there could compute only "one" justification for each subclass relationship.

In this paper, we proposed an extension of that idea and design an algorithm that can effectively and efficiently compute "all" justifications for a given subclass relationship. Empirical results on a large number of representative random subclass relationships in SNOMED CT are reported which demonstrate that (i) the proposed modularization-based approach makes it practicable to find all justifications in most cases in SNOMED CT; (ii) the first ten justifications, if any, for a subsumption can be computed in an acceptable runtime, and can be displayed in an incremental manner, i.e., the ontology developers may inspect the first justification while the reasoner continues to find more; and (iii) there is a high degree of commonality among justifications for a subsumption of interest.

BACKGROUND ON DESCRIPTION LOGICS AND JUSTIFICATIONS

In this section, we first introduce the DL $\mathcal{EL}^+$, which is an extension of the underlying logical formalism of SNOMED CT and which is a logical underpinning of OWL 2 EL Profile. Here the DL lingo is adopted, whereas the corresponding OWL constructors are only given for quick reference in Table 1. Then, we give the notion of justification or minimal axiom set, which lies at the heart of logical explanation support in DL systems.

Like other DLs, an $\mathcal{EL}^+$ signature is the disjoint union $\mathbf{S} = \mathsf{CN} \cup \mathsf{RN}$ of the sets of concept names and role names. *$\mathcal{EL}^+$ concept descriptions* (or *complex concepts*) are defined inductively as follows: concept names $A \in \mathsf{CN}$, the bottom concept $\bot$, and the top concept $\top$ are $\mathcal{EL}^+$ concept descriptions. If C, D are $\mathcal{EL}^+$ concept descriptions and $r \in \mathsf{RN}$ a role name, then the conjunction $C \sqcap D$ and existential restriction $\exists r.C$ are also $\mathcal{EL}^+$ concept descriptions. An *$\mathcal{EL}^+$ ontology* $\mathcal{O}$ is a finite set of axioms of the following general forms: general concept inclusion (GCI) $C \sqsubseteq D$, role inclusion $r_1 \circ \cdots r_n \sqsubseteq s$, $\mathrm{Domain}(r) \sqsubseteq C$, and $\mathrm{Range}(r) \sqsubseteq C$. For convenience, we write $\mathsf{Sig}(\mathcal{O})$ (respectively, $\mathsf{Sig}(\alpha)$, $\mathsf{Sig}(C)$) to denote the signature of ontology $\mathcal{O}$ (respectively, axiom α, concept C), i.e., concept and role names occurring in it.

The standard set-theoretic semantics is merely summarized in Table 1, and interested readers are referred to Refs. 21, 22 for detailed explanation. Rather, some intuition is portrayed by means of an example. Table 2 depicts a small example of $\mathcal{EL}^+$ ontology $\mathcal{O}_{\mathsf{med}}$ with 15 axioms motivated by the medical ontology SNOMED CT.

Axioms α_1-α_4 are examples of *primitive concept definitions* which provide only necessary conditions, whereas axioms α_5-α_7 are examples of (*fully defined) concept definitions* which provide both necessary and sufficient conditions. For instance, α_1 states that every appendix is necessarily a body part of the heart which is intendedly incorrect. No sufficient condition is given as to when a body part can be regarded as an appendix. Appendicitis is in contrast fully defined in α_5 with not only necessary but also sufficient conditions, i.e., any inflammation that occurs on an appendix is considered appendicitis. GCIs are typically used to add constraints not already mentioned in concept definitions as in α_{11} and are used to bridge different levels of granularity among concepts in the ontology[23]. Also, each primitive concept definition is a special case of GCI, and each fully defined concept definition can be written by means of two GCIs. Finally, role inclusions generalize at least *role hierarchy* axiom, as in α_{15}; and can be used to impose *reflexivity*, *transitivity*, and *right identity* on roles which are illustrated in axioms α_{12}, α_{13} and α_{14}, respectively. Right identity is particularly useful in biomedical ontologies because it can be used to allow inheritance of certain properties along the aggregation ($\mathsf{part\text{-}of}$) hierarchy[10,24]. With the right identity axiom α_{14} in $\mathcal{O}_{\mathsf{med}}$, for example, if a disease locates in the part (e.g., finger), then it is inferred to locate in the whole (e.g., hand) as well.

The main inference problem for concepts is *subsumption query*: given an ontology $\mathcal{O}$ and two concept descriptions C, D, check if C is subsumed by (i.e., more specific than; or subclass of) D w.r.t. $\mathcal{O}$, written

Table 1 Syntax and semantics of $\mathcal{EL}^+$.

Name	OWL constructor	Syntax	Semantics
top	Thing	$\top$	$\Delta^\mathcal{I}$
conjunction	ObjectIntersectionOf	$C \sqcap D$	$C^\mathcal{I} \cap D^\mathcal{I}$
exists restriction	ObjectSomeValuesFrom	$\exists r.C$	$\{x \in \Delta^\mathcal{I} \mid \exists y \in \Delta^\mathcal{I} : (x,y) \in r^\mathcal{I} \wedge y \in C^\mathcal{I}\}$
concept inclusion	SubClassOf	$C \sqsubseteq D$	$C^\mathcal{I} \subseteq D^\mathcal{I}$
role inclusion	SubObjectPropertyOf	$r_1 \circ \cdots \circ r_n \sqsubseteq s$	$r_1^\mathcal{I} \circ \cdots \circ r_n^\mathcal{I} \subseteq s^\mathcal{I}$
domain restriction	ObjectPropertyDomain	$\mathrm{Domain}(r) \sqsubseteq C$	$\{x \in \Delta^\mathcal{I} \mid (x,y) \in r^\mathcal{I}\} \subseteq C^\mathcal{I}$
range restriction	ObjectPropertyRange	$\mathrm{Range}(r) \sqsubseteq C$	$\{y \in \Delta^\mathcal{I} \mid (x,y) \in r^\mathcal{I}\} \subseteq C^\mathcal{I}$

Table 2 An example $\mathcal{EL}^+$ ontology $\mathcal{O}_{\mathsf{med}}$. The mark $\star$ denotes the axioms in the reachability-based module with signature $\{\mathsf{Endocarditis}\}$, whereas † and ‡ denote the axioms in the two justifications for $\mathsf{Endocarditis} \sqsubseteq_{\mathcal{O}_{\mathsf{med}}} \mathsf{HeartDisease}$.

