Axiom Pinpointing in Large $\mathcal{EL}^+$ Ontologies via SAT and SMT Techniques

Roberto Sebastiani and Michele Vescovi

April 23, 2015

Technical Report # DISI-15-010  Version 1.0

Under submission to Journal on Artificial Intelligence Research.
Axiom Pinpointing in Large $\mathcal{EL}^+$ Ontologies via SAT and SMT Techniques

Roberto Sebastiani  
roberto.sebastiani@disi.unitn.it

Michele Vescovi  
michele.vescovi@disi.unitn.it

DISI, Università di Trento  
Via Sommarive 5, I-38123, Povo, Trento, Italy

Abstract

The quest for tractable logic-based languages arising from the field of bio-medical ontologies has raised a lot of attention on lightweight (i.e. less expressive but tractable) description logics, like $\mathcal{EL}$ and its family. To this extent, automated reasoning techniques in these logics have been developed for computing not only concept subsumptions, but also to pinpoint the (minimal) sets of axioms causing each subsumption. This task, called axiom pinpointing, allows the user for debugging possibly-huge ontologies, by identifying the minimal subsets of axioms in the ontology which cause undesired inferences.

In this work we present a novel approach for axiom pinpointing in the logic $\mathcal{EL}^+$ and its sub-logics. In a nutshell, the basic idea is to first encode the full classification of the input ontology $T$ into a polynomial-size Horn propositional formula $\phi^T_{\mathcal{EL}^+}$; then, for each subsumption inference $\alpha_i$ under analysis, a sequence of sets of literals of $\phi^T_{\mathcal{EL}^+}$ corresponding to the complete sequence of minimal sets of axioms in the original ontology $T$ from which $\alpha_i$ can be inferred—is enumerated, by exploiting an ad-hoc combination of modern SAT and SMT techniques.

We have implemented this process into a tool, $\mathcal{EL}^+$SAT, and we show in an extensive empirical evaluation that $\mathcal{EL}^+$SAT is extremely efficient in exhaustively enumerating the minimal sets of axioms, even with huge medical ontologies like SNOMED-CT.

1. Introduction

Motivations. In contrast to the trend of the previous two decades (see e.g. Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2003), in which the research in Description Logic ($\mathcal{DL}$) has focused on investigating increasingly expressive logics, the more recent quest for tractable logic-based languages arising from the field of bio-medical ontologies and Semantic Web has attracted in the last years a lot of attention on lightweight (i.e. less expressive but tractable) Description Logics, like $\mathcal{EL}$ and its family (Baader, Brandt, & Lutz, 2005; Baader, Lutz, & Suntisrivaraporn, 2006b; Baader, Peñaloza, & Suntisrivaraporn, 2007; Konev, Walther, & Wolter, 2008c; Bienvenu, 2008; Lutz, Tomán, & Wolter, 2009; Peñaloza & Sertkaya, 2010b). $\mathcal{EL}$ allows for conjunctions, existential restrictions and supports TBoxes with general concept inclusions; the logic $\mathcal{EL}^+$ (Baader et al., 2005, 2006b, 2007) extends $\mathcal{EL}$ by adding also complex role-inclusion axioms.

$\mathcal{EL}^+$ is of particular relevance due to its algorithmic properties and due to its capability of expressing several important and widely-used bio-medical ontologies, such as SNOMED-CT (Spackman et al., 1997; Spackman, 2000; Suntisrivaraporn et al., 2007), NCI (Sioutos et al., 2007), GENEONTOLOGY (The G. O. Consortium, 2000) and the major-

©2010 AI Access Foundation. All rights reserved.
ity of Galen (Rector & Horrocks, 1997). For example, according to IHTSDO\textsuperscript{1}, SNOMED-CT\textsuperscript{2} “is the most comprehensive, multilingual clinical terminology in the world. The use of a standard clinical terminology makes a significant contribution towards improving the quality and safety of healthcare by enabling consistent representation of meaning in an electronic health record.” Notice that some of these ontologies are very large (e.g., SNOMED-CT has more than 300,000 axioms).

Unfortunately, often large ontologies contain incongruences, that is, they may allow for inferring undesired logical implications, which are not coherent with the domain knowledge the ontology aims at representing. For instance, in previous versions of SNOMED-CT, the concept “Amputation of finger” could be established to imply “Amputation of upper limb” (see Baader & Suntisrivaraporn, 2008). (This bug was eventually fixed in later editions of the ontology.) Obviously the presence of such incongruences can be a very-serious problem in actual applications of ontologies like SNOMED-CT. (E.g., consider a hypothetical legal controversy between an insurance company and a finger amputee who, in principle, could claim the much-higher reimbursement for an upper-limb amputation.) It is thus of primary importance to have automated devices able not only to efficiently derive implications between concepts, but also to precisely and efficiently identify the erroneous sets of axioms causing an undesired implication.

**The Problem.** Given an $\mathcal{EL}^+$ ontology $T$, concept subsumption is the problem of deciding if a concept $C$ is a subconcept of another concept $D$ in $T$ (e.g., “is Amputation-of-Finger a subconcept of Amputation-of-Upper-Limb in SNOMED-CT?”). Classification is the problem of inferring all the subsumptions among atomic concepts which can be derived from the axioms. Both concept subsumption and classification can be done, e.g., by inferring from $T$ the subsumption(s) in the form $C \sqsubseteq D$ by means of applying ad-hoc completion rules (Baader & Suntisrivaraporn, 2008). Notice that, once the classification is computed, concept subsumption reduces to checking if a subsumption is contained in the classification.

**Axiom pinpointing** is the problem of finding one (respectively some or all) minimal subset of axioms in $T$ from which $C \sqsubseteq D$ can be inferred (e.g., “Find a minimal set of axioms in SNOMED-CT which are responsible for the fact that Amputation-of-Finger is established to be a subconcept of Amputation-of-Upper-Limb?”; Baader & Suntisrivaraporn, 2008). Following Baader and colleagues, we call these sets “MinA’s (‘Minimal Axiom sets’). More informally, axiom pinpointing is the problem of finding one (respectively some or all) minimal explanation(s) for a (typically undesired) concept-subsumption relation which can be derived from the ontology. Importantly, in $\mathcal{EL}^+$ not only concept subsumption and classification, but also the problem of finding one MinA are tractable (Baader et al., 2007; Peñaloza & Sertkaya, 2010b). Therefore, axiom pinpointing in $\mathcal{EL}^+$ is used for debugging complex bio-medical ontologies, like those mentioned above (see, e.g., Baader, Lutz, & Suntisrivaraporn, 2006a; Baader & Suntisrivaraporn, 2008; Suntisrivaraporn, 2009).

**Goals and Proposed Approach.** We start noticing that the problem of axiom pinpointing in $\mathcal{EL}^+$ is characterized by the simplicity of the logical constructors of $\mathcal{EL}^+$ on the one hand, and by the potential huge dimensions of the ontologies on the other hand. This suggests that SAT-based techniques should be natural candidates to approach this problem.

\footnotesize{\textsuperscript{1}International Health Terminology Standards Development Organization, http://www.ihtsdo.org/.
\textsuperscript{2}Systematized NOmenclature of MEDicine - Clinical Terms.}
In this work—which was presented in a preliminary form at CADE-22 conference (Sebastiani & Vescovi, 2009b)—we build on previous work from the literature of \(\mathcal{EL}^+\) reasoning (Baader et al., 2006b, 2007; Baader & Suntisrivaraporn, 2008) and of SAT and SMT (see e.g. Biere, Heule, van Maaren, & Walsh, 2009; Barrett, Sebastiani, Seshia, & Tinelli, 2009), and we propose a novel approach to axiom pinpointing in \(\mathcal{EL}^+\) and its sub-logics, which exploits an ad-hoc combination of SAT and SMT techniques.

Our idea is to first encode the full classification of the input ontology \(T\) into a polynomial-size Horn propositional formula \(\phi^\text{all}_T\), representing the deduction steps performed by the classification algorithms proposed by Baader et al. (2006b, 2007). \(\phi^\text{all}_T\) is such that

- atomic propositions label concepts and inferred subsumption relations from \(T\), and
- Horn clauses encode subsumption relations and applications of completion rules.

(Importantly, \(\phi^\text{all}_T\) is computed only once.) Then concept subsumption and, more interestingly, axiom pinpointing is directly performed by Boolean reasoning steps on \(\phi^\text{all}_T\):

(i) Concept-subsumption checks are performed by simply asserting axiom-labelling propositions and by applying standard Boolean Constraint Propagation (BCP) (Moskewicz, Madigan, Zhao, Zhang, & Malik, 2001), on \(\phi^\text{all}_T\), exploiting the SAT under assumptions technique by Eén and Sörensson (2004).

(ii) For each subsumption relation \(a_i\) derivable from \(T\), (the labelling propositions of) one minimal subset of axioms in \(T\) which allow to derive \(a_i\) (a MinA) is computed very efficiently by exploiting conflict analysis under assumptions (Moskewicz et al., 2001; Eén & Sörensson, 2004), using an ad-hoc technique which is inspired by those used for the extraction of unsatisfiable cores in SAT and in SMT (e.g. Lynce & Silva, 2004; Cimatti, Griggio, & Sebastiani, 2011).

(iii) The enumeration of some or all MinAs for \(a_i\) is performed efficiently by an ad-hoc technique which is inspired to the All-SMT approach of (Lahiri, Nieuwenhuis, & Oliveras, 2006), where the ontology \(T\) plus other information is implicitly used as background “theory”.

Notice that (iii) requires building a polynomial-size formula \(\phi^\text{all}_T\), in contrast to the exponential-size formula required by the process described in (Baader et al., 2007).

In the short version of the paper (Sebastiani & Vescovi, 2009b), we implemented the three functionalities above into a tool called \(\mathcal{EL}^+\)SAT and performed a preliminary empirical evaluation on the available ontologies. The results showed empirically that, thanks to the exploitation of the techniques implemented inside a modern SAT solver, concept subsumption (i) and one-MinA discovery (ii) are both performed instantaneously even with huge ontologies, confirming thus the potential of our novel approach. Unfortunately, the performances in the full enumeration of MinAs (step (iii)) were far from satisfactory yet.

In this paper we have refined the techniques for single-MinA pinpointing (ii) and, more importantly, we have aggressively attacked the problem of full enumeration of MinAs (iii).

First, we have developed and implemented a new SAT-based modularization technique—which we call **Cone-of-influence (COI) Modularization** because it is inspired to the cone-of-influence reduction used in Model Checking (see e.g. Clarke, Grumberg, & Peled, 1999)—
computing instantaneously from $\phi_T^{all}$ the subset of \( T \) the subset of (the labeling propositions of) all the axioms which are potentially relevant for the inference of a goal subsumption \( a_i \). This aims at drastically reducing the search space in the enumeration of MinAs. We also show both theoretically and empirically that the axiom subset obtained by COI modularization is always a subset of that computed by the modularization technique introduced by Baader & Suntisrivaraporn, 2008; Suntisrivaraporn, 2009.

Second, we have developed and implemented a SMT-like theory propagation technique (see e.g., Sebastiani, 2007) which allows for discovering the MinAs much earlier.\(^3\)

Third, we have decomposed All-MinA enumeration into a three-phase process: (a) compute the subset \( T' \) of all potentially-relevant axioms by COI modularization; (b) compute from scratch from \( T' \) the new formula $\phi_T^{all}$; (c) perform the All-MinA enumeration (iii) on $\phi_T^{all}$ instead of on $\phi_T^{all}$. Notice that typically \( T' \) and $\phi_T^{all}$ are order of magnitude smaller than \( T \) and $\phi_T^{all}$ respectively, so that step (b) is immediate and hence All-MinA enumeration requires exploring a dramatically-smaller search space.

We have run a very-extensive empirical evaluation of $\mathcal{EL}^+\text{SAT}$ on the four ontologies above, comparing the various versions of $\mathcal{EL}^+\text{SAT}$ with one another and with the state-of-the-art tool CEL (Baader et al., 2006a), showing that the renewed version of $\mathcal{EL}^+\text{SAT}$ is extremely efficient in exhaustively enumerating all MinAs, even with huge ontologies.

Content. The rest of the paper is organized as follows. In Section 2 we give the necessary background for the comprehension of this work; in particular, in Section 2.1 we introduce the description logic $\mathcal{EL}^+$ and we present the previous approach to classification, modularization and axiom pinpointing in the logic $\mathcal{EL}^+$; in Section 2.2 we provide the necessary notions about SAT and All-SMT techniques. In Section 3 we present the basic version of our SAT-/All-SMT-based procedures for concept subsumption and axiom pinpointing (both for extracting one or for enumerating all the MinAs). In Section 4 we introduce important improvements to our techniques for enumerating all the MinAs. In Section 5 we present an extensive empirical evaluation of our MinA-enumeration techniques. In Section 6 we survey the main related work. Finally, in Section 7 we summarize the main contributions of this work and we draw our conclusions.