Axiom ID	$\mathcal{O}^{\mathsf{reach}}_{\mathsf{med}}$	$\mathcal{S}_1$	$\mathcal{S}_2$	Axioms		
α_1				Appendix	$\sqsubseteq$	BodyPart $\sqcap$ $\exists$part-of.Heart
α_2	$\star$	†	‡	Endocardium	$\sqsubseteq$	Tissue $\sqcap$ $\exists$part-of.HeartValve $\sqcap$ $\exists$part-of.HeartWall
α_3	$\star$	†		HeartValve	$\sqsubseteq$	BodyValve $\sqcap$ $\exists$part-of.Heart
α_4	$\star$		‡	HeartWall	$\sqsubseteq$	BodyWall $\sqcap$ $\exists$part-of.Heart
α_5				Appendicitis	$\equiv$	Inflammation $\sqcap$ $\exists$has-location.Appendix
α_6	$\star$	†	‡	Endocarditis	$\equiv$	Inflammation $\sqcap$ $\exists$has-location.Endocardium
α_7				Pancarditis	$\equiv$	Inflammation $\sqcap$ $\exists$has-exact-location.Heart
α_8	$\star$	†	‡	Inflammation	$\sqsubseteq$	Disease $\sqcap$ $\exists$acts-on.Tissue
α_9	$\star$	†	‡	HeartDisease	$\equiv$	Disease $\sqcap$ $\exists$has-location.Heart
α_{10}	$\star$			Tissue $\sqcap$ Disease	$\sqsubseteq$	$\bot$
α_{11}				HeartDisease $\sqcap$ $\exists$causative-agent.Virus	$\sqsubseteq$	ViralDisease $\sqcap$ $\exists$has-state.NeedsTreatment
α_{12}	$\star$			ϵ	$\sqsubseteq$	part-of
α_{13}	$\star$			part-of $\circ$ part-of	$\sqsubseteq$	part-of
α_{14}	$\star$	†	‡	has-location $\circ$ part-of	$\sqsubseteq$	has-location
α_{15}				has-exact-location	$\sqsubseteq$	has-location

$C \sqsubseteq_\mathcal{O} D$ or $\mathcal{O} \models C \sqsubseteq D$. The identification of subsumption relationships between *all* pairs of concept names occurring in $\mathcal{O}$ is known as *ontology classification*. From $\mathcal{O}_{\mathsf{med}}$, the following subsumptions can be inferred:

$$\mathsf{Appendicitis} \sqsubseteq_{\mathcal{O}_{\mathsf{med}}} \mathsf{HeartDisease} \qquad (1)$$

and that:

$$\mathsf{Endocarditis} \sqsubseteq_{\mathcal{O}_{\mathsf{med}}} \mathsf{HeartDisease}. \qquad (2)$$

The first subsumption relation says, relative to $\mathcal{O}_{\mathsf{med}}$, that "every case of appendicitis has to be considered a heart disease," which is obviously not deliberate in the medical domain. This is an example of a false positive inference result. The second subsumption relation however is intuitive and intended to be had.

In this paper, we focus on the problem of finding (all) *justifications* as for why a particular subsumption follows from the ontology. Not only are justifications useful in helping to see what goes wrong in the unintended subsumption (1), it also helps to gain insight into the anticipated subsumption (2). To justify a particular subconcept/superconcept relationship, we need to extract core axioms from the ontology, namely those that are necessarily responsible for the relationship in question. To this end, we define the notion of a *justification* or *MinA* as follows.

Definition 1 [Justification] Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology, A, B concept names such that $A \sqsubseteq_\mathcal{O} B$. Then, a subset $\mathcal{S} \subseteq \mathcal{O}$ is a *minimal axiom set (MinA)* or *justification* for $A \sqsubseteq_\mathcal{O} B$ if and only if $A \sqsubseteq_\mathcal{S} B$ and, for every $\mathcal{S}' \subset \mathcal{S}$, $A \not\sqsubseteq_{\mathcal{S}'} B$.

With respect to the small ontology $\mathcal{O}_{\mathsf{med}}$, it is not difficult to verify that the only justification for subsumption (1) is $\{\alpha_1, \alpha_5, \alpha_8, \alpha_9, \alpha_{14}\}$. Focusing on the five axioms in this justification, it can easily be observed that the culprit that links Appendicitis to HeartDisease is α_1, which incorrectly defines appendix to be a body part of heart. Using justifications this way helps make possible the process of ontology debugging, as merely a few axioms need to be inspected manually by the ontology developer. In general, however, justifications may not be unique, and the number of justifications for a particular subsumption may be exponential in the worst case[18]. Subsumption (2), for instance, has two justifications: $\{\alpha_2, \alpha_3, \alpha_6, \alpha_8, \alpha_9, \alpha_{14}\}$ and $\{\alpha_2, \alpha_4, \alpha_6, \alpha_8, \alpha_9, \alpha_{14}\}$. In other words, endocarditis is an inflammation that occurs on endocardium which is part of the heart valve and the heart wall. The fact that it locates on the heart could either come from the heart valve (i.e., α_3) or the heart wall (i.e., α_4), giving rise to the two justifications above.

Several techniques for finding justifications have been proposed in the literature (see, for instance, Refs. 13, 14, 16, 18, 20, 25, 26) which can be roughly categorized into two approaches: the *glass box* approach, which modifies an existing subsumption testing algorithm in such a way that it keeps track of axioms used for reasoning; and the *black box* approach, which exploits an existing subsumption testing algorithm off the shelf and identifies by means of multiple subsumption tests which axioms are indispensable. The method proposed here is based on the black box approach and on the polynomial-time subsumption testing algorithm[7, 27] implemented in the CEL reasoning system[9, 19].

Since the complexity of finding justifications depends, by and large, on the number of axioms in the ontology under consideration, heuristics have been devised in order to help reduce this number and thus improve the performance of black box justification finding algorithms (see, for example, Refs. 16, 28). To this end, we introduce the notion of module in general and the notion of module that covers all justifications for a subsumption of interest.

Definition 2 [Module] Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology, C, D $\mathcal{EL}^+$ concept descriptions, and $\mathbf{S}$ a signature. Then, an $\mathcal{O}' \subseteq \mathcal{O}$ is a *subsumption module (for short, a module) for* $C \sqsubseteq D$ *in* $\mathcal{O}$ whenever: $C \sqsubseteq_{\mathcal{O}} D$ if, and only if, $C \sqsubseteq_{\mathcal{O}'} D$; and similarly, an $\mathcal{O}' \subseteq \mathcal{O}$ is a *module for* $\mathbf{S}$ *in* $\mathcal{O}$ whenever: for each subsumption σ with $\mathsf{Sig}(\sigma) \subseteq \mathbf{S}$, $\mathcal{O} \models \sigma$ if, and only if, $\mathcal{O}' \models \sigma$. Moreover, an $\mathcal{O}' \subseteq \mathcal{O}$ is a *strong subsumption module for* $C \sqsubseteq D$ *in* $\mathcal{O}$ if, for all justifications $\mathcal{S}$ for $C \sqsubseteq_{\mathcal{O}} D$, $\mathcal{S} \subseteq \mathcal{O}'$; and finally, an $\mathcal{O}' \subseteq \mathcal{O}$ is a *strong subsumption module for* $\mathbf{S}$ *in* $\mathcal{O}$ if, for each subsumption σ with $\mathsf{Sig}(\sigma) \subseteq \mathbf{S}$, if $\mathcal{S}$ is a justification for $\mathcal{O} \models \sigma$, then $\mathcal{S} \subseteq \mathcal{O}'$.

Listing 1 Computation of "one" justification for $A \sqsubseteq_{\mathcal{O}} B$.

```
function naive-one-just($A, B, \mathcal{O}$)
  $\mathcal{S} \leftarrow \mathcal{O}$
  for each axiom $\alpha \in \mathcal{O}$
    if $A \sqsubseteq_{\mathcal{S}\setminus\{\alpha\}} B$ then
      $\mathcal{S} \leftarrow \mathcal{S} \setminus \{\alpha\}$
  return $\mathcal{S}$
```

MODULARIZATION-BASED APPROACH TO FINDING "ONE" JUSTIFICATION

In $\mathcal{EL}^+$, the problem of finding one justification is tractable[18]. In fact, there is a straightforward algorithm to extract a justification by going through all the axioms one by one and checking if the subsumption still holds in absence of each of them. This algorithm is described as naive-one-just in Listing 1. It basically tries to prune each and every axiom in the ontology, unless doing so disposes of the subsumption in question. Since subsumption checking is polynomial and there are linearly many axioms to try, this algorithm runs in polynomial time.