For conciseness and readability, we have moved the proofs of all the theoretical results into Appendix A and the less-prominent part of our empirical results into Appendix B. Note for reviewers. This paper is quite long, mostly due to the following facts.

First, we have included a very long background section (Section 2). This is due to the fact that this work bridges the expertise of the DL and SAT/SMT communities, and we cannot assume that one reader from one community has also the necessary background in the other. (However, a reader can easily skip a part which is already on his/her expertise.)

Second, to improve comprehension, the paper contains more than 20 incremental examples, which help introducing progressively the main concepts and algorithms.

Third, for the reviewers’ convenience, we have included a very long appendix (A and B), containing the proofs of the theorems, and 10 pages of tables of empirical results (which are summarized in the main sections). If this paper is accepted, this appendix will be moved into an online appendix, drastically reducing the size of the camera-ready version.

\(^3\) The main ideas about theory propagation were also suggested in (Sebastiani & Vescovi, 2009b), but they were not implemented there.
2. Background

In this section we provide the background knowledge on the reasoning in the description logic $\mathcal{EL}^+$ (Section 2.1) and on the Boolean reasoning techniques (Section 2.2) which is necessary for the full understanding of the paper.

2.1 Classification, Subsumption and Axiom Pinpointing in $\mathcal{EL}^+$

We overview the main notions concerning the logic $\mathcal{EL}^+$ and the problem of solving concept subsumption, classification, and axiom pinpointing in it.

2.1.1 The Logic $\mathcal{EL}^+$.

The description logic $\mathcal{EL}^+$ belongs to the $\mathcal{EL}$ family, a group of lightweight description logics which allow for conjunctions, existential restrictions and support general concept inclusions’ TBoxes (Baader et al., 2006b); $\mathcal{EL}^+$ extends $\mathcal{EL}$ adding complex role inclusion axioms. In more details, the concept descriptions in $\mathcal{EL}^+$ are inductively defined through the constructors listed in the upper part of Table 1 at p. 7, starting from a set of primitive concepts and a set of primitive roles. (We use the uppercase letters $X, X_i, Y, Y_i$ to denote generic concepts, the uppercase letters $C, C_i, D, D_i, E, E_i$ to denote concept names and the lowercase letters $r, r_i, s$ to denote role names.)

An $\mathcal{EL}^+$ TBox (or ontology) is a finite set of general concept inclusion (GCI) and role inclusion (RI) axioms as defined in the lower part of Table 1 at p. 7. If $\mathcal{A}$ is a generic set of concept inclusions, we denote by $|\mathcal{A}|$ (the size of $\mathcal{A}$) the number of axioms in $\mathcal{A}$. Given a TBox $\mathcal{T}$, we denote with $\text{PC}_T$ the set of the primitive concepts for $\mathcal{T}$, i.e. the smallest set of concepts containing: (i) the top concept $\top$, and (ii) all concept names used in $\mathcal{T}$. We denote with $\text{PR}_T$ the set of the primitive roles for $\mathcal{T}$, i.e. the set of all the role names used in $\mathcal{T}$. We use the expression $X \equiv Y$ as an abbreviation of the two GCIs $X \sqsubseteq Y$ and $Y \sqsubseteq X$.

The peculiarity of $\mathcal{EL}^+$ w.r.t. its sub-logic $\mathcal{EL}$ is that it allows for complex role inclusion axioms with composition. This is of particular relevance because it can be used to express not only role hierarchy (e.g., $r \sqsubseteq s$) but also important role properties such us transitivity (e.g., $r \circ r \sqsubseteq r$) and right or left-identity (e.g., respectively $r \circ s \sqsubseteq r$ or $s \circ r \sqsubseteq r$). With a small abuse of notation, in the examples we represent role inclusions with the symbol $\sqsubseteq_r$, in order to better distinguish RIs from GCIs axioms.

Formally, the signature of a concept [resp. role, axiom, axiom set], denoted with signature(), is the set of all the concept and role names occurring in the description of the concept [resp. role, axiom, axiom set]. In particular the signature of a TBox $\mathcal{T}$, signature($\mathcal{T}$), is the set of concept and role names occurring in $\mathcal{T}$: signature($\mathcal{T}$) = $\text{PC}_T \cup \text{PR}_T$.
Example 2.1. We consider the following small $\mathcal{EL}^+$ ontology in the medical domain adapted from the one proposed by Suntisrivaraporn (2009), which we also call $O_{med}$. 4

\[
\begin{align*}
\text{Appendix} & \sqsubseteq \text{BodyPart} \sqcap \exists \text{partOf}. \text{Intestine} & a_1 \\
\text{Endocardium} & \sqsubseteq \text{Tissue} \sqcap \exists \text{partOf}. \text{HeartValve} & a_2 \\
\text{Pericardium} & \sqsubseteq \text{Tissue} \sqcap \exists \text{containsIn}. \text{Heart} & a_3 \\
\text{Appendicitis} & \sqsubseteq \text{Inflammation} \sqcap \exists \text{hasLocation}. \text{Appendix} & a_4 \\
\text{Endocarditis} & \sqsubseteq \text{Inflammation} \sqcap \exists \text{hasLocation}. \text{Endocardium} & a_5 \\
\text{Pericarditis} & \sqsubseteq \text{Inflammation} \sqcap \exists \text{hasLocation}. \text{Pericardium} & a_6 \\
\text{Inflammation} & \sqsubseteq \text{Disease} \sqcap \exists \text{actsOn}. \text{Tissue} & a_7 \\
\text{Disease} \sqcap \exists \text{hasLocation}. \text{Heart} & \sqsubseteq \text{HeartDisease} & a_8 \\
\text{HeartDisease} & \sqsubseteq \exists \text{hasState}. \text{NeedsTreatement} & a_9 \\
\text{partOf} \circ \text{partOf} & \sqsubseteq_r \text{partOf} & a_{10} \\
\text{hasLocation} \circ \text{containsIn} & \sqsubseteq_r \text{hasLocation} & a_{11}
\end{align*}
\]

The signature of the whole ontology $O_{med}$ is $\text{signature}(O_{med}) = \{\text{Appendix, BodyPart, Intestine, Endocardium, Tissue, HeartValve, Pericardium, Heart, Appendicitis, Inflammation, Endocarditis, Pericarditis, Disease, HeartDisease, NeedsTreatement}\} \cup \{\text{partOf, containsIn, actsOn, hasLocation, hasState}\}$.

The ontology expresses the relations between some kinds of inflammatory disease and body parts or their states. Notice that the ontology is made of nine GCI s and two RIs. In particular the RI axiom $a_{10}$ express the transitivity of partOf while $a_{11}$ is a right-identity which express the fact that everything that has location in some body part which is contained in a more general body part is located also in that second body part.  

The semantics of $\mathcal{EL}^+$ is defined in terms of interpretations. An interpretation $I$ is a couple $I = (\Delta^I, \mathcal{I})$, where $\Delta^I$ is the domain, i.e. a non-empty set of individuals, and $\mathcal{I}$ is the interpretation function which maps each concept name $C$ to a set $C^I \subseteq \Delta^I$ and maps each role name $r$ to a binary relation $r^I \subseteq \Delta^I \times \Delta^I$. In the right-most column of Table 1 at p. 7 the inductive extensions of $\mathcal{I}$ to arbitrary concept descriptions are defined. An interpretation $I$ is a model of a given TBox $T$ if and only if the conditions in the Semantics column of Table 1 at p. 7 are respected for every GCI and RI axiom in $T$. A TBox $T'$ is a conservative extension of the TBox $T$ if every model of $T'$ is also a model of $T$, and every model of $T$ can be extended to a model of $T'$ by appropriately defining the interpretations of the additional concept and role names.

Given the concepts $X$ and $Y$, $Y$ subsumes $X$ w.r.t. the TBox $T$, written $X \sqsubseteq_T Y$ (or simply $X \sqsubseteq Y$ when it is clear to which TBox we refer to), iff $X^I \subseteq Y^I$ for every model $I$ of $T$. The computation of all subsumption relations between concept names occurring in $T$ is called classification of $T$. Concept subsumption and classification in $\mathcal{EL}^+$ can be performed in polynomial time (Baader et al., 2005, 2007). In particular, Baader et al. (2005, 2007) solved the problem of classifying an $\mathcal{EL}^+$ TBox as a subcase of the polynomial-time

---

4. We stress that all the sample ontologies included in this work are used only to clarify some points of the exposition. We neither care they are completely inclusive of their specific domains nor we care care about the correctness of these (sub)ontologies, both from the medical and the ontology-design perspective.
Axiom Pinpointing in large $\mathcal{EL}^+$ ontologies via SAT and SMT techniques

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>top</td>
<td>$\top$</td>
</tr>
<tr>
<td>conjunction</td>
<td>$X \cap Y$</td>
</tr>
<tr>
<td>existential restriction</td>
<td>$\exists r.X$</td>
</tr>
<tr>
<td>general concept inclusion</td>
<td>$X \sqsubseteq Y$</td>
</tr>
<tr>
<td>role inclusion</td>
<td>$r_1 \circ \ldots \circ r_n \sqsubseteq s$</td>
</tr>
</tbody>
</table>

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

algorithm for concept subsumption in which all the possible concept subsumptions in the TBox are deduced.

2.1.2 Normalization.
In $\mathcal{EL}^+$ it is convenient to establish and work with a normal form of the input problem, which helps to make explanations, proofs, reasoning rules and algorithms simpler and more general. Usually the following normal form for the $\mathcal{EL}^+$ TBoxes is considered (Baader et al., 2005, 2006b; Baader & Peñaloza, 2007; Baader et al., 2007):

$$
(C_1 \cap \ldots \cap C_k) \sqsubseteq D \quad \quad \quad k \geq 1 \\
C \sqsubseteq \exists r.D \\
\exists r.C \sqsubseteq D \\
r_1 \circ \ldots \circ r_n \sqsubseteq s 
$$

such that $C_1, \ldots, C_k, D \in \mathcal{PC}_T$ and $r_1, \ldots, r_n, s \in \mathcal{PR}_T$. A TBox $T$ can be turned into a normalized TBox $T'$ that is a conservative extension of $T$ (Baader et al., 2005), by introducing new concept names. In a nutshell, normalization consists in rewriting axioms in the form $C \sqsubseteq C_1 \cap \ldots \cap C_n$ into $C \sqsubseteq C_1, \ldots, C \sqsubseteq C_n$, and in substituting, when needed, instances of complex concepts of the forms $\exists r.C$ and $C_1 \cap \ldots \cap C_k$ with fresh concept names (namely, $C'$ and $C''$), adding the axioms $C' \sqsubseteq \exists r.C$ [resp. $\exists r.C \sqsubseteq C'$] and $C'' \sqsubseteq C_1, \ldots, C'' \sqsubseteq C_k$ [resp. $(C_1 \cap \ldots \cap C_k) \sqsubseteq C''$] for every complex concept substituted in the right [resp. left] part of an axiom. The normal TBox $T'$ resulting from this process is composed of two kinds of axioms:

- some top-level axioms representing the original axioms of $T$ (with complex sub-concepts substituted by the fresh concept names);
- some definition axioms representing the labeling of complex sub-concepts with the newly introduced concept names.

If a complex concept appears on the same side (left or right) of different axioms only one definition axiom is necessary.

Normalization can be performed in linear time w.r.t. the size of $T$, and the size of $T'$ is linear w.r.t. that of $T$ (Baader et al., 2005). We call normal concept of a normal TBox $T'$ every non-conjunctive concept description occurring in the concept inclusions of $T'$; we call $\mathcal{NC}_{T'}$ the set of all the normal concepts of $T'$. (I.e., the set $\mathcal{NC}_{T'}$ consists in all the concepts of the form $C$ or $\exists r.C$, with $C \in \mathcal{PC}_{T'}$ and $r \in \mathcal{PR}_{T'}$.)
Example 2.2. The following set of axioms, that we call $O_{milk}$, is adapted from a fragment of the ontology not-Galen, and represents some facts and relationships concerning the concept Milk:

\[
\begin{align*}
\text{BodyFluid} & \sqsubseteq \text{Fluid} & m_1 \\
\text{Liquid} & \sqsubseteq \text{Fluid} & m_2 \\
\text{BodyFluid} & \equiv \text{BodySubstance} \sqcap \exists \text{hasPhysicalState.liquidState} & m_3 \\
\text{BodySubstance} & \sqsubseteq \text{Substance} & m_4 \\
\text{Milk} & \sqsubseteq \text{BodySubstance} & m_5 \\
\text{Milk} & \sqsubseteq \exists \text{hasPhysicalState.liquidState} & m_6 \\
\text{Milk} & \sqsubseteq \exists \text{isActedOnSpecificallyBy.\{Secretion \sqcap \exists \text{isFunctionOf.Breast}\}} & m_7 \\
\text{SecretedSubstance} & \equiv \text{Substance} \sqcap \exists \text{isActedOnBy.Secretion} & m_8 \\
\text{Liquid} & \equiv \text{Substance} \sqcap \exists \text{hasPhysicalState.liquidState} & m_9 \\
\text{liquidState} & \equiv \text{PhysicalState} \sqcap \exists \text{hasState.liquid} & m_{10} \\
\text{isActedOnSpecificallyBy} & \sqsubseteq \text{isActedOnBy} & m_{11}
\end{align*}
\]

We consider the normalization of axioms $m_3$ and $m_7$. Since $m_3$ is an equivalence, it is split into a couple of inclusion axioms:

\[
\begin{align*}
\text{BodyFluid} & \sqsubseteq \text{BodySubstance} \sqcap \exists \text{hasPhysicalState.liquidState} & m_{3a} \\
\text{BodySubstance} \sqcap \exists \text{hasPhysicalState.liquidState} & \sqsubseteq \text{BodyFluid} & m_{3b}
\end{align*}
\]

Then we split $m_{3a}$ into two distinct axioms and introduce a primitive concept $N$ labeling $\exists \text{hasPhysicalState.liquidState}$. The resulting normalization is:

\[
\begin{align*}
\text{BodyFluid} & \sqsubseteq \text{BodySubstance} \\
\text{BodyFluid} & \sqsubseteq N \\
N & \sqsubseteq \exists \text{hasPhysicalState.liquidState} \\
\text{BodySubstance} \sqcap N & \sqsubseteq \text{BodyFluid} \\
\exists \text{hasPhysicalState.liquidState} & \sqsubseteq N.
\end{align*}
\]

The first, second and fourth axioms are top-level ones, whilst the third and fifth are definitions of $N$.

The normalization of $m_7$ requires the introduction of another fresh concept name $M$ labeling $(\text{Secretion} \sqcap \exists \text{isFunctionOf.Breast})$. The definition of $M$ is then split into two axioms:

\[
\begin{align*}
\text{Milk} & \sqsubseteq \exists \text{isActedOnSpecificallyBy.M} \\
M & \sqsubseteq \text{Secretion} \\
M & \sqsubseteq \exists \text{isFunctionOf.Breast}.
\end{align*}
\]

\[\diamondsuit\]
Table 2: Completion rules of the concept subsumption algorithm for EL+. A rule reads as follows: if the assertions/axioms in the left column belong to $A$, the GCI/RI of the central column belongs to $T$, and the assertion of the right column is not already in $A$, then the assertion of the right column is added to $A$.

<table>
<thead>
<tr>
<th>Subsumption assertions ($\ldots \in A$)</th>
<th>TBox’s axioms ($\ldots \in T$)</th>
<th>... added to $A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X \sqsubseteq C_1$, $X \sqsubseteq C_2$, \ldots, $X \sqsubseteq C_k$, $k \geq 1$</td>
<td>$C_1 \land \cdots \land C_k \sqsubseteq D$</td>
<td>$X \sqsubseteq D$</td>
</tr>
<tr>
<td>$X \sqsubseteq C$</td>
<td>$C \sqsubseteq \exists r. D$</td>
<td>$X \sqsubseteq \exists r. D$</td>
</tr>
<tr>
<td>$X \sqsubseteq \exists r. E$, $E \sqsubseteq C$</td>
<td>$\exists r. C \sqsubseteq D$</td>
<td>$X \sqsubseteq D$</td>
</tr>
<tr>
<td>$X \sqsubseteq \exists r. D$</td>
<td>$r \sqsubseteq s$</td>
<td>$X \sqsubseteq \exists s. D$</td>
</tr>
<tr>
<td>$X \sqsubseteq \exists r_1. E_1$, \ldots, $E_{n-1} \sqsubseteq \exists r_n. D$, $n \geq 2$</td>
<td>$r_1 \circ \cdots \circ r_n \sqsubseteq s$</td>
<td>$X \sqsubseteq \exists s. D$</td>
</tr>
</tbody>
</table>

2.1.3 Concept Subsumption in EL+.

Given a normalized TBox $T$ over the set of primitive concepts $PC_T$ and the set of primitive roles $PR_T$, the subsumption algorithm for EL+ proposed by Baader et al. (2007) generates and extends a set $A$ of assertions through the completion rules defined in Table 2 at p. 9. By “assertion” we mean every known or deduced subsumption relation between normal concepts of the TBox $T$. The algorithm starts with the initial set $A = \{a_i \in T \mid a_i$ is a GCI $\} \cup \{ C \sqsubseteq C, C \sqsubseteq \top \mid C \in PC_T \}$ and extends $A$ using the rules of Table 2 at p. 9, until no more assertions can be added. (Notice that a rule is applied only if it extends $A$.) We call propositional completion rules the first two completion rules in Table 2 at p. 9, and non-propositional completion rules the other three rules.

Example 2.3. We report all the subsumption relations that can be inferred in $O_{med}$ (Example 2.1 at p. 5) from its original axioms. Once $O_{med}$ is turned into normal form its full classification $A_{O_{med}}$ is the following:

$$\begin{align*}
\text{Appendix} & \sqsubseteq \text{BodyPart} & a'_1 & \text{hasLoc.Heart} \sqsubseteq \text{New} & a_0 \\
\text{Endocardium} & \sqsubseteq \text{Tissue} & a'_2 & \exists \text{partOf.Intestine} & a''_1 \\
\text{Pericardium} & \sqsubseteq \text{Tissue} & a'_3 & \exists \text{partOf.HeartValve} & a''_2 \\
\text{Appendicitis} & \sqsubseteq \text{Inflammation} & a'_4 & \exists \text{containedIn.Heart} & a''_3 \\
\text{Endocarditis} & \sqsubseteq \text{Inflammation} & a'_5 & \exists \text{hasLoc.Endocardium} & a''_4 \\
\text{Pericarditis} & \sqsubseteq \text{Inflammation} & a'_6 & \exists \text{hasLoc.Pericardium} & a''_5 \\
\text{Inflammation} & \sqsubseteq \text{Disease} & a'_7 & \exists \text{actsOn.Tissue} & a''_6 \\
\text{Disease} & \sqsubseteq \text{New} & a'_8 & \exists \text{hasState.NeedsTreat.} & a_9 \\
\text{HeartDisease} & \sqsubseteq \exists \text{hasState.NeedsTreat.} & a_9 & b_1 & r_1(a'_4, a'_7) \\
\text{Appendicitis} & \exists \text{Disease} & b_2 & r_2(a'_4, a''_7) \\
\text{Endocarditis} & \exists \text{Disease} & b_3 & r_1(a'_5, a'_7) \\
\text{Pericarditis} & \exists \text{Disease} & b_5 & r_1(a'_6, a'_7) \\
\text{Pericarditis} & \exists \text{hasLoc.Heart} & b_7 & r_2(a'_6, a''_7, a_{11}) \\
\text{Pericarditis} & \exists \text{HeartDisease} & b_9 & r_1(b_5, b_6, a'_8) \\
\end{align*}$$

In particular, the first seventeen GCIs ($a'_1, a''_1, \ldots, a'_8, a_0, a_9$ in the first nine rows) plus the two RIs $a_{10} \overset{\text{def}}{=} \text{partOf} \circ \text{partOf} \sqsubseteq \text{partOf}$ and $a_{11} \overset{\text{def}}{=} \text{hasLocation} \circ \text{containedIn} \sqsubseteq \text{hasLocation}$ compose the normalization of $O_{med}$. We labeled $a'_1$ (and $a''_1$) the normal-form top-level
Consider the subsumption relation $C \sqsubseteq_D D_i$, with $C_i, D_i \in \mathcal{PC}_T$. If $C_i \sqsubseteq_S D_i$ for some set $S \subseteq T$ of axioms, then $S$ is called an axiom set (nMinA) for $C_i \sqsubseteq D_i$ w.r.t. $T$. If $C_i \nsubseteq D_i$ for every $S' \subseteq S$, then $S$ is called a minimal axiom set (MinA) for $C_i \subseteq D_i$ w.r.t. $T$.

Example 2.4. In the ontology $O_{milk}$ of Example 2.2 at p. 8, a MinA for $\text{Milk} \sqsubseteq \text{SecretedSubstance}$ w.r.t. $O_{milk}$ is given by the original axioms $\{m_4, m_5, m_7, m_8, m_{11}\}$.\footnote{This is the only MinA for this subsumption in $O_{milk}$.} In particular, $m_4$ and $m_5$ are necessary to infer $\text{Milk} \sqsubseteq \text{Substance}$, while from (the normalization of) $m_7$ and $m_{11}$, it follows that $\text{Milk} \sqsubseteq \exists \text{isActedOnBy} . \text{Secretion}$. Finally, $\text{Milk} \sqsubseteq \text{SecretedSubstance}$ can be inferred from the two previous premises and the definition of $\text{SecretedSubstance}$ in $m_8$.\footnote{A monotone propositional formula is a propositional formula whose only connectives are $\land$ and $\lor$.} 

Baader et al. (2007) proposed a technique for computing all MinAs for $C_i \sqsubseteq D_i$ w.r.t. $T$: during the classification of $T$, a pinpointing formula (namely $\Phi_{C_i \sqsubseteq D_i}$) is built, which is a monotone propositional formula\footnote{A monotone propositional formula is a propositional formula whose only connectives are $\land$ and $\lor$.} on the set of variables $P_T \triangleq \{s_{[ax_j]} \mid ax_j \in T\}$ s.t., for every $O \subseteq T$, $O$ is a MinA for $C_i \sqsubseteq_D D_i$ iff $\{s_{[ax_j]} \mid ax_j \in O\}$ is a minimal valuation of $\Phi_{C_i \sqsubseteq D_i}$. In a nutshell, the process of building $\Phi_{C_i \sqsubseteq D_i}$ works as follows. Every axiom $ax_j \in T$ is encoded with a propositional variable $s_{[ax_j]}$ and every deduced assertion
Axiom Pinpointing in large $\mathcal{EL}^+$ ontologies via SAT and SMT techniques

$a_j \in \mathcal{A}$ is encoded into a monotone propositional formula. Consider each application of a completion rule $r$ during the classification of $\mathcal{T}$, and let $\phi_r$ be the conjunction of the labels (i.e., variables and monotone formulas) of the axioms and the assertions in the preconditions of $r$ (respectively). If a new assertion is deduced as consequence of $r$, then it is labeled with the formula $\phi_r$. Otherwise, if the assertion in the consequence of $r$ is already in $\mathcal{A}$ and it is labeled with $\psi$, then its label is updated with $\psi' = \psi \lor \phi_r$, unless $\phi_r \models \psi$.

The Baader et al.’s (2007) all-MinAs algorithm consists thus in (i) building $\Phi^{C_i \sqsubseteq D_i}$ and (ii) computing all minimal valuations of $\Phi^{C_i \sqsubseteq D_i}$. This algorithm, however, has serious limitations in terms of complexity: first, the algorithm for generating $\Phi^{C_i \sqsubseteq D_i}$ requires intermediate logical checks, each of them involving the solution of an NP-complete problem; second, the size of $\Phi^{C_i \sqsubseteq D_i}$ can be exponential w.r.t. that of $\mathcal{T}$ (Baader et al., 2007). More generally, Baader et al. (2007) proved also that there is no output-polynomial algorithm for computing all MinAs (unless P=NP). (To the best of our knowledge, there is no publicly-available implementation of the all-MinAs algorithm above.) Consequently, Baader et al. (2007) concentrated the effort on finding polynomial algorithms for finding one MinA at a time, proposing a linear-search minimization algorithm which allowed for finding MinAs for FULL-GALEN efficiently. This technique was further improved by Baader and Suntisriaraporn (2008) by means of a binary-search minimization algorithm, and by a novel algorithm exploiting the notion of reachability-modules, which allowed to efficiently find MinAs for the much bigger SNOMED-CT ontology. We refer the readers to the literature (Baader et al., 2007; Baader & Suntisriaraporn, 2008) for a detailed description.

Furthermore, Suntisriaraporn, 2009 solved the all-MinAs problem with a different approach based on the techniques of the Hitting Set Tree (HST), where the universal set is the whole ontology and the set of the all MinAs is collection of the minimal subsets to be found. In particular the hitting set tree is expanded along the algorithm computing, at the end, all the MinAs for the given subsumption. In this approach the optimized algorithm and the linear minimization algorithm exposed above are used as subroutines to initialize the algorithm and to minimize the resulting sets respectively. However, also this techniques has the major drawback of performance in large-scale ontologies.

Therefore, the above technique has been implemented in combination with the reachability-based module-extraction technique (Suntisriaraporn, 2009), which drastically reduces the search space of the HST algorithm. We briefly describe such technique in Section 2.1.5, referring the readers to the literature (Suntisriaraporn, 2009) for a much more detailed explanation.