With SNOMED CT, however, this naïve approach does not work due to an obvious reason: it requires almost half a million subsumption tests to find a single justification. To overcome this obstacle, we have proposed a justification finding paradigm based on so-called modularization[20]. The paradigm essentially comprises two stages as shown in Fig. 1 (*left*): (*i*) extract a module $\mathcal{O}'$ from the ontology $\mathcal{O}$, and then (*ii*) find one justification from the module $\mathcal{O}'$.

Observe that the definitions of module given in Definition 2 are generic in the sense that the whole ontology is itself a module. In order for our justification finding paradigm to be effective, however, modules should be small enough to make difference. Therefore, we are interested in certain sufficient conditions that not only give us a module in the sense imposed in Definition 2 but also guarantee relevancy of extracted axioms. The proposed justification finding paradigm employs the *reachability-based module* which has been first introduced in Ref. 29.

Definition 3 [Reachability-based modules] Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology, $\mathbf{S} \subseteq \mathsf{Sig}(\mathcal{O})$ a signature, and $x, y \in \mathsf{Sig}(\mathcal{O})$ concept or role names. We say that x is *connectedly reachable* from $\mathbf{S}$ w.r.t. $\mathcal{O}$ (for short, *reachable* from $\mathbf{S}$ or $\mathbf{S}$-*reachable*) iff $x \in \mathbf{S}$ or there is an axiom (either GCI or RI) $\alpha_L \sqsubseteq \alpha_R \in \mathcal{O}$ s.t. $x \in \mathsf{Sig}(\alpha_R)$ and, for all $y \in \mathsf{Sig}(\alpha_L)$, y is reachable from $\mathbf{S}$.

We say that an axiom $\beta_L \sqsubseteq \beta_R$ is *reachable*

Listing 2 Modularization-based computation of "one" justification $\mathcal{S}$ for $A \sqsubseteq_{\mathcal{O}} B$.

```
1  function mod-one-just($A, B, \mathcal{O}$)
2    $\mathcal{O}_A^{\text{reach}} \leftarrow$ extract-module($\mathcal{O}, \{A\}$)
3    return naive-one-just($A, B, \mathcal{O}_A^{\text{reach}}$)
4
5  function extract-module($\mathcal{O}, \mathbf{S}$)
6    $\mathcal{O}_{\mathbf{S}} \leftarrow \emptyset$
7    queue $\leftarrow$ active-axioms($\mathbf{S}$)
8    while not empty(queue) do
9      $(\alpha_L \sqsubseteq \alpha_R) \leftarrow$ fetch(queue)
10     if $\mathsf{Sig}(\alpha_L) \subseteq \mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_{\mathbf{S}})$ then
11       $\mathcal{O}_{\mathbf{S}} \leftarrow \mathcal{O}_{\mathbf{S}} \cup \{\alpha_L \sqsubseteq \alpha_R\}$
12       enqueue(queue, active-axioms($\mathsf{Sig}(\alpha_R)$) $\setminus \mathcal{O}_{\mathbf{S}}$)
13   return $\mathcal{O}_{\mathbf{S}}$
```

from $\mathbf{S}$ w.r.t. $\mathcal{O}$ (for short, $\mathbf{S}$-*reachable*) if, for all $x \in \mathsf{Sig}(\beta_L)$, x is $\mathbf{S}$-reachable. *The reachability-based module* for $\mathbf{S}$ in $\mathcal{O}$, denoted by $\mathcal{O}_{\mathbf{S}}^{\text{reach}}$, is the smallest set of all axioms in $\mathcal{O}$ that are $\mathbf{S}$-reachable, i.e., $\mathcal{O}_{\mathbf{S}}^{\text{reach}} = \{\alpha \in \mathcal{O} | \alpha \text{ is } \mathbf{S}\text{-reachable in } \mathcal{O}\}$.

It seems worth mentioning that by definition this module is unique for a given ontology and signature, so we can speak of *the* reachability-based module. Listing 2 outlines a method extract-module for extracting the reachability-based module given as input an $\mathcal{EL}^+$ ontology $\mathcal{O}$ and a signature $\mathbf{S}$. Here, the ontology is viewed as a mapping active-axioms : $\mathsf{Sig}(\mathcal{O}) \rightarrow 2^{\mathcal{O}}$, which maps each symbol in the ontology to a set of axioms from the ontology. More precisely, active-axioms(x) comprises all and only axioms $\alpha_L \sqsubseteq \alpha_R \in \mathcal{O}$ such that x occurs in α_L. The intuition is that every axiom $\alpha \in$ active-axioms(x) is 'active' for x in the sense that, for some $y \in \mathsf{Sig}(\mathcal{O})$, y potentially is x-reachable via α. For convenience, we define active-axioms$(\mathbf{S}) := \bigcup_{x \in \mathbf{S}}$ active-axioms(x) for a signature $\mathbf{S} \subseteq \mathsf{Sig}(\mathcal{O})$.

Starting from the symbols in a given signature $\mathbf{S}$, the function extract-module keeps adding $\mathbf{S}$-reachable axioms into $\mathcal{O}_{\mathbf{S}}$. $\mathbf{S}$-reachability is checked by a simple set inclusion (line 10). When a new axiom is added to $\mathcal{O}_{\mathbf{S}}$, reachability propagates to the symbols on its right-hand side, i.e., $\mathsf{Sig}(\alpha_R)$. Unless previously accomplished, the active axioms for these new symbols are enqueued for later processing (line 12). This strategy seems to work well with ontologies whose symbols are evenly distributed. If a symbol x occurs on the left-hand side of every axiom in the ontology, then active-axioms(x) yields that ontology as a whole. Unfortunately, this uneven distribution does occur in SNOMED CT stemming from the use of role grouping, i.e., an ontology modelling discipline that requires every existential restriction to be nested inside the special role called roleGroup. For instance, the right-hand side of α_2 in $\mathcal{O}_{\mathsf{med}}$ (Table 2) is to be rewritten to:

$$\begin{aligned}\mathsf{Tissue} &\sqcap \exists \mathsf{roleGroup}.(\exists \mathsf{part\text{-}of}.\mathsf{HeartValve}) \\ &\sqcap \exists \mathsf{roleGroup}.(\exists \mathsf{part\text{-}of}.\mathsf{HeartWall}),\end{aligned}$$

should this modelling discipline be employed. An example concept TetralogyOfFallot was provided in Ref. 30 to rationalize the need of role grouping in SNOMED CT. In essence, the concept describes a collection of four medical conditions, each with two facets: the site and the morphology. These two facets of each condition must be grouped together, or the interpretation of the concept could be intertwined.

The implication of this modelling discipline to our module extraction method is that roleGroup occurs virtually everywhere, and thus queue in the module extraction method (Listing 2) will be populated with a large number of axioms in SNOMED CT which results in unnecessarily slow module extraction. By inspecting role grouping and the ontological structure under scrutiny, it is actually the case that the symbol roleGroup is irrelevant to the module extraction method on SNOMED CT. In fact, roleGroup is always reachable whenever there is an existential restriction on the right-hand side of an axiom, and every existential restriction on the left-hand side always come with roleGroup. To this end, the implementation of extract-module also employs this SNOMED CT-specific optimization to boost extraction running time. The empirical results reported later in the article show that it is a split of second to extract a module from SNOMED CT, and that the modules are generally very small compared to the ontology where the number of axioms can be reduced up to four orders of magnitude.