2.1.5 Axiom Pinpointing with Reachability-based Modularization in $\mathcal{EL}^+$

Real-world (medical) $\mathcal{EL}^+$ ontologies are often huge in size (e.g., SNOMED-CT’09 has more than 300,000 axioms). Thus, despite the low complexity of the pinpointing problem in $\mathcal{EL}^+$, the handling of such ontologies is out of the reach of the pinpointing algorithms described in previous sections.

Luckily, for a given subsumption assertion $a_i \overset{\text{def}}{=} C_i \sqsubseteq D_i$ which can be derived from $\mathcal{T}$, it is typically the case that only a minority of the axioms in $\mathcal{T}$ may have any role in any derivation of $a_i$. Thus, a key idea is to identify a priori a strict subset $\mathcal{M}_{a_i}$ of $\mathcal{T}$ which is sufficient to perform every possible derivation of $a_i$. (Hence $\mathcal{M}_{a_i}$ contains
every MinA for \( a_i \). \( M_n \) is called a *module* and the process of producing it is called *modularization*. Different forms of modularization have been proposed in literature (Noy & Musen, 2003; Seidenberg & Rector, 2006; Grau, Horrocks, Kazakov, & Sattler, 2007; Konev, Lutz, Walther, & Wolter, 2008a, 2008b; Suntisrivaraporn, Qi, Ji, & Haase, 2008), including semantic or syntactic and logic-based or not. In particular, in order to improve the efficiency of axiom-pinpointing algorithms in \( \mathcal{EL}^+ \) described in previous sections, Baader and Suntisrivaraporn (2008) proposed the *reachability-based modularization* technique (see also Suntisrivaraporn, 2009) that has been lately extended to harder logics by Suntisrivaraporn et al. (2008).

We recall from the work of Suntisrivaraporn (2009) the basic facts about reachability-based modularization.

**Definition 2** (\( \Sigma \)-reachable symbols, axioms, reachability-based module). Given an \( \mathcal{EL}^+ \) TBox \( T \) and a signature \( \Sigma \subseteq \text{signature}(T) \), the set of the \( \Sigma \)-reachable symbols of \( T \) is recursively defined as follows:

(i) every symbol \( x \in \Sigma \) is \( \Sigma \)-reachable;

(ii) for every axiom \( \langle C \sqsubseteq D \rangle \) of \( T \), if all the symbols in \( \text{signature}(C) \) are \( \Sigma \)-reachable, then all symbols \( y \in \text{signature}(D) \) are \( \Sigma \)-reachable.

Given the set of the \( \Sigma \)-reachable symbols, an axiom \( \langle \hat{C} \sqsubseteq \hat{D} \rangle \in T \) is a \( \Sigma \)-reachable axiom of \( T \) if \( x \) is \( \Sigma \)-reachable for every symbol \( x \in \text{signature}(\hat{C}) \). The \( \Sigma \)-reachability-based module for \( \Sigma \) in \( T \), denoted by \( M^\text{reach}_\Sigma \), is the set of all the \( \Sigma \)-reachable axioms of \( T \).

(With a little abuse of notation, if \( \Sigma \) consists only of a single concept name \( C \), then we denote its reachability-based module by \( M^\text{reach}_C \) rather than by \( M^\text{reach}_\{C\} \).)

**Example 2.5.** Consider again the ontology \( O_{med} \) in Example 2.1 at p. 5. The reachability-based module for the signature \( \Sigma_{\text{Pericarditis}} \) is \( M^\text{reach}_{\text{Pericarditis}} = \{a_3, a_6, a_7, a_8, a_9, a_{11}\} \). In fact, starting from the symbol \( \text{Pericarditis} \), axiom \( a_6 \) is included in the module and the symbols \( \text{Pericarditis}, \text{Inflammation}, \text{hasLocation} \) and \( \text{Pericardium} \) are marked as \( \Sigma_{\text{Pericarditis}} \)-reachable. From \( \text{Pericardium} \) and \( \text{Inflammation} \), axioms \( a_3 \) and \( a_7 \) are included in the module and hence \( \text{Tissue}, \text{containedIn}, \text{Heart Disease}, \text{actsOn} \) are added to the \( \Sigma_{\text{Pericarditis}} \)-reachable symbols. The three left-side symbols of \( a_8 \) (i.e. \( \text{Disease}, \text{hasLocation}, \text{Heart} \)) are now \( \Sigma_{\text{Pericarditis}} \)-reachable, so that \( a_9 \) is also added to the module. Hence \( \text{HearthDisease} \) becomes \( \Sigma_{\text{Pericarditis}} \)-reachable, so that \( a_9 \) is added to the module, making \( \text{HasState}, \text{NeedsTreatment} \) \( \Sigma_{\text{Pericarditis}} \)-reachable. Moreover, since both \( \text{containedIn} \) and \( \text{hasLocation} \) are \( \Sigma_{\text{Pericarditis}} \)-reachable, then also \( a_{11} \) is added to the module. No other axiom can then be added to the module.

Intuitively, an axiom \( ax \) of \( T \) is included in the reachability-based module \( M^\text{reach}_\Sigma \) for \( \Sigma \) if and only if the symbols of \( \Sigma \) syntactically refer to the symbols in \( ax \), either directly or indirectly via other axioms of \( T \). All the axioms which are thus “syntactically connected” to \( \Sigma \) are included in the reachability-based module for \( \Sigma \).

Notice that the reachability-based modularization is a purely *syntactic* technique, because the semantic of the axioms and of the operators involved are not considered in the construction of the modules. Moreover, notice that this modularization techniques is fully independent from the completion rules used in the classification of \( T \). In the following
we will refer to this technique either calling it reachability-based modularization or (more generically) syntactic modularization.

**Property 1.** Let $\Sigma$ be a signature on the TBox $\mathcal{T}$, and let $\hat{C}, \hat{D}$ be arbitrary $\mathcal{EL}^+$ concept descriptions such that $\text{signature}(\hat{C}) \subseteq \Sigma$. Then $\hat{C} \sqsubseteq_{\mathcal{T}} \hat{D}$ if and only if $\hat{C} \sqsubseteq_{\mathcal{M}_{\text{reach}}} \hat{D}$.

Thus, for every subsumption relation $C \sqsubseteq_{\mathcal{T}} D$, the process of axiom pinpointing plus reachability-based modularization for $a_i$ consists in:

(i) computing the reachability-based module $\mathcal{M}_{\text{reach}}^C$,

(ii) applying the axiom pinpointing algorithm to $\mathcal{M}_{\text{reach}}^C$ instead than to $\mathcal{T}$.

Suntisrivaraporn (2009) computes reachability-based modules through a queue-based algorithm which iteratively adds axioms to the initially empty module, starting from the given input signature. The algorithm is shown to be quadratic w.r.t. $|\mathcal{T}|$. However, if $|\mathcal{M}_{a_i}| << |\mathcal{T}|$ (as it is often the case), then the modularization process can drastically improve the efficiency of the pinpointing process.

### 2.2 Basics on CDCL SAT Solving

For the best comprehension of the content of the next sections, we recall some notions on SAT and Conflict-Driven Clause-Learning (CDCL) SAT solving which are necessary for the comprehension of our work. For a much deeper description, and for other advanced SAT techniques which are not useful for our discourse, we refer the reader to the literature (e.g., Silva & Sakallah, 1996; Zhang & Malik, 2002; Eén & Sörensson, 2004; Lynce & Silva, 2004; Eén & Biere, 2005; Biere, 2008).

#### 2.2.1 Basics on SAT and Notation.

We assume the standard syntactic and semantic notions of propositional logic. Given a non-empty set of primitive propositions $\mathcal{P} = \{p_1, p_2, \ldots\}$, the language of propositional logic is the least set of formulas containing $\mathcal{P}$ and the primitive constants $\top$ and $\bot$ ("true" and "false") and closed under the set of standard propositional connectives $\{\neg, \land, \lor, \to, \leftrightarrow\}$.

We call a **propositional atom** (also called propositional **variable**) every primitive proposition in $\mathcal{P}$, and a **propositional literal** (positive literal) or its negation (negative literal). We implicitly remove double negations: e.g., if $l$ is the negative literal $\neg p_i$, by $\neg l$ we mean $p_i$ rather than $\neg \neg p_i$. We represent a truth assignment as a conjunction of literals $\text{\cap}_i l_i$ (or analogously as a set of literals $\{l_i\}_i$) with the intended meaning that a positive [resp. negative] literal $p_i$ means that $p_i$ is assigned to true [resp. false]. We say that $\{\mu_1, \ldots, \mu_k\}$ is a complete set of partial truth assignments satisfying $\phi$ iff (i) $\mu_i \models \phi$ for every $i$ and (ii) every total assignment $\mu$ s.t. $\mu \models \phi$ is such that $\mu_i \subseteq \mu$ for some $i$.

A propositional formula is in **conjunctive normal form**, CNF, if it is written as a conjunction of disjunctions of literals: $\text{\cap}_i \text{\lor}_j l_{ij}$. Each disjunction of literals $\text{\lor}_j l_{ij}$ is called a **clause**. Notationally, we often write clauses as implications: "$(\text{\cap}_i l_i) \to (\text{\lor}_j l_j)$" for "$\text{\lor}_i \neg l_i \lor \text{\lor}_j l_j$"; also, if $\eta$ is a conjunction of literals $\text{\cap}_i l_i$, we write $\neg \eta$ for the clause $\text{\lor}_i \neg l_i$, and vice versa.

A **unit clause** is a clause with only one literal. A **Horn clause** is a clause containing at most one positive literal, and a **Horn formula** is a conjunction of Horn clauses. Notice
SatValue DPLL (formula $\varphi$) {
  1. $\mu = \emptyset$; vars = Atoms($\varphi$);
  2. Preprocess($\varphi$, $\mu$);
  3. while (1) {
      while (1) {
        status = BCP($\varphi$, $\mu$);
        if (status == unknown) {
          if (TooManyClauses($\varphi$)) {
            DeleteSomeInactiveClauses($\varphi$);
            break;
          }
          if (status == sat) {
            return sat;
          }
          else { // status == conflict
            $\psi = \text{FalsifiedClause}(\varphi, \mu)$;
            $\psi' = \text{AnalyzeConflict}(\varphi, \mu, \psi)$;
            $\varphi = \varphi \land \psi'$; // $\psi'$ learned temporarily
            Backtrack($\psi', \mu, \varphi$);
            if (DecisionLevel($\mu$) == 0) {
              return unsat;
            }
          }
          } }
        DecideNextBranch($\varphi$, $\mu$);
      } }
  }
}

Figure 1: Schema of a CDCL DPLL SAT solver.

that Horn clauses are either unary positive clauses, or they contain at least one negative literal. A definite Horn clause is a non-unary Horn clause containing exactly one positive literal and at least one negative one, and a definite Horn formula is a conjunction of definite Horn clauses. (Intuitively, definite Horn formulas represents sets of implications between propositional variables ($\bigwedge_{i=1}^{n} p_i \rightarrow p$ s.t. $n > 0$.)

Notice that a definite Horn formula $\phi$ is always satisfiable, since it is satisfied by both the assignments $\mu_T$ and $\mu_L$ which assign all variables to true and to false respectively. Notice also that, for every subset $\{p_i\}_i$ of propositional variables in $\phi$, $\phi \land \bigwedge_{i} p_i$ and $\phi \land \bigwedge_{i} \neg p_i$ are satisfied by $\mu_T$ and $\mu_L$ respectively. Thus, in order to falsify a definite Horn formula $\phi$, it is necessary to conjoin to it at least one positive and one negative literal.

The problem of detecting the satisfiability of a propositional CNF formula, also referred as the SAT problem, is NP-complete. A SAT solver is a tool able to solve the SAT problem. The problem of detecting the satisfiability of a propositional Horn formula, also referred as the Horn-SAT problem, is polynomial.
2.2.2 CDCL SAT Solving.

Most state-of-the-art SAT procedures are evolutions of the Davis-Putnam-Longemann-Loveland (DPLL) procedure (Davis & Putnam, 1960; Davis, Longemann, & Loveland, 1962) and they are based on the CDCL paradigm (Silva & Sakallah, 1996; Zhang, Madigan, Moskewicz, & Malik, 2001). A high-level schema of a modern CDCL DPLL engine is shown in Figure 1 at p. 14. The propositional formula $\varphi$ is in CNF; the assignment $\mu$ is initially empty, and it is updated in a stack-based manner. DPLL considers all and only the Boolean variables in $\text{vars} \triangleq \text{Atoms}(\varphi)$ (if not specified otherwise).