In Ref. 29, it has been shown that, in order to query subsumption $A \sqsubseteq B$ against an ontology $\mathcal{O}$, it suffices to consider axioms in the reachability-based module $\mathcal{O}_A^{\text{reach}}$ for $\mathbf{S} = \{A\}$. Following this observation, a connection between modules and justifications can be established in the case where the subsumption holds. Given an ontology $\mathcal{O}$ and concept names A, B such that $A \sqsubseteq_{\mathcal{O}} B$; then, $A \sqsubseteq_{\mathcal{O}_A^{\text{reach}}} B$. It immediately implies that there is a minimal subset $\mathcal{S} \subseteq \mathcal{O}_A^{\text{reach}} \subseteq \mathcal{O}$ such that $A \sqsubseteq_{\mathcal{S}} B$ and, for every $\mathcal{S}' \subset \mathcal{S}$, $A \not\sqsubseteq_{\mathcal{S}'} B$. Obviously, such a minimal subset $\mathcal{S}$ of $\mathcal{O}_A^{\text{reach}}$ is a justification for $A \sqsubseteq_{\mathcal{O}} B$. Listing 2 summarizes the proposed justification finding paradigm that allows CEL to be able to find a justification.

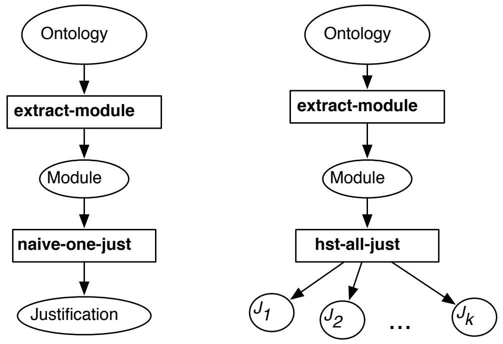

Fig. 1 The modularization-based paradigm for finding one justification (*left*) and all justifications (*right*).

From the example in Table 2, the reachability-based module for $\mathbf{S} = \{\mathsf{Endocarditis}\}$ contains α_2, α_3, α_4, α_6, α_8, α_9, α_{10}, α_{12}, α_{13} and α_{14}; and it can be verified that the two justifications for subsumption (2) are indeed covered by this module.

MODULARIZATION-BASED APPROACH TO FINDING "ALL" JUSTIFICATIONS

The paradigm presented in the previous section essentially comprises two stages as shown in Fig. 1 (left): (i) extract a module $\mathcal{O}'$ from the ontology $\mathcal{O}$, and then (ii) compute a justification from the module $\mathcal{O}'$ by the axiom pruning algorithm. This section describes an extension to this paradigm by retaining the first stage and employing another algorithm for the second stage to find "all" justifications from the module $\mathcal{O}'$. Clearly, in order for this paradigm to be complete, i.e., all existing justifications can be identified, the module has to cover all the justifications. To this end, we have shown that the reachability-based module has precisely this coverage property, i.e., it is a strong subsumption module as defined in Definition 2. As a direct consequence of this result, it suffices to consider only axioms in the module and ignore those other axioms when "all" justifications are to be computed.

Lemma 1 ($\mathcal{O}_A^{\mathsf{reach}}$ is a strong subsumption module) *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology, A, B concept names such that $A \sqsubseteq_{\mathcal{O}} B$, and $\mathcal{O}_A^{\mathsf{reach}} \subseteq \mathcal{O}$ the reachability-based module for $\mathbf{S} = \{A\}$ in $\mathcal{O}$. Then, $\mathcal{S} \subseteq \mathcal{O}_A^{\mathsf{reach}}$, for all justifications $\mathcal{S}$ for $A \sqsubseteq_{\mathcal{O}} B$.*

This lemma suggests that the proposed paradigm in Fig. 1 (*right*) is complete in the sense that no justifications, if any, will be missed out. The paradigm then applies a modification of the known *hitting set tree (HST) algorithm*[31], as well as some optimization techniques[32]. The intuition behind this algorithm is to attempt to remove some axioms found in the already known justifications to see if the subsumption still holds. If this is affirmative, it clearly means that another justification not containing those removed axioms must exist. This justification must be different from all the known ones since it does away with some axioms found in each of them, and thus can be extracted by using any axiom pruning method such as naive-one-just in Listing 1.

Listing 3 HST justification finding algorithm.

```
1  function hst-all-justs($A, B, \mathcal{O}$)
2    global J ← ∅, H ← ∅
3    $\mathcal{S}$ ← naive-one-just($A, B, \mathcal{O}$)
4    J ← {$\mathcal{S}$}
5    for each axiom $\alpha \in \mathcal{S}$
6      expand-hst($A, B, \mathcal{O}\setminus\{\alpha\}, \{\alpha\}$)
7    return J
8
9  procedure expand-hst($A, B, \mathcal{O}, H$)
10   if there exists some $H' \in \mathbf{H}$ such that $H' \subseteq H$ or
11      $H'$ contains a prefix-path $P$ with $P = H$ then
12     return                    (early path termination ⊗ )
13   if there exists some $\mathcal{S} \in \mathbf{J}$ such that $H \cap \mathcal{S} = \emptyset$ then
14     $\mathcal{S}' \leftarrow \mathcal{S}$                          (justification reuse)
15   else
16     $\mathcal{S}'$ ← naive-one-just($A, B, \mathcal{O}$)
17   if $\mathcal{S}' \neq \emptyset$ then
18     $\mathbf{J} \leftarrow \mathbf{J} \cup \{\mathcal{S}'\}$
19     for each axiom $\alpha \in \mathcal{S}'$ do
20       expand-hst($A, B, \mathcal{O}\setminus\{\alpha\}, H \cup \{\alpha\}$)
21   else
22     $\mathbf{H} \leftarrow \mathbf{H} \cup \{H\}$              (normal termination ⊙ )
```

Listing 3 describes in detail a pinpointing algorithm based on a hitting set tree (HST) expansion that finds all justifications for a particular subsumption $A \sqsubseteq B$ w.r.t. $\mathcal{O}$. In the procedure hst-all-justs, two global variables $\mathbf{J}$ and $\mathbf{H}$ are declared and initialized with $\emptyset$. They are used throughout the HST expansion to store the computed justifications and hitting sets, respectively. In line 3, a first justification $\mathcal{S}$ is computed in the usual way by the function naive-one-just. The root of the HST is formed with $\mathcal{S}$ as its label. Branches from the root are then spawned by calling the recursive procedure expand-hst for every axiom $\alpha \in \mathcal{S}$ (line 5–6). Line 10 to 14 in expand-hst implement two optimizations for the HST algorithms[16,31,32] that help reduce the size of the HST and minimize calls to the sub-procedure naive-one-just, and thus calls to the subsumption testing procedure.

Early path termination An HST branch can be pruned if a *similar* one has been considered earlier. Here, similarity is determined by the set of removed axioms or the hitting set H.

Justification reuse A known justification $\mathcal{S}$ can be reused in the current HST branch if this justification $\mathcal{S}$ does not use any of the removed axioms in H, i.e., $\mathcal{S}$ and H are disjoint.

In line 16 in expand-hst, the reduced ontology $\mathcal{O}\backslash H$, i.e., the original ontology dispensed with axioms in H, is then pruned to see if another justification $\mathcal{S}'$ exists. If yes ($\mathcal{S}' \neq \emptyset$), $\mathcal{S}'$ is the next justification that labels a new node in the HST, and branches from this new node are expanded by means of recursive calls to expand-hst in line 17–20. The expansion is carried out by removing each of the axioms in $\mathcal{S}'$ from the current reduced ontology, i.e., $\mathcal{O}\backslash(H \cup \{\alpha\})$ for each $\alpha \in \mathcal{S}'$. Otherwise ($\mathcal{S}' = \emptyset$), the subsumption no longer holds in the reduced ontology $\mathcal{O}\backslash H$, and thus a hitting set H of all justifications is obtained in line 21–22.

To fully understand hst-all-justs and expand-hst, consider a small ontology $\mathcal{O}_2$ comprising 5 axioms:

$$\begin{array}{llll}
\alpha_1: & A \sqsubseteq P_1 \sqcap Q_1, & & \\
\alpha_2: & P_1 \sqsubseteq P_2 \sqcap Q_2, & \alpha_4: & P_2 \sqsubseteq B, \\
\alpha_3: & Q_1 \sqsubseteq P_2 \sqcap Q_2, & \alpha_5: & Q_2 \sqsubseteq B;
\end{array}$$

and entailing the subsumption $\sigma = (A \sqsubseteq B)$. This example can be generalized to $\mathcal{O}_n$ comprising α_1, $P_n \sqsubseteq B$, $Q_n \sqsubseteq B$, $P_i \sqsubseteq P_{i+1} \sqcap Q_{i+1}$ and $Q_i \sqsubseteq P_{i+1} \sqcap Q_{i+1}$, for $i < n$. Although the size of $\mathcal{O}_n$ is linear in n, there exist 2^n distinct justifications for $A \sqsubseteq_{\mathcal{O}_n} B$. In $\mathcal{O}_2$, There are 4 justifications for σ. Fig. 2 demonstrates the process of computing all justifications by Listing 3.

To begin with, $\mathcal{O}_2$ is pruned by naive-one-just to obtain the first justification $\{\alpha_1, \alpha_2, \alpha_4\}$, which is the label of the root node n_0 of the HST. The first branch terminates immediately since $\mathcal{O}_2\backslash\{\alpha_1\}$ does not entail σ (marked by $\odot$ in n_1). On the other hand, $\mathcal{O}_2\backslash\{\alpha_2\} \models \sigma$, and thus the second justification $\{\alpha_1, \alpha_3, \alpha_4\}$ can be computed by pruning $\mathcal{O}_2\backslash\{\alpha_2\}$ which labels n_2. This process continues to expand HST until it finds all other justifications for σ and tree exploration is exhausted. Observe that the underlined label of n_9 in the right-most branch is the result of the *justification reuse* optimization, where a known justification not containing α_4, viz. $\{\alpha_1, \alpha_3, \alpha_5\}$, is taken from the previously expanded node n_5.

Finally, we apply reachability-based modularization to Listing 3 to obtain the optimized method for finding all justification (see Listing 4). With the SNOMED CT-specific optimization described in the previous section, module extraction time is minuscule compared to the time required for HST expansion, pruning axioms and testing subsumption. This proposed method is highly effective on SNOMED CT since the large search space given by all the axioms in this medical ontology is drastically reduced to a much smaller module.

Listing 4 Modularization-based computation of "all" justifications for $A \sqsubseteq_{\mathcal{O}} B$.

```
function mod-all-justs($A, B, \mathcal{O}$)
  $\mathcal{O}_A^{\text{reach}} \leftarrow$ extract-module($\mathcal{O}, \{A\}$)
  return hst-all-justs($A, B, \mathcal{O}_A^{\text{reach}}$)
```

EXPERIMENTAL RESULTS

We have implemented the proposed modularization-based algorithms by using CEL[9] as the subsumption oracle required in Listing 1 which is a sub-procedure for Listing 4. The implemented algorithm is tested using a DL version of SNOMED CT, which contains 379 691 concept names and 62 role names. It is an $\mathcal{EL}^+$ ontology with 379 704 axioms, 13 of which are role axioms. Henceforth, we refer to this $\mathcal{EL}^+$ ontology as $\mathcal{O}^{\text{SNOMED}}$. All experimental results presented in the paper have been carried out and measured by a PC with 2.40 GHz Pentium-4 processor and 1 GB of memory.

Results on modularization

As mentioned earlier, the modularization-based approach to finding all justifications (Listing 4) will be effective if two conditions can be attained: (i) modules are relatively small and (ii) module extraction is fast. Extensive experiments on reachability-based module extraction w.r.t. a singleton signature, i.e., extraction of $\mathcal{O}_A^{\text{reach}}$ for a concept name A, in SNOMED CT have been performed, and the results are satisfactory according to the two conditions. The reachability-based modules comprise only 31 axioms on average and 262 axioms at the maximum; these figures account for only 0.008% and 0.069% of the size of SNOMED CT in terms of the number of axioms. Thanks to the SNOMED CT-specific optimization described above, extraction of such reachability-based modules was very fast, especially when compared with subsumption testing time or HST expansion time. Apart from an initialization for module extraction which took less than a minute and was performed only once when $\mathcal{O}^{\text{SNOMED}}$ was loaded, extraction of each module

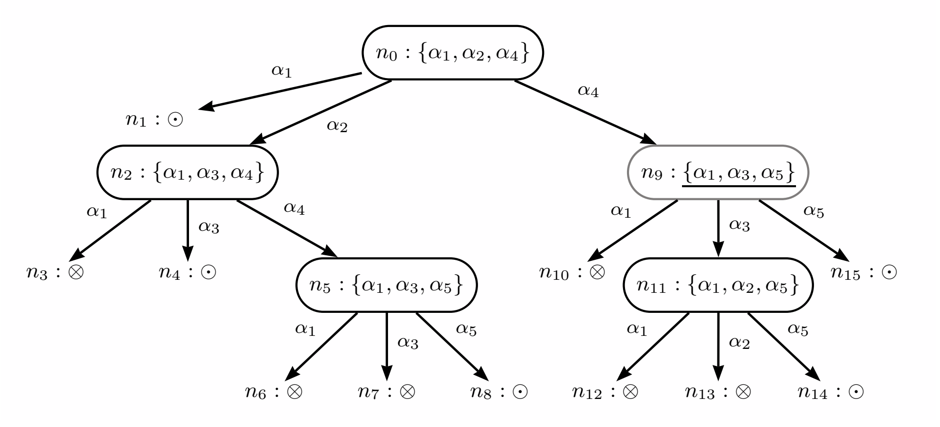

Fig. 2 The hitting set tree produced by the HST justification finding algorithm (Listing 3) on the example ontology $\mathcal{O}_2$.

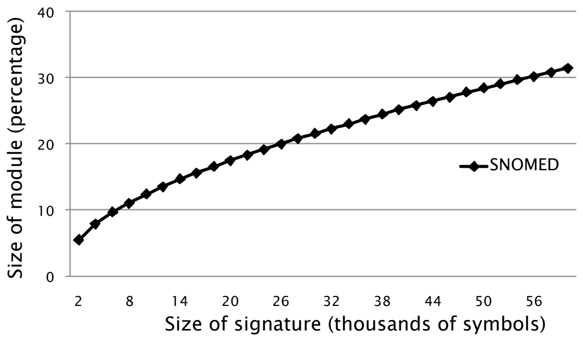

Fig. 3 Module sizes for large signatures. The vertical axis represents in percentage the module size (i.e., % of ♯axioms in the module w.r.t. ♯axioms in $\mathcal{O}^{\text{SNOMED}}$).

took 8.2 *milli*seconds on average and 5.46 s at the maximum. It is worth noting that once a module $\mathcal{O}_A^{\text{reach}}$ had been extracted, it could be used in the justification finding method for multiple subsumptions with A as the left-hand concept.