Initially, Preprocess simplifies $\varphi$ as much as possible, and updates $\mu$ accordingly (see e.g., Eén & Biere, 2005; Biere, 2008). In the main loop, after the inner loop, DecideNextBranch chooses an unassigned literal $l$ from $\varphi$ according to some heuristic criterion, and adds it to $\mu$. (This operation is called decision, $l$ is called decision literal and the number of decision literals in $\mu$ after this operation is called the decision level of $l$.) In the inner loop, BCP iteratively deduces literals $l$ from the current assignment and updates $\varphi$ and $\mu$ accordingly; this step is repeated until either (i) $\mu$ falsifies some clause in $\varphi$, or (ii) no more Boolean variable in $\text{vars}$ can be assigned (s.t. $\mu$ satisfies $\varphi$), or (iii) no more literals can be deduced, returning conflict, sat or unknown respectively. If status is unknown, then DPLL exits the inner loop, looking for the next decision. (We temporarily ignore steps 7–8, which we will discuss later.) If status is sat, then DPLL returns sat. If status is conflict, then, starting from the falsified clause $\psi$, AnalyzeConflict detects the subset $\eta$ of $\mu$ which caused the falsification of $\psi$ (conflict set), and returns $\psi' \triangleq \neg \eta$. (This process is called conflict analysis, and is described in more details below.) $\psi'$ is then temporarily added to $\varphi$ (this process is called learning). Then Backtrack uses $\psi'$ to decide the decision level to backtrack to (namely $\text{blevel}$), and then it backtracks to $\text{blevel}$, popping out of $\mu$ all literals whose decision level is strictly greater than $\text{blevel}$, and updating $\varphi$ accordingly (this process is called backjumping). If $\text{blevel}$ is 0, then a conflict exists even without branching, so that DPLL returns unsat.

BCP is based on Boolean Constraint Propagation, that is, the iterative application of unit propagation: if a unit clause $l$ occurs in $\varphi$, then $l$ is added to $\mu$, all negative occurrences of $l$ are declared false and all clauses with positive occurrences of $l$ are declared satisfied. Current CDCL SAT solvers include extremely fast implementations of BCP based on the two-watched-literals scheme (Moskewicz et al., 2001). The two-watched-literals scheme is based on the idea that if a clause has more than two unassigned literals, then a single application of unit propagation will not produce a new unit clause. Thus, in order to efficiently detect unit clauses to be propagated, it is not necessary to visit all the clauses. This scheme maintains the property that only two different unassigned literals on each clause are watched by a pointer. When a watched literal is assigned to false, the pointer moves looking for another unassigned literal to watch; if none is found, then a new unit clause is detected. Satisfied clauses are not removed; rather, they are lazily detected and ignored when performing propagations. This scheme requires, for every literal, only the storage of its current assignment status (true, false, unassigned) and the list of the pointers to the clauses it watches (which, in this way, are immediately accessible from the literal).

Importantly, a complete run of BCP requires an amount of steps which is linear in the number of clauses containing the negation of some of the propagated literals. Also, if $\varphi$ is
a Horn formula, then one run of BCP is sufficient to decide its satisfiability: if BCP(\(\varphi, \{\}\)) returns conflict, then \(\varphi\) is unsatisfiable; otherwise \(\varphi\) is satisfiable because, since all unit clauses have been removed from \(\varphi\), all remaining clauses contain at least one negative literal, so that assigning all unassigned literals to false would satisfy \(\varphi\).

**AnalyzeConflict** works as follows (Silva & Sakallah, 1996; Moskewicz et al., 2001; Zhang et al., 2001). Each literal is tagged with its decision level, e.g., the literal corresponding to the \(n\)th decision and the literals derived by unit-propagation after it are labeled with \(n\); each non-decision literal \(l\) in \(\mu\) is also tagged by a link to the clause \(\psi_l\) causing its unit-propagation (called the antecedent clause of \(l\)). When a clause \(\psi\) is falsified by the current assignment — in which case we say that a conflict occurs and \(\psi\) is the conflicting clause — a conflict clause \(\psi'\) is computed from \(\psi\) s.t. \(\psi'\) contains only one literal \(l_u\) whose negation has been assigned at the last decision level. \(\psi'\) is computed starting from \(\psi' = \psi\) by iteratively resolving \(\psi'\) with the antecedent clause \(\psi_l\) of some literal \(l\) in \(\psi'\) (typically the last-assigned literal in \(\psi'\), see Zhang & Malik, 2002), until some stop criterion is met. E.g., with the 1st-UFP Scheme the last-assigned literal in \(\psi'\) is the one always picked, and the process stops as soon as \(\psi'\) contains only one literal \(l_u\) assigned at the last decision level; with the Decision Scheme, \(\psi'\) must contain only decision literals, including the last-assigned one. Consequently, Backtrack jumps up to the smallest decision level s.t. all but one literals in \(\psi'\) is falsified, and hence unit-propagates the remaining literal (i.e., the literal \(l_u\) above).

Intuitively, with backjumping DPLL goes back to the oldest decision where it would have acted differently if \(\psi'\) had been known, undoing all decisions and unit-propagations performed after then. Notice that, unlike classic chronological backtracking, backjumping allows for jumping up many decision levels, significantly pruning the search. Learning prevents DPLL from generating again in future the conflict set \(\eta \equiv \neg \psi'\): as soon as all but one literals in \(\eta\) are in \(\mu\), then the remaining literal is assigned to false by BCP on \(\psi'\). (To this extent, \(\psi'\) is also called blocking clause.) Consequently, learning allows for significantly pruning the search space.

One potential problem with learning is that it may require adding to \(\varphi\) up to an exponential number of clauses. If so, this could cause a blowup in space, and a drastic slowdown of BCP. However, it is not necessary to add permanently the conflict clause \(\psi'\) to \(\varphi\); rather, it is sufficient to keep it only as long as it is active, that is, as long as it is the antecedent clause of some literal in the current assignment. Since there may be at most \(|\text{Atoms}(\varphi)|\) active clauses, if clauses are discharged when they are no more active, then the blowup in the number of learned clauses is avoided. The only drawback of clause discharging is that the same conflict set can in principle be generated more than once. Nevertheless, this is guaranteed to preserve the correctness, completeness and termination of the algorithm (see, e.g., (Zhang et al., 2001; Nieuwenhuis, Oliveras, & Tinelli, 2006; Lahiri et al., 2006)).

In practice, typically CDCL SAT solvers discharge inactive clauses lazily: when an excessive number of learned clauses is revealed, then some of the inactive clauses are discharged according to some heuristic criterium (steps 7–8 in Figure 1 at p. 14).\(^7\) This typically guarantees a very good compromise between the benefits of learned clauses for pruning future branches and the overhead due to their maintenance.

\(^7\) Notice that different SAT solvers may implement different clause-discharging strategies, so that Steps 7–8 are not necessarily placed after BCP as in Figure 1 at p. 14.
2.2.3 CDCL SAT Solving Under Assumptions.

The schema in Figure 1 at p. 14 can be adapted to check also the satisfiability of a CNF propositional formula $\varphi$ under a set of assumptions $L \triangleq \{l_1, ..., l_k\}$. (From a purely-logical viewpoint, this corresponds to check the satisfiability of $\bigwedge_{l_i \in L} l_i \land \varphi$.) This works as follows: $l_1, ..., l_k$ are initially assigned to true, they are tagged as decision literals and added to $\mu$, then the decision level is reset to 0 and DPLL enters the external loop. If $\bigwedge_{l_i \in L} l_i \land \varphi$ is consistent, then DPLL returns sat; otherwise, DPLL eventually backtracks up to level 0 and then stops, returning conflict. Importantly, if AnalyzeConflict uses the Decision Scheme mentioned above, then the final conflict clause will be in the form $\bigvee_{l_j \in L'} \neg l_j$ s.t. $L'$ is the (possibly much smaller) subset of $L$ which actually caused the inconsistency revealed by the SAT solver (i.e., s.t. $\bigwedge_{l_j \in L'} l_j \land \varphi$ is inconsistent). In fact, at the very last branch, AnalyzeConflict will iteratively resolve the conflicting clause with the antecedent clauses of the unit-propagated literals until only decision literals are left: since this conflict has caused a backtrack up to level 0, these literals are necessarily all part of $L$.

This technique is very useful in some situations. First, sometimes one needs checking the satisfiability of a (possibly very big) formula $\varphi$ under many different sets of assumptions $L_1, ..., L_N$. If this is the case, instead of running DPLL on $\bigwedge_{l_i \in L} l_i \land \varphi$ for every $L_i$ —which means parsing the formulas and initializing DPLL from scratch each time— it is sufficient to parse $\varphi$ and initialize DPLL only once, and run the search under the different sets of assumptions $L_1, ..., L_N$. This is particularly important when parsing and initialization times are relevant w.r.t. solving times. In particular, if $\varphi$ is a Horn formula, solving $\varphi$ under assumptions requires only one run of BCP, whose computational cost depends linearly only on the clauses where the unit-propagated literals occur.

Second, this technique can be used in association with the use of selector variables: all the clauses $\psi_i$ of $\varphi$ can be substituted by the corresponding clauses $s_i \rightarrow \psi_i$, all $s_i$s being fresh propositional variables, which are initially assumed to be true (i.e., $L = \{s_i \mid \psi_i \in \varphi\}$). If $\varphi$ is unsatisfiable, then the final conflict clause will be of the form $\bigvee_{s_k \in L'} \neg s_k$, s.t. $\{\psi_k \mid s_k \in L'\}$ is the actual subset of clauses which caused the inconsistency of $\varphi$. This technique is used to compute the unsatisfiable cores of CNF propositional formulas in SAT (Lynce & Silva, 2004), and it has been extended to work with SMT (see Cimatti et al., 2011).

2.2.4 CDCL SAT Solving for SMT and All-SMT

Satisfiability Modulo Theories (SMT) is the problem of deciding the satisfiability of a (typically quantifier-free) first-order formula with respect to some decidable first-order theory. If we look at SMT from a SAT perspective, then the problem is equivalent to searching for a truth assignment $\mu$ satisfying a propositional formula $\varphi$ which is consistent w.r.t. some propositional theory $T$.\footnote{From a more standard SMT perspective, given an input quantifier-free first-order formula $\varphi_{fo}$ and some decidable first-order theory $T_{fo}$, the problem consists in searching for a set of literals $\mu_{fo}$ in $\varphi_{fo}$ whose conjunction tautologically entails $\varphi_{fo}$, and is consistent in $T_{fo}$. Thus, the propositional formula $\varphi$ is the Boolean abstraction of the input quantifier-free first-order formula $\varphi_{fo}$, in which each first-order atom is substituted by a fresh propositional atom, and the propositional theory $T$ is the Boolean abstraction of the set of clauses which can be built on atoms occurring in $\varphi_{fo}$ and are valid in the first-order theory.} To this extent, most SMT solvers are based on the lazy SMT paradigm, where a CDCL SAT solver is used to enumerate propositional truth assignments.
... if (status == sat) {
    \eta = T-solver(\mu);
    if (\eta == \emptyset) // \mu consistent w.r.t. T
        return sat;
    else { // \mu inconsistent w.r.t. T
        \psi = \neg \eta;
        \phi' = \text{AnalyzeConflict}(\phi, \mu, \psi);
        \phi = \phi \land \phi'; //\phi' learned temporarily
        \text{Backtrack}(\phi', \mu, \phi);
        if (\text{DecisionLevel}(\mu) == 0)
            return \text{unsat};
    }
}...

... if (status == sat) {
    \eta = T-solver(\mu);
    if (\eta != \emptyset) // \mu inconsistent w.r.t. T
        \psi = \neg \eta;
    else // \mu consistent w.r.t. T
        \psi' = \text{AnalyzeConflict}(\phi, \mu, \psi);
        \phi = \phi \land \psi'; //\psi' learned temporarily
        \text{Backtrack}(\psi', \mu, \phi);
        if (\text{DecisionLevel}(\mu) == 0)
            return;
}...

Figure 2: Left: SMT variant of Steps 10-11 in Figure 1 at p. 14. Right: All-SMT variant of Steps 10-11 in Figure 1 at p. 14. (The last five steps are identical to steps 14-18 in Figure 1 at p. 14.)

\(\mu_1, \mu_2, \ldots\) satisfying \(\phi\). Consider Figure 2 at p. 18 left. Every time a satisfying assignment \(\mu\) is generated, \(\mu\) is fed to a theory solver (\(T\)-solver) which checks its consistency w.r.t. \(T\); if \(\mu\) is consistent w.r.t. \(T\), then the whole procedure returns sat; if \(\mu\) is inconsistent w.r.t. \(T\), then the theory solver returns (the negation \(\eta\) of) one clause \(\psi\) valid in \(T\) (called theory lemma) which is violated by \(\mu\), which is then used by the CDCL SAT solver as a conflicting clause for triggering the backjumping and learning mechanism. The process is repeated until either a consistent assignment is found — so that \(\phi\) is consistent w.r.t. \(T\) — or no more truth assignment can be enumerated — so that \(\phi\) is inconsistent w.r.t. \(T\). Notice that the efficiency of the whole process strongly benefits from the fact that \(T\)-solver returns minimal \(T\)-inconsistent subsets \(\eta\).