In addition, we have performed more experiments on reachability-based module extraction to see how it would behave if the signature of interest grew. Signatures of varying sizes between 2000 and 60 000 symbols have been sampled and fed into the extraction algorithm. Fig. 3 depicts the results that show a slow growth of module size relative to the controlled growth of input signature. On average the reachability-based modules are four orders of magnitude smaller than the original ontology. Surprisingly, it needed almost 15% of $\text{Sig}(\mathcal{O}^{\text{SNOMED}})$ in order to obtain as large a module as 30% of its axioms. This result suggests that module extraction based on reachability is quite robust, both in terms of extraction time and module size.

Module sizes versus justification sizes

To get a grip of what results we may expect to achieve, some previous results[20] w.r.t. finding "one" justifications in $\mathcal{O}^{\text{SNOMED}}$ are summarized here for reference. The version of SNOMED CT used for experiments entails a false positive subsumption that "amputation of finger is amputation of hand." It was not possible within 24 h for naive-one-just in Listing 1 to find a justification for this false positive subsumption, whereas mod-one-just in Listing 2 took less than a second to find a justification which comprises merely 6 axioms. This highly effective optimization technique stems from the fact that the module contains only 57 axioms, as opposed to hundreds of thousands in $\mathcal{O}^{\text{SNOMED}}$. Overall sizes of modules in $\mathcal{O}^{\text{SNOMED}}$, as well as of justifications (MinAs), are presented in Fig. 4. On average the reachability-based modules are *four orders of magnitude* smaller than the original ontology. For details on modularization experimental results see Ref. 29. Observe that, though the modules in SNOMED CT are relatively small, the actual sizes of justifications are smaller but not by much. On average, the justification size is 13.36% that of modules. As easily visible from the chart, the majority of subsumptions (i.e., about 80% depicted by the first two bars in Fig. 4) have a justification of size ten or less axioms.

Results on finding justifications

Note that the majority of over five million subsumption relationships in SNOMED CT have a single justification. But, since we focus here on experiments on finding "all" justifications in $\mathcal{O}^{\text{SNOMED}}$, only subsumptions known to have strictly more than one justification are considered. To this end, we have sampled 18 673

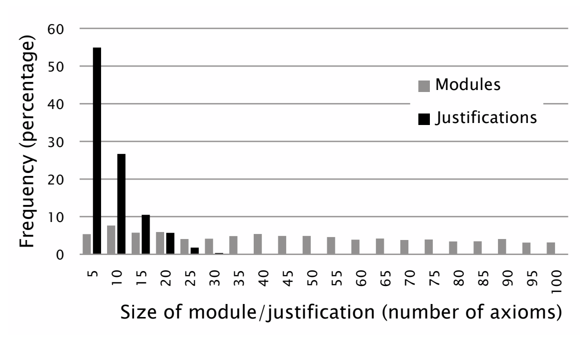

Fig. 4 Histogram of the sizes of justifications, in comparison with those of modules, for the sampled subsumptions in $\mathcal{O}^{\textsc{Snomed}}$. The vertical axis represents in percentage the number of justifications/modules.

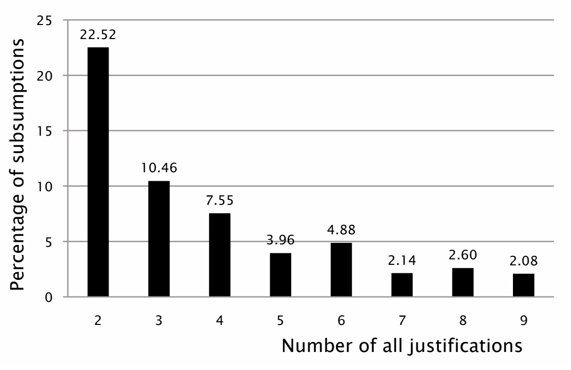

Fig. 5 Histogram of the numbers of *all* justifications for easy-samples. The vertical axis represents in percentage the number of subsumptions, e.g., about 10% of the sampled subsumptions in easy-samples have 3 justifications.

Table 3 Statistical results on the computed justifications in $\mathcal{O}^{\textsc{Snomed}}$ (average/maximum).

Samples	♯Justs	Just size	♯Common ax. (μ)	♯All ax. (ν)
easy-samples (56.19%)	3.7 / 9	8.0 / 26	4.8 / 22	12.3 / 39
hard-samples (43.81%)	10 / 10	16.4 / 45	7.0 / 30	32.5 / 63

subsumption relationships that are well-distributed and representative of the entirety of the ontology.

Among the sampled subsumptions are hard cases with more than 100 justifications. Despite the optimizations, the constructed hitting set tree was very large, and it took more than 24 h and up to 72 h to compute all justifications. For this reason and in order to gather sufficient statistics, the number of computed justifications was limited in our experiment to 10, and the samples would therefore be divided into two groups, namely, easy-samples which comprises subsumptions having between 2 and 9 justifications; and hard-samples which comprises subsumptions having at least 10 justifications.

Based on all the subsumptions considered, 10 492 (56.19%) subsumptions belong to easy-samples, and 8181 (43.81%) subsumptions to hard-samples. Table 3 shows the average/maximum numbers of justifications (♯Justs) and their sizes, respectively, in the first two column. Given a subsumption σ, the number of common axioms in all its justifications, in symbols $\mu(\sigma)$, is defined by $|\bigcap_{\text{justifications } \mathcal{S} \text{ for} \sigma} \mathcal{S}|$, i.e., the number of those axioms that mutually occur in all justifications for σ. Likewise, the number of all relevant axioms in all its justifications, in symbols $\nu(\sigma)$, is defined by $|\bigcup_{\text{justifications } \mathcal{S} \text{ for} \sigma} \mathcal{S}|$, i.e., the number of all those axioms that occur in some justification for σ. Table 3 depicts in the third column the average/-maximum numbers (μ) of common axioms, and in the forth column the average/maximum numbers (ν) of all relevant axioms. The average ratio μ/ν, which indicates the degree of commonality of the computed justifications, is 0.39 and 0.22 for easy-samples and hard-samples, respectively. This degree suggests how much the justifications for the same subsumption have in common, and more attention may be paid on the shared axioms when debugging the subsumption as they play a role in each and every justification.

The relative distribution of ♯Justs below ten is shown in Fig. 5. More than half of all the considered subsumptions (51.51%) have 7 justifications or less, i.e., the median of ♯Justs for easy-samples and hard-samples collectively is 7. (It is worth noting that, when considering all subsumptions in $\mathcal{O}^{\textsc{Snomed}}$ including those with one justification, the median of ♯Justs is one.) Although nothing can be said about the distribution of ♯Justs larger than 9, it is known from the test results that about 43% have ten or more justifications. Among the hard sampled subsumptions, the largest known number of justifications per subsumption is 158, whereas the largest known justification comprises 45 axioms.

Using the modularization-based HST pinpointing algorithm on $\mathcal{O}^{\textsc{Snomed}}$, the average time to compute all justifications for a subsumption in easy-samples was 8.8 s, whereas the average time to compute the first 10 justifications for a subsumption in hard-samples was 37.8 s.