This technique is further improved by means of early pruning and theory-propagation: the theory solver is invoked also on partial assignments \(\mu\) which do not satisfy \(\phi\) yet; if \(\mu\) is found \(T\)-inconsistent, then a theory lemma is produced and the SAT solver backtracks without further expanding \(\mu\); if instead \(\mu\) is found \(T\)-consistent, and if the theory solver is able to perform some deduction in the form \(T \land \mu' \models l\), s.t. \(\mu' \subseteq \mu\) and \(l\) is a literal representing a truth assignment to some unassigned atom in \(\phi\), then \(l\) is unit-propagated by the SAT solver and the theory lemma \(\mu \to l\) can be learned. We refer the reader to (Sebastiani, 2007; Barrett et al., 2009) for a survey on SMT and SMT-solving techniques.

An important extension of SMT is All-SMT (Lahiri et al., 2006), in which it is enumerated a complete set \(\{\mu_i\}\) of (possibly-partial) \(T\)-consistent truth assignment satisfying \(\phi\). This technique modifies the lazy SMT schema as follows (see Figure 2 at p. 18 right): whenever a (possibly-partial) propositional model \(\mu\) is generated and \(\mu\) is found consistent

\(T_{fo}\). Typically the theory solver reasons at first-order level, checking the consistency in \(T_{fo}\) of the corresponding set of first-order literals \(\mu_{fo}\).
in $T$ by $T$-solver, then $\psi \overset{\text{def}}{=} \neg \mu$ is treated as a conflicting clause, and AnalyzeConflict is applied, generating a conflict clause which is used to backjump, and then to continue the search. At the end of the search, the models $\mu_1, ..., \mu_n$ found [resp. the conflict sets $\eta_1, ..., \eta_k$ returned by $T$-solver] subsume all $T$-consistent [resp. $T$-inconsistent] total truth assignments propositionally satisfying $\varphi$.

As with the pure SAT case of Figure 1 at p. 14, it is not necessary to add permanently to $\varphi$ each conflict clause $\psi'$; rather, it is sufficient to keep it only as long as it is active, preserving the correctness, completeness and termination of the enumerating algorithm (see, e.g., (Zhang et al., 2001; Nieuwenhuis et al., 2006; Lahiri et al., 2006)), while avoiding the blowup in the number of learned clauses. As before, the only drawback of clause discharging is that the same model or conflict set can in principle be generated more than once, potentially worsening the global efficiency. As before, implementing the “lazy” clause-discharging approach of Steps 7-8 in Figure 1 at p. 14 represents a good compromise. According to the empirical evaluation of Lahiri et al. (2006), redundancies in enumeration appear to be rare and have low impact on performances.

3. Axiom Pinpointing via Horn SAT and Conflict Analysis

In this section we start presenting our novel contributions. In order not to break the flow of the discourse, the proofs of the new results have been moved to Appendix A at p. i.

3.1 Classification and Concept Subsumption via Horn SAT solving

We temporarily assume that $T$ is the result of a normalization process, as described in Section 2.1. (We will consider some issues related to the normalization at the end of Section 3.2.) We consider first the problem of concept subsumption. Then we can build a Horn propositional formula $\phi_T$ representing the classification of the input ontology $T$.

**Definition 3** ($\mathcal{EL}^+2\text{sat}(a_i)$, $\phi_T$). Let $T$ be an $\mathcal{EL}^+$ TBox in normal form and let $A$ be the classification of $T$. We introduce the set of propositional variables $\{p_{|X|} \mid X \in \text{NC}_T\}$, that we call concept variables, s.t. each concept variable $p_{|X|}$ is uniquely-associated to the respective concept $X$. For each assertion $a_i \in A$, we define the propositional encoding of $a_i$, written $\mathcal{EL}^+2\text{sat}(a_i)$, as follows:

\[
\begin{align*}
(p_{|C_1|} \land \ldots \land p_{|C_k|}) \rightarrow p_{|D|} & \quad \text{if } a_i \text{ is of type } (C_1 \sqcap \ldots \sqcap C_k) \subseteq D, k \geq 1; \\
\mathcal{EL}^+2\text{sat}(a_i) & \overset{\text{def}}{=} p_{|C|} \rightarrow p_{|\exists r.D|} \quad \text{if } a_i \text{ is of type } C \subseteq \exists r.D \\
p_{|\exists r.C|} \rightarrow p_{|D|} & \quad \text{if } a_i \text{ is of type } \exists r.C \subseteq D.
\end{align*}
\]

Then we define the following CNF Horn propositional formula:

\[
\phi_T \overset{\text{def}}{=} \bigwedge_{a_i \in A} \mathcal{EL}^+2\text{sat}(a_i).
\]

Notice that we do not encode trivial axioms of the form $C \subseteq C$ and $C \subseteq \top$ because they would generate valid clauses like $p_{|C|} \rightarrow p_{|C|}$ and $p_{|C|} \rightarrow \top$. 

\[9\] Here we assume that the output models/conflict sets are produced in output one-by-one and then possibly deleted, otherwise we need at least the possibly-exponential space to store all them.
Since the clauses (5)-(7) are definite Horn clauses, $\phi_T$ is a definite Horn formula. Thus, $\phi_T$ is satisfiable, and it is necessary to conjoin it with at least one positive and one negative literal in order to make it unsatisfiable.

**Theorem 1.** Given an $\mathcal{EL}^+$ TBox $T$ in normal form, for every pair of concept names $C, D$ in $\text{PC}_T$, $C \sqsubseteq_T D$ if and only if the Horn propositional formula $\phi_T \land p_C \land \lnot p_D$ is unsatisfiable.

In practice, in order to build $\phi_T$, we initially set it to an empty set of clauses; then we run the classification algorithm of Section 2.1: for every (non-trivial) original axiom of $T$ or every deduced assertion $a_i$ of the form (1)-(3) which is added to $A$, we add to $\phi_T$ the clause $\mathcal{EL}^+2sat(a_i)$.

**Remark 1.** Notice that, since the classification algorithm of Section 2.1 terminates after a polynomial number of rule applications and $|A| \leq |\text{PC}_T|^2 \cdot |\text{PR}_T|$ (see Section 2.1.3), $\phi_T$ is worst-case polynomial in size w.r.t. $|T|$ and can be generated in polynomial time. ◯

Once $\phi_T$ has been generated, in order to perform concept subsumption, we exploit the technique of CDCL SAT solving under assumptions described in Section 2.2: once $\phi_T$ is parsed and DPLL is initialized, each subsumption query $C_i \sqsubseteq_T D_i$ corresponds to solving $\phi_T$ under the assumption list $L_i \doteq \{p_{C_i}, \lnot p_{D_i}\}$. This corresponds to one single run of BCP, whose cost depends linearly only on the clauses where the unit-propagated literals occur. In practice, if $C_i \sqsubseteq_T D_i$ then $\phi_T$ contains the clause $p_{C_i} \rightarrow p_{D_i}$, so that BCP stops as soon as $p_{C_i}$ and $\lnot p_{D_i}$ are unit-propagated.

**Example 3.1.** Consider the classification $\mathcal{A}_{\text{Omed}}$ in Example 2.3 at p. 9.\(^{10}\) Then we have:

\[
\phi_{\text{Omed}} \doteq \begin{align*}
& P(\text{Appendix}) \rightarrow P(\text{BodyPart}) \quad a_1^i \quad \land \quad P(\text{Appendix}) \rightarrow P(\text{partOf} \text{ Intestine}) \quad a_1^v \\
& \land P(\text{Endocardium}) \rightarrow P(\text{Tissue}) \quad a_2^i \quad \land P(\text{Endocardium}) \rightarrow P(\text{partOf} \text{ HeartValve}) \quad a_2^v \\
& \land P(\text{Pericardium}) \rightarrow P(\text{Tissue}) \quad a_3^i \quad \land P(\text{Pericardium}) \rightarrow P(\text{containsIn} \text{ Heart}) \quad a_3^v \\
& \land P(\text{Appendicitis}) \rightarrow P(\text{Inflammation}) \quad a_4^i \quad \land P(\text{Appendicitis}) \rightarrow P(\text{hasLocation} \text{ Appendicitis}) \quad a_4^v \\
& \land P(\text{Pericarditis}) \rightarrow P(\text{Inflammation}) \quad a_5^i \quad \land P(\text{Pericarditis}) \rightarrow P(\text{hasLocation} \text{ Endocarditis}) \quad a_5^v \\
& \land P(\text{Inflammation}) \rightarrow P(\text{Disease}) \quad a_6^i \quad \land P(\text{Inflammation}) \rightarrow P(\text{actsOn} \text{ Tissue}) \quad a_6^v \\
& \land P(\text{Disease}) \lor P(\text{New}) \rightarrow P(\text{HeartDisease}) \quad a_7^i \quad \land P(\text{hasLocation} \text{ Heart}) \rightarrow P(\text{New}) \quad a_7^v \\
& \land P(\text{HeartDisease}) \rightarrow P(\text{hasState} \text{ NeedsTreatement}) \quad a_8^i \quad \land P(\text{hasState} \text{ NeedsTreatement}) \quad a_8^v \\
& \land P(\text{Appendicitis}) \rightarrow P(\text{actsOn} \text{ Tissue}) \quad b_1 \quad \land P(\text{actsOn} \text{ Tissue}) \quad b_1 \\
& \land P(\text{Pericarditis}) \rightarrow P(\text{actsOn} \text{ Tissue}) \quad b_2 \quad \land P(\text{actsOn} \text{ Tissue}) \quad b_2 \\
& \land P(\text{Pericarditis}) \rightarrow P(\text{actsOn} \text{ Tissue}) \quad b_3 \quad \land P(\text{actsOn} \text{ Tissue}) \quad b_3 \\
& \land P(\text{Pericarditis}) \rightarrow P(\text{actsOn} \text{ Tissue}) \quad b_4 \quad \land P(\text{actsOn} \text{ Tissue}) \quad b_4 \\
& \land P(\text{Pericarditis}) \rightarrow P(\text{actsOn} \text{ Tissue}) \quad b_5 \quad \land P(\text{actsOn} \text{ Tissue}) \quad b_5 \\
& \land P(\text{Pericarditis}) \rightarrow P(\text{actsOn} \text{ Tissue}) \quad b_6 \quad \land P(\text{actsOn} \text{ Tissue}) \quad b_6 \\
& \land P(\text{Pericarditis}) \rightarrow P(\text{actsOn} \text{ Tissue}) \quad b_7 \quad \land P(\text{actsOn} \text{ Tissue}) \quad b_7 \\
& \land P(\text{Pericarditis}) \rightarrow P(\text{actsOn} \text{ Tissue}) \quad b_8 \quad \land P(\text{actsOn} \text{ Tissue}) \quad b_8 \\
& \land P(\text{Pericarditis}) \rightarrow P(\text{actsOn} \text{ Tissue}) \quad b_9 \quad \land P(\text{actsOn} \text{ Tissue}) \quad b_9 \\
& \land P(\text{Pericarditis}) \rightarrow P(\text{actsOn} \text{ Tissue}) \quad b_{10} \quad \land P(\text{actsOn} \text{ Tissue}) \quad b_{10}
\end{align*}
\]

The clauses in the first nine rows represent the encoding of the (normalized) axioms of $\mathcal{O}_{\text{med}}$, while the clauses in the last five rows represent the encoding of the other subsumption

\[^{10}\text{In this section and in the following ones, with a small abuse of notation, we use the names $\mathcal{O}_{\text{med}}$ and $\mathcal{O}_{\text{milk}}$ to identify both the original ontology and its normalized version.}\]
relations deduced from the axioms of $\mathcal{O}_{\text{med}}$. We use the same labeling of Example 2.3 at p. 9.

For instance, performing concept subsumption on the query $\text{Pericarditis} \sqsubseteq \text{HeartDisease}$ corresponds to solve $\phi_{\mathcal{O}_{\text{med}}}$ under the assumption list $\{p[\text{Pericarditis}], \neg p[\text{HeartDisease}]\}$, which is clearly unsatisfiable because clause $b_0$ of $\phi_{\mathcal{O}_{\text{med}}}$ is falsified by the two assumed literals.

Instead, if the query is $\text{Appendicitis} \sqsubseteq \text{HeartDisease}$, this corresponds to solve $\phi_{\mathcal{O}_{\text{med}}}$ under the assumptions $\{p[\text{Appendicitis}], \neg p[\text{HeartDisease}]\}$. This leads to the unit-propagation in $\phi_{\mathcal{O}_{\text{med}}}$ of $p[\text{Inflammation}]$ and $p[\text{hasLocation}\_\text{Appendix}]$ from $a'_4$ and $a''_4$ respectively, then to the unit-propagation of $p[\text{Disease}]$ and $p[\text{actsOn}\_\text{Tissue}]$ from $a'_7$ and $a''_7$ (or equivalently from $b_1$ and $b_2$) respectively. After that, no more atom can be unit-propagated and no clause is falsified. Thus, since $\phi_{\mathcal{O}_{\text{med}}}$ is a Horn formula, we can conclude that it is satisfiable, and that $\text{Appendicitis} \sqsubseteq \text{HeartDisease}$ is not a subsumption relation deducible from $\mathcal{O}_{\text{med}}$. \\$

3.2 Computing single and all MinAs via Conflict Analysis

We consider the general problem of generating all MinAs. We build a more-general Horn propositional formula $\phi^\text{all}_T$ representing the complete classification DAG of the input normalized ontology $\mathcal{T}$.\textsuperscript{11} The size of $\phi^\text{all}_T$ is polynomial w.r.t. that of $\mathcal{T}$.