Recall that our proposed algorithm consists of three core computing components: module extraction, HST construction, and subsumption testing (see Listing 3 and 4). Interestingly, most of the overall computation time was spent on subsumption testing. On average, the algorithm made 178 subsumption calls for easy-samples and 770 subsumption calls for hard-samples. Small fractions of time were used for module extraction and HST construction. Precisely, the module extraction took 0.01 and 0.02 s for easy-samples and hard-samples, respectively; whereas the HST construction excluding subsumption testing took 0.07 and 0.09 s for easy-samples and hard-samples, respectively.

RELATED WORK

Several techniques for finding justifications have been proposed in the literature which can be roughly categorized into two approaches: the glass box and the black box approach.

The main tenet of the glass box approach is to modify an existing reasoning algorithm in such a way that it keeps track of axioms used for reasoning. Several work, e.g. Refs. 13, 14, 25, 33, follow this approach. The main drawback of this approach is that when the reasoning algorithm is modified, the justification finding algorithm must be modified accordingly. Moreover, original axioms may have to be manipulated or normalized for internal processing. Such a normalization usually hinders axiom tracking in the glass box approach. Finally, several optimizations have to be disabled to allow for finding justifications in this way because otherwise they would make permanent change to the structures of axioms[16].

The black box approach, on the other hand, does not modify the internals of an existing reasoning algorithm but rather exploits it per se. (Subsumption test on line 4 in Listing 1 provides an example.) Justification finding algorithms are obtained by repeatedly calling the subsumption algorithm as a black box and by dispensing with axioms in the ontology whenever deemed irrelevant to the subsumption in question. Listing 1 describes the most straightforward axiom pruning method that removes at most one axiom in every iteration. The so-called sliding window technique[16] improves the naïve algorithm slightly by allowing a number of axioms (window size) to be removed from the ontology in every iteration. Although this can reduce the number of subsumption tests, there is still a trade-off because pruning has to be repeated over and over until the window size reaches 1 which is the same as Listing 1. In our earlier work[20], we proposed to employ the binary search technique of "halve and check" to prune axioms. Instead of going through axioms one by one, this technique partitions the ontology into two halves, and subsumption is checked w.r.t. each half. This technique requires a logarithmic number of subsumption tests in the best case but could also degrade to linearity if the justification and ontology are not much different in size. Since reachability-based modules are readily so small, i.e., on average 13.36% of the axioms in the module are relevant in the sense that it occurs in a justification, it did not pay off to use this idea. In fact, a separate experiment has shown that such an advanced pruning strategy actually degrades the overall performance.

Various techniques for extracting fragments of ontologies have been proposed in the literature. One technique[34] considers only concept definitions in the ontology, not GCIs nor role inclusions, and as such does not produce a module according to Definition 2. Another technique[35] is similar to our reachability-based modularization but does not make distinction between left-hand and right-hand symbols. It yields a very large module, and in many cases, an entire ontology. Syntactic locality-based modules[36] have been defined for the Web Ontology Language (OWL). A minimal such module has been shown to correspond to our reachability-based module[22] and can also be employed in a justification finding framework for OWL[37]. Extensions to both locality-based modules and reachabiltiy-based modules have resulted in "nested locality-based modules"[38,39] and "bidirectional reachability-based modules"[40,41]. Given a subsumption $A \sqsubseteq B$, each of these modules is obtained by two extractions, one w.r.t. the subsumer B and the other w.r.t. the subsumee A. These modules are often smaller than the reachability-based modules, but they are nevertheless specific to only one particular subsumption and require longer time for extraction.

CONCLUSIONS AND DISCUSSION

This article describes a framework for finding all justifications in SNOMED CT. The large scale but simple structure of this medical ontology can effectively be handled by modularization. We have proposed to employ the reachability-based modularization as to greatly reduce the search space and then employ the hitting set tree algorithm for finding all justifications from the module. For this approach to work, we have shown that the reachability-based modules cover all justifications for the subsumption of interest.

We have performed an extensive number of experiments to empirically prove the effectiveness and efficiency of this approach. One advantage of this algorithm is that it computes the first justification in less than half a second and generates next justifications one after the other in an incremental manner. This means that, while the computation is being carried out, partial outputs, i.e., some justifications, are readily available for inspection by the ontology developer. An excessively long computation of all justifications in certain "hard" cases can be interrupted when the developer has enough information for debugging.

This work has demonstrated the effectiveness of search space reduction before HST expansion algorithm starts. Any modules that cover all justifications can be employed in this framework. In particular, nested locality-based modules[39] and bidirectional reachability-based modules[41] could be subject for future investigation.

Our experimental results regarding axiom commonality across justifications have demonstrated that the various justifications for a particular subsumption of interest are highly interdependent. The core axioms that occur in all justifications likely play a role in better understanding of the structure of subsumptions. In fact, one could define a hierarchy of subsumptions, in which a lower subsumption can be said to depend on an upper subsumption. The similar notion of *justificatory structure* has been introduced[42]. For future work, we plan to formalize the notion of subsumption hierarchy and analyse its usefulness in ontology design and development, especially of SNOMED CT.

Acknowledgements: This work was supported by Thailand Research Fund under grant MRG5380245 and the National Research University Project of Thailand Office for Higher Education Commission. The author would like to thank the anonymous reviewers for their fruitful suggestions and language correction.