3.2.1 Building the formula $\phi^\text{all}_T$

In order to make the explanation simpler, we assume wlog. that in all the axioms of $\mathcal{T}$ all $\sqcap$'s and $\circ$'s are binary, i.e., that $1 \leq k \leq 2$ in (2) and $1 \leq n \leq 2$ in (4). This is not restrictive, since, e.g., each GCI axiom of the form $C_1 \sqcap \ldots \sqcap C_k \sqsubseteq D$ in $\mathcal{T}$ can be rewritten into the set $\{C_1 \sqcap C_2 \sqsubseteq C_{1,2}, C_{1,2} \sqcap C_3 \sqsubseteq C_{1,3}, \ldots, C_{1,k-1} \sqcap C_k \sqsubseteq D\}$, and each RI axiom of the form $r_1 \circ \ldots \circ r_n \sqsubseteq s$ can be rewritten into the set $\{r_1 \circ r_2 \sqsubseteq r_{1,2}, r_{1,2} \circ r_3 \sqsubseteq r_{1,3}, \ldots, r_{1,n-1} \circ r_n \sqsubseteq s\}$, each $C_{i} \sqcap$ and $r_{1,j}$ being a fresh concept name and a fresh role name respectively.\textsuperscript{12}

**Definition 4 ($\phi^\text{all}_T, \phi_{T(\text{so})}, \phi_{T(\text{po})}$).** Let $\mathcal{T}$ be an $\mathcal{EL}^+$ TBox in normal form and let $\mathcal{A}$ be the classification of $\mathcal{T}$. We consider the concept variables $\{p[X] \mid X \in \text{NC}_\mathcal{T}\}$. We introduce the set of propositional variables $\mathcal{P}_\mathcal{A} \overset{\text{def}}{=} \{s_{[a]} \mid a \in \mathcal{A}\}$, that we call assertion [resp. axiom] selector variables. Then we define $\phi^\text{all}_T \overset{\text{def}}{=} \phi_{T(\text{so})} \wedge \phi_{T(\text{po})}$, where

- $\phi_{T(\text{so})}$ is the conjunction of all the clauses:
  \[
  \{s_{[a]} \rightarrow \mathcal{EL}^+ \text{2sat}(a_i) \mid a_i \in \mathcal{A}\}
  \]
  that we call assertion clauses, and

- $\phi_{T(\text{po})}$ is the conjunction of all the clauses:
  \[
  \{(s_{[a]} \land s_{[a'] \land s_{[a']}}) \rightarrow s_{[a_k]} \mid a_i, a_j, a_k \in \mathcal{A}, a_j \in \mathcal{T} \text{ and } \text{r}(a_i, a_j, a_j, a_k)\}
  \]
  that we call rule clauses. With $\text{r}(a_i, a_j, a_j, a_k)$ we mean that $\text{r}$ is one of the completion rules of the classification algorithm of Section 2.1 and $a_i, a_j, a_j, a_k$ are valid instances

\textsuperscript{11} Here “complete” means “including also the rule applications generating already-generated assertions”.

\textsuperscript{12} Notice, however, that this is just a simplifying assumption to make our explanation easier: in practice our encodings and algorithms deal with n-ary operators.
of (respectively) the preconditions (left and central columns of Table 2 at p. 9) and of the conclusion (right column of Table 2 at p. 9) of \( r \). (Some require only one assertion \( a_i \) and one axiom \( a_j \) as premises of the rule \( r \); in these cases let \( s_{[a_i]} \) be \( T \).)

Notice that (9) and (10) are definite Horn clauses and, hence, \( \phi^{all} \) is a definite Horn formula.

**Proposition 2.** The size of the formula \( \phi^{all}_T \) defined in Definition 4 at p. 21 is worst-case polynomial in the size of the TBox \( T \).

The result in Theorem 1 at p. 20 extends straightforwardly to \( \phi^{all}_T \), as described in the following.

**Theorem 3.** Given an \( \mathcal{EL}^+ \) TBox \( T \) in normal form, for every \( S \subseteq T \) and for every pair of concept names \( C, D \) in \( \mathcal{PC}_T \), \( C \sqsubseteq_S D \) if and only if the Horn propositional formula

\[
\phi^{all}_{T(\text{po})} \land \bigwedge_{ax_i \in S} s_{[ax_i]} \land \neg s_{[C \sqsubseteq D]} \text{ is unsatisfiable.}
\]

**Theorem 4.** Given an \( \mathcal{EL}^+ \) TBox \( T \) in normal form, for every \( S \subseteq T \) and for every pair of concept names \( C, D \) in \( \mathcal{PC}_T \), \( C \sqsubseteq_S D \) if and only if the Horn propositional formula

\[
\phi^{all}_T \land \bigwedge_{ax_i \in S} s_{[ax_i]} \land p_C \land \neg p_D \text{ is unsatisfiable.}
\]

**Corollary 5.** Given an \( \mathcal{EL}^+ \) TBox \( T \) in normal form, for every pair of concept names \( C, D \) in \( \mathcal{PC}_T \), \( C \subseteq_T D \) if and only if the Horn propositional formula \( \phi^{all}_T \land \bigwedge_{ax_i \in T} s_{[ax_i]} \land \neg s_{[C \sqsubseteq D]} \) [resp. \( \phi^{all}_{T(\text{po})} \land \bigwedge_{ax_i \in T} s_{[ax_i]} \land \neg s_{[C \sqsubseteq D]} \)] is unsatisfiable.

Intuitively, \( \phi^{all}_{T(\text{po})} \) mimics the whole classification process, each rule clause representing one rule application. Thus, if a SAT solver is fed the formula \( \phi^{all}_{T(\text{po})} \land \bigwedge_{ax_i \in S} s_{[ax_i]} \land \neg s_{[C \sqsubseteq D]} \) of Theorem 3 at p. 22 (or \( \phi^{all}_{T(\text{po})} \) under the assumption list \( \{\neg s_{[C \sqsubseteq D]}\} \cup \{s_{[ax_i]} \mid ax_i \in S\} \)), then all the variables \( s_{[ax_i]} \) s.t. \( a_j \) can be deduced from \( S \) are instantly unit-propagated. If (and only if) \( C \subseteq_S D \), then also \( s_{[C \sqsubseteq D]} \) is unit-propagated, causing a conflict. Similarly, if the formula \( \phi^{all}_T \land \bigwedge_{ax_i \in S} s_{[ax_i]} \land p_C \land \neg p_D \) of Theorem 4 at p. 22 is fed to the SAT solver, then if (and only if) \( C \subseteq_S D \), then \( s_{[C \sqsubseteq D]} \) is unit-propagated, which causes a conflict against the assertion clause \( s_{[C \sqsubseteq D]} \rightarrow (p_C \rightarrow p_D) \) in \( \phi^{all}_T \) and the unit clauses \( p_C \land \neg p_D \).

Notice that, in general, there may be more than one way of deducing \( C \subseteq D \) from \( S \). This corresponds to the fact that there may be more than one unit-propagation sequence leading to the propagation of \( s_{[C \sqsubseteq D]} \). (We will investigate this issue in Section 3.2.3.)

**Remark 2.** Theorem 4 at p. 22 suggest that, once the formula \( \phi^{all}_T \) is generated, it is possible to reason in terms of every subset \( S \) of \( T \) by “selecting” all and only the axioms we are interested in. This requires no new formula generation or computation on \( S \) or \( T \). Rather, it is sufficient to restrict the list of the assumptions for each query on \( \phi^{all}_T \) to the set of the selector variables of the axioms of \( S \) and to the selector variable of the query.
Example 3.2. Consider the ontology $\mathcal{O}_{med}$ in Examples 2.1 at p. 5 and 2.3 at p. 9. Then

$$\phi_{\mathcal{O}_{med}}^\text{all} \overset{\text{def}}{=} \phi_{\mathcal{O}_{med}(so)} \land \phi_{\mathcal{O}_{med}(po)}.$$ 

Notice that $s_{[a10]}$ and $s_{[a11]}$ refer to the RI axioms $a_{10}$ and $a_{11}$ in Example 2.1 at p. 5, so that no corresponding assertion rule occurs in $\phi_{\mathcal{O}_{med}(so)}$.

Consider the formula $\phi_{\mathcal{O}_{med}}^\text{all} \land \bigwedge_{i=0,\ldots,11} s_{[a_i]} \land \lnot P[\text{Pericarditis}] \land \lnot P[\text{HeartDisease}]$. The propagation of $P[\text{Pericarditis}]$ and $\lnot P[\text{HeartDisease}]$ causes the propagation of $\lnot s_{[b_0]}$ from the last but one clause of $\phi_{\mathcal{O}_{med}(so)}$. The propagation of $s_{[a_9]}$, $s_{[a_3]}$, $s_{[a_6]}$, $s_{[a_7]}$ and $s_{[a_11]}$ causes that of $s_{[b_3]}$, $s_{[b_7]}$ and hence of $s_{[b_9]}$ (from the fifth, seventh and eighth clauses of $\phi_{\mathcal{O}_{med}(po)}$). Thus, since also $s_{[a_8]}$ is propagated, the nine-th clause in $\phi_{\mathcal{O}_{med}(po)}$ is falsified. Thus, we conclude that $\text{Pericarditis} \subseteq \mathcal{O}_{med} \text{ HeartDisease}$.

It is easy to see, instead, that the formula $\phi_{\mathcal{O}_{med}}^\text{all} \land \bigwedge_{\forall i \in \mathcal{O}_{med}} s_{[a_i]} \land P[\text{Appendicitis}] \land \lnot P[\text{HeartDisease}]$ is satisfiable, from which we conclude that $\text{Appendicitis} \subseteq \mathcal{O}_{med} \text{ HeartDisease}$.

Example 3.3. We report some sample clauses from the formula $\phi_{\mathcal{O}_{med}(po)}^\text{all}$ for the ontology $\mathcal{O}_{milk}$ of Example 2.2 at p. 8. (On the right side we show the mapping between the axiom/assertion selector variables included in the sample clauses and the concept inclusions

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{example-clauses.png}
\caption{Example clauses for $\mathcal{O}_{milk}$.
\textbf{Example 3.3.} We report some sample clauses from the formula $\phi_{\mathcal{O}_{med}(po)}^\text{all}$ for the ontology $\mathcal{O}_{milk}$ of Example 2.2 at p. 8. (On the right side we show the mapping between the axiom/assertion selector variables included in the sample clauses and the concept inclusions

\end{figure}
they represent.)

\[
\phi^{all}_{\text{GCI}}(po) \overset{\text{def}}{=} \ldots \land \ldots \land \text{m}_0 = \exists \text{hasPhysState}.\text{liquidState} \subseteq \text{N} \\
\text{s}_{[\text{m}_5]} \land \text{s}_{[\text{m}_4]} \rightarrow \text{s}_{[\text{n}_1]} \land \ldots \land \text{m}_1 = \text{BodyFluid} \subseteq \text{Fluid} \quad \text{n}_1 = \text{Milk} \subseteq \text{Substance} \\
\text{s}_{[\text{m}_6]} \land \text{s}_{[\text{m}_3]} \rightarrow \text{s}_{[\text{n}_2]} \land \ldots \land \text{m}_2 = \text{Liquid} \subseteq \text{Fluid} \quad \text{n}_2 = \text{Milk} \subseteq \text{N} \\
\text{s}_{[\text{m}_5]} \land \text{s}_{[\text{n}_2]} \land \text{s}_{[\text{m}_3]} \rightarrow \text{s}_{[\text{n}_3]} \land \ldots \land \text{m}_3 = \text{BodySubstance} \land \text{N} \subseteq \text{BodyFluid} \quad \text{n}_3 = \text{Milk} \subseteq \text{BodyFluid} \\
\text{s}_{[\text{m}_1]} \land \text{s}_{[\text{n}_2]} \land \text{s}_{[\text{m}_4]} \rightarrow \text{s}_{[\text{n}_4]} \land \ldots \land \text{m}_4 = \text{BodySubstance} \subseteq \text{N} \quad \text{n}_4 = \text{Milk} \subseteq \text{Liquid} \\
\text{s}_{[\text{m}_3]} \land \text{s}_{[\text{m}_4]} \rightarrow \text{s}_{[\text{n}_5]} \land \ldots \land \text{m}_5 = \text{Milk} \subseteq \text{BodySubstance} \quad \text{n}_5 = \text{Milk} \subseteq \text{Fluid} \\
\text{s}_{[\text{n}_4]} \land \text{s}_{[\text{m}_4]} \rightarrow \text{s}_{[\text{n}_6]} \land \ldots \land \text{m}_6 = \text{Milk} \subseteq \exists \text{hasPhysState}.\text{liquidState} \\
\text{m}_9 = \text{Substance} \land \text{N} \subseteq \text{Liquid} \\
\ldots
\]