REFERENCES

1. Stearns M, Price C, Spackman K, Wang A (2001) SNOMED clinical terms: Overview of the development process and project status. In: Proceedings of the 2001 AMIA Annual Symposium, Hanley&Belfus, pp 662–6.
2. Spackman K (2005) Rates of change in a large clinical terminology: Three years experience with SNOMED clinical terms. In: Proceedings of the 2005 AMIA Annual Symposium, Hanley&Belfus, pp 714–8.
3. Spackman K, Campbell K, Cote R (1997) SNOMED RT: A reference terminology for health care. In: Proceedings of the 1997 AMIA Annual Fall Symposium, Hanley&Belfus, pp 640–4.
4. O'Neil M, Payne C, Read J (1995) Read Codes Version 3: A user led terminology. *Methods of Information in Medicine* **34**, 187–921.
5. Spackman K (2001) Normal forms for Description Logic expressions of clinical concepts in SNOMED RT. In: Proceedings of the 2001 AMIA Annual Symposium, Hanley&Belfus, pp 627–31.
6. Baader F, Calvanese D, McGuinness D, Nardi D, Patel-Schneider P (eds) (2007) *The Description Logic Handbook: Theory, Implementation and Applications*, 2nd edn, Cambridge University Press.
7. Baader F, Brandt S, Lutz C (2005) Pushing the $\mathcal{EL}$ envelope. In: Proceedings of the 19th International Conference on Artificial Intelligence (IJCAI'05), Morgan-Kaufmann Publishers, Edinburgh, UK, pp 364–9.
8. Motik B, Grau BC, Horrocks I, Wu Z, Fokoue A, Lutz C (2009) OWL 2 web ontology language profiles, http://www.w3.org/TR/owl2-profiles/.
9. Baader F, Lutz C, Suntisrivaraporn B (2006) CEL—a polynomial-time reasoner for life science ontologies. In: Furbach U, Shankar N (eds) *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR'06)*, Springer-Verlag, vol 4130 of *Lecture Notes in Artificial Intelligence*, pp 287–91.
10. Suntisrivaraporn B, Baader F, Schulz S, Spackman K (2007) Replacing SEP-triplets in SNOMED CT using tractable Description Logic operators. In: Riccardo Bellazzi JH, Ameen Abu-Hanna (eds) *Proceedings of the 11th Conference on Artificial Intelligence in Medicine (AIME'07)*, Springer-Verlag, vol 4594 of *Lecture Notes in Computer Science*, pp 287–91.
11. Schulz S, Markó K, Hahn U (2007) Spatial location and its relevance for terminological inferences in bio-ontologies. *BMC Bioinformatics* **8**, 134–46.
12. Ceusters W, Smith B, Kumar A, Dhaen C (2004) SNOMED CT's problem list: Ontologists' and logicians' therapy suggestions. In: Marius Fieschi YCJL, Enrico Coiera (eds) *Proceedings of the Medinfo 2007 Congress*, Studies in Health Technology and Informatics (SHTI-series), IOS Press, pp 482–91.
13. Schlobach S, Cornet R (2003) Non-standard reasoning services for the debugging of Description Logic terminologies. In: Gottlob G, Walsh T (eds) *Proceedings of the 17th International Conference on Artificial Intelligence (IJCAI-03)*, Morgan-Kaufmann Publishers, Acapulco, Mexico, pp 355–62.
14. Meyer T, Lee K, Pan J, Booth R (2006) Finding maximally satisfiable terminologies for the Description Logic $\mathcal{ALC}$. In: Proceedings of the 21st National Conference on Artificial Intelligence (AAAI'06), pp 269–74.
15. Kalyanpur A (2006) Debugging and repair of OWL ontologies. PhD thesis, Univ of Maryland.
16. Kalyanpur A, Parsia B, Horridge M, Sirin E (2007) Finding all justifications of OWL DL entailments. In: Proceedings of the 6th International Semantic Web Conference (ISWC'07), pp 267–80.
17. Horridge M (2011) Justification based explanation in ontologies. PhD thesis, Univ of Manchester.
18. Baader F, Peñaloza R, Suntisrivaraporn B (2007) Pinpointing in the Description Logic $\mathcal{EL}^+$. In: Proceedings of the 30th German Conference on Artificial Intelligence (KI'07), LNAI, Springer, Osnabrück, Germany, pp 52–67.

19. Suntisrivaraporn B (2008) A lightweight description logic reasoner for large-scale biomedical ontologies, http://cel.googlecode.com.
20. Baader F, Suntisrivaraporn B (2008) Debugging SNOMED CT using axiom pinpointing in the Description Logic $\mathcal{EL}^+$. In: Proceedings of the 3rd Knowledge Representation in Medicine Conference (KR-MED'08): Representing and Sharing Knowledge Using SNOMED, Phoenix AZ, USA.
21. Baader F, Lutz C, Suntisrivaraporn B (2007) Is tractable reasoning in extensions of the Description Logic $\mathcal{EL}$ useful in practice? *Journal of Logic Language and Information Special Issue on Method for Modality M4M*.
22. Suntisrivaraporn B (2009) Polynomial-time reasoning support for design and maintenance of large-scale biomedical ontologies. PhD thesis, TU Dresden.
23. Horrocks I, Rector A, Goble C (1996) A Description Logic based schema for the classification of medical data. In: Baader F, Buchheit M, Jeusfeld MA, Nutt W (eds) *Knowledge Representation Meets Databases, Proceedings of the 3rd Workshop KRDB'96, Budapest, Hungary, August 13, 1996*, vol 4 of *CEUR Workshop Proceedings*.
24. Spackman K (2000) Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED RT. *American Medical Informatics Association Fall Symposium* Fall Symposium Special Issue.
25. Parsia B, Sirin E, Kalyanpur A (2005) Debugging OWL ontologies. In: Proceedings of the 14th International Conference on World Wide Web (WWW-2005), ACM, pp 633–40.
26. Baader F, Peñaloza R (2010) Axiom pinpointing in general tableaux. *J Logic Comput* **20**, 5–34.
27. Baader F, Brandt S, Lutz C (2008) Pushing the $\mathcal{EL}$ envelope further. In: Clark K, Patel-Schneider PF (eds) *In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*.
28. Ji Q, Qi G, Haase P (2009) A relevance-directed algorithm for finding justifications of DL entailments. In: Proceedings of the 4th Asian Semantic Web Conference (ASWC'09), pp 306–20.
29. Suntisrivaraporn B (2008) Module extraction and incremental classification: A pragmatic approach for $\mathcal{EL}^+$ ontologies. In: Bechhofer S, Hauswirth M, Hoffmann J, Koubarakis M (eds) *Proceedings of the 5th European Semantic Web Conference (ESWC'08)*, Springer-Verlag, vol 5021 of *Lecture Notes in Computer Science*, pp 230–44.
30. Spackman K, Dionne R, Mays E, Weis J (2002) Role grouping as an extension to the Description Logic of Ontylog, motivated by concept modeling in SNOMED. In: Proceedings of the 2002 AMIA Annual Symposium, Hanley&Belfus, pp 712–6.
31. Reiter R (1987) A theory of diagnosis from first principles. *Artif Intell* **32**, 57–95.
32. Greiner R, Smith B, Wilkerson R (1989) A correction to the algorithm in Reiter's theory of diagnosis. *Artif Intell* **41**, 79–88.
33. Baader F, Hollunder B (1995) Embedding defaults into terminological knowledge representation formalisms. *Journal of Automated Reasoning* **14**, 149–80.
34. Seidenberg J, Rector A (2006) Web ontology segmentation: Analysis, classification and use. In: Proceedings of the 15th International Conference on World Wide Web (WWW-2006), ACM, pp 13–22.
35. Noy N, Musen M (2003) The PROMPT suite: Interactive tools for ontology mapping and merging. *International Journal of Human Computer Studies* **59**, 983–1024.
36. Cuenca Grau B, Horrocks I, Kazakov Y, Sattler U (2007) Just the right amount: Extracting modules from ontologies. In: Proceedings of the 16th International Conference on World Wide Web (WWW'07), ACM, Banff, Canada, pp 717–26.
37. Suntisrivaraporn B, Qi G, Ji Q, Haase P (2008) A modularization-based approach to finding all justifications for OWL DL entailments. In: Proceedings of the 3th Asian Semantic Web Conference (ASWC'08), Lecture Notes in Computer Science, Springer-Verlag, pp 1–15.
38. Cuenca Grau B, Horrocks I, Kazakov Y, Sattler U (2008) Modular reuse of ontologies: Theory and practice. *J Artif Intell Res* **31**, 273–318.
39. Sattler U, Schneider T, Zakharyaschev M (2009) Which kind of module should i extract? In: Proceedings of the 2009 International Workshop on Description Logics (DL'09), CEUR-WS.
40. Nortje R, Britz A, Meyer T (2009) Finding $\mathcal{EL}^+$ justifications using the Earley parsing algorithm. In: Proceedings of the 5th Australasian Ontology Workshop (AOW'09), CRPIT.
41. Nortje R, Britz K, Meyer T (2011) Bidirectional reachability-based modules. In: Proceedings of the 24th International Workshop on Description Logics (DL'11), CEUR-WS.
42. Bail S, Horridge M, Parsia B, Sattler U (2011) The justificatory structure of the NCBO BioPortal ontologies. In: Proceedings of the 10th International Semantic Web Conference (ISWC'11), Springer, pp 67–82.