We notice that, assuming all the \text{s}_{[\text{m}_i]}'s, there are two distinct chains of unit-propagations leading to propagate \text{s}_{[\text{n}_3]}: one from \{\text{s}_{[\text{m}_0]}, \text{s}_{[\text{m}_1]}, \text{s}_{[\text{m}_3]}, \text{s}_{[\text{m}_5]}, \text{s}_{[\text{m}_6]}\}, propagating \text{s}_{[\text{n}_2]}, \text{s}_{[\text{n}_3]} and \text{s}_{[\text{n}_5]}; and another from \{\text{s}_{[\text{m}_0]}, \text{s}_{[\text{m}_2]}, \text{s}_{[\text{m}_4]}, \text{s}_{[\text{m}_5]}, \text{s}_{[\text{m}_6]}, \text{s}_{[\text{m}_9]}\}, propagating \text{s}_{[\text{n}_1]}, \text{s}_{[\text{n}_2]}, \text{s}_{[\text{n}_4]} and \text{s}_{[\text{n}_5]}; corresponding respectively to the deduction of \text{n}_2, \text{n}_3 and \text{n}_5 from the axioms \{\text{m}_0, \text{m}_1, \text{m}_3, \text{m}_5, \text{m}_6\} and to that of \text{n}_1, \text{n}_2, \text{n}_4 and \text{n}_5 from \{\text{m}_0, \text{m}_2, \text{m}_4, \text{m}_5, \text{m}_6, \text{m}_9\}. Thus we can conclude that \{\text{m}_0, \text{m}_1, \text{m}_3, \text{m}_5, \text{m}_6\} and \{\text{m}_0, \text{m}_2, \text{m}_4, \text{m}_5, \text{m}_6, \text{m}_9\} are two \text{nMinAs for} \text{n}_5 \overset{\text{def}}{=} \text{Milk} \subseteq \text{O}_\text{GCI} \text{Fluid}. Since they are also minimal, they are also \text{MinAs for} \text{n}_5. 

\diamond

In practice, in order to build the formula \phi^{all}_{\text{T}}(po), we run an extended version of the classification algorithm of Section 2.1, whose pseudo-code representation is presented in Figure 3 at p. 25, and which is based on the following main steps:

1. initially set \phi_{\text{T}(so)} and \phi^{all}_{\text{T}(po)} to the empty set of clauses. Then for every non-trivial GCI axiom \text{a}_i \in \text{T}, add to \phi_{\text{T}(so)} the corresponding assertion clause of type (9);

2. for every newly-deduced assertion \text{a}_i \in \text{A}, add to \phi_{\text{T}(so)} the corresponding assertion clause of type (9);

3. for every possible rule instantiation \text{r}(\text{a}_1, \text{a}_1', \text{a}_j, \text{a}_k) of a completion rule \text{r} of Table 2 at p. 9 (either extending \text{A} or not), add to \phi^{all}_{\text{T}(po)} the corresponding rule clause of type (10).

(Notice that step 3. is novel w.r.t. the classification algorithm of Section 2.1 when applied to already-generated assertions in \text{A}.) We perform step 2. and 3. in a queue-based manner, which ensures that every possible distinct (i.e. with different antecedents) rule application is applied only once. This is achieved with the following strategy: initially all GCI axioms are added to a queue \text{Q} and all axioms are included in \text{A}. At each iteration one assertion \text{a}_h is dequeued, and steps 2. and/or 3. are applied to all and only the rule applications whose antecedents are exactly \text{a}_h and one or two of the previously-dequeued axioms/assumptions \text{a}_1, ..., \text{a}_{h-1}. The novel assertions \text{a}_k which are deduced by the rule applications in step 2 are enqueued into \text{Q} and added to \text{A}. This process ends when \text{Q} is empty.

The algorithm exposed above requires a polynomial amount of steps w.r.t. the size of \text{T}. In fact, it avoids to repeat the same rule application (i.e. with exactly the same antecedents and consequence) more than once, and each rule application leads to the introduction of one or two clauses. Therefore the algorithm requires linear time w.r.t. the size of \phi^{all}_{\text{T}} that, in Proposition 2 at p. 22, has been proved to be at most polynomial in the size of \text{T}.
ClauseSet build-ϕ'all (NormalizedOntology T)
// Initialization of Q, A, ϕT(so), ϕ'all
1. Q = ∅; A = ∅; ϕT(so) = ∅; ϕ'all = ∅;
2. for each primitive concept C in T {
   3. add C ⊆ C and C ⊆ T to A; introduce s[C⊆C] = s[C⊆T] = T;
   4. enqueue {C ⊆ C, C ⊆ T} into Q;
3. for each GCI or RI axiom ax in T {
   4. add ax to A; introduce s[ax];
   5. if (ax is a non-trivial GCI axiom) then {
      6. add the clause (s[ax] → EL+2sat(ax)) to ϕT(so);
      7. enqueue ax into Q; }
4. // Updating A, B and Q (B is the set of all handled assertions)
5. B = ∅;
6. while (Q is not empty) {
      7. dequeue ak from Q;
      8. for each rule instance r(ai, a'i, aj, ak) such that {ai, a'i, aj} \ B = {ak} {
           9. if ak ∉ A then {
               10. add ak to A; introduce s[ak];
               11. add the clause (s[ak] → EL+2sat(ak)) to ϕT(so);
               12. enqueue ak into Q;
            }
           13. add the clause ((s[ai] ∧ s[a'i] ∧ s[aj]) → s[ak]) to ϕ'all;
           14. B = B ∪ {ak};
       }
    15. return ϕ'all = ϕT(so) ∨ ϕ'all;
}

Figure 3: Polynomial-time algorithm building the formula ϕ'all. Q is a queue of assertions, A and B are sets of assertions.

3.2.2 Computing single MinAs.

We consider some sub-ontology S ⊆ T. Once ϕ'all is generated, in order to compute single MinAs, we can exploit the technique of CDCL SAT solving under assumptions adopting the Decision Scheme, as described in Section 2.2. Theorem 4 at p. 22 guarantees that, given the set of the axiom selector variables PS = {s[aj] | axj ∈ S}, after ϕ'all is parsed and DPLL is initialized, deciding if C1 ⊆ S D1 holds is equivalent to solving ϕ'all under the assumption

\[ L_i = P_S \cup \{ p_{c1}, \neg p_{d1} \} \]

which corresponds to a single run of BCP and possibly one run of AnalyzeConflict and Backtrack, whose cost depends linearly only on the clauses where the unit-propagated literals occur. If BCP does not return conflict, then sat is returned without even performing conflict analysis. If BCP returns conflict, as explained in Section 2.2, then AnalyzeConflict produces a conflict clause \[ \psi_{S;D} = \{ p_{C1}, \neg p_{D1} \} \]

which is a conflict clause for C1 ⊆ S D1. In fact, since ϕ'all is a definite Horn formula, the presence of both \[ p_{C1} \] and \[ \neg p_{D1} \] in \[ L_i \] is necessary for causing the conflict, so that, due to the Decision Scheme, the conflict set necessarily contains both of them. (Intuitively, AnalyzeConflict implicitly spans upward the classification sub-DAG rooted in C1 ⊆ D1 and having S* as
variable-list ComputeOneMinA(formula $\phi^{all}_{T(po)}$, variable-list $P$, literal $\neg s[C_i \subseteq D_i]$)

1. if (DpllUnderAssumptions($\phi^{all}_{T(po)}$, $P \cup \{-s[C_i \subseteq D_i]\}$) == unsat) \{ // if $C_i \not\subseteq S D_i$
  2. \[ P_{S^*} = \{s_{[a_i]} \mid a_i \in S^*\}; \]
  // s.t. $\psi^{C_i,D_i} \triangleq s[C_i,D_i] \vee \wedge_{a_i \in S^*} \neg s_{[a_i]}$ is the conflict clause returned
  3. for each axiom selector variable $s_{[ax_j]}$ in $P_{S^*}$ \{ 
    4. \[ L = P_{S^*} \setminus \{s_{[ax_j]}\}; \]
    5. if (DpllUnderAssumptions($\phi^{all}_{T(po)}$, $L$) == unsat) // if $C_i \not\subseteq S^* \setminus \{ax_j\} D_i$
    6. \[ P_{S^*} = L; \}
    7. return $P_{S^*} \setminus \{-s[C_i \subseteq D_i]\}; \}
  8. else
  9. return $\emptyset$;

Figure 4: SAT-based variant of the MinA-extracting algorithm of Baader et al. (2007).

leaf nodes, which contains all and only the nodes of the assertions which have been used to generate $C_i \subseteq D_i$.)

Analogously, by Theorem 3 at p. 22, deciding if $C_i \subseteq S D_i$ holds corresponds also to solving $\phi^{all}_{T(po)}$ under the assumption list $L_i \triangleq P \cup \{-s[C_i \subseteq D_i]\}$, so that the algorithm for pinpointing is changed only in the fact that $\phi^{all}_{T(po)}$ and $\{-s[C_i \subseteq D_i]\}$ are used instead of $\phi^{all}_T$ and $\{s[C_i], \neg p[D_i]\}$ respectively, returning the clause $\psi^{C_i,D_i} \triangleq s[C_i,D_i] \vee \wedge_{a_i \in S^*} \neg s_{[a_i]}$ s.t. $S^*$ is an nMinA for $C_i \subseteq S D_i$. Thus, wlog. in the remaining part of this section we will reason using $\phi^{all}_{T(po)}$ and $\{-s[C_i \subseteq D_i]\}$. (The same results, however, can be obtained using $\phi^{all}_T$ and $\{s[C_i], \neg p[D_i]\}$ instead.)

In order to produce a minimal set of axioms, we can apply the SAT-based variant of the linear minimization algorithm of Baader et al. (2007) in Figure 4 at p. 26. Given $\phi^{all}_{T(po)}$, the set of the axiom variables $P$ and the query selector variable $s[C_i \subseteq D_i]$, ComputeOneMinA computes the set of the axiom selector variables representing one MinA $S^*$ if $C_i \subseteq S D_i$, or returns the empty set otherwise. (As before, we assume that $\phi^{all}_{T(po)}$ has been parsed and DPLL has been initialized.) First, as described above, $\phi^{all}_{T(po)}$ is solved under the assumption list $L_i \triangleq P \cup \{-s[C_i \subseteq D_i]\}$. If satisfiable, the whole procedure returns the empty set. Otherwise, DpllUnderAssumptions return unsat, producing the conflict clause $\psi^{S^*}_{S^*} \triangleq s[S^*]$ and hence the set $P_{S^*} = \{s_{[a_i]} \mid a_i \in S^*\}$ s.t. $S^*$ is an nMinA. Then the algorithm tries to remove one-by-one (the selection variables of) the axioms $ax_j$s in $S^*$, each time checking whether the reduced set of axioms $S^* \setminus \{ax_j\}$ is still such that $C_i \subseteq S^* \setminus \{ax_j\} D_i$. To this extent, the minimization algorithm in rows 3–6 is identical to that in (Baader et al., 2007), except that the check $C_i \subseteq S^* \setminus \{ax_j\} D_i$ is performed by invoking DpllUnderAssumptions over $\phi^{all}_{T(po)}$ and $P_{S^*} \setminus \{s_{[ax_j]}\}$, as justified by Theorem 3 at p. 22.

This minimization schema can be further improved as follows. If DpllUnderAssumptions in row 5 performs also conflict analysis and returns (the conflict clause corresponding to) an nMinA $S'$ s.t. $S' \subseteq S^* \setminus \{ax_j\}$, then in row 6 $P_{S^*}$ is assigned to $P_{S'}$ rather than to $L$, so that all axioms in $(S^* \setminus \{ax_j\}) \setminus S'$ will not be selected in next loops. As an alternative
choice, one can implement instead a SAT-based version of the binary-search variant of the minimization algorithm presented in (Baader & Suntisrivaraporn, 2008).

3.2.3 Computing all MinAs.

We consider some sub-ontology $S \subseteq T$. We describe a way of generating all MinAs for $C_i \sqsubseteq S$ $D_i$ from $\phi_{T(p)}^{all}$ and $\{-s[C_i[D_i]]\}$. (As before, the same results can be obtained if $\phi_p^{all}$ and $\{p[C_i], \neg p[D_i]\}$ are used instead.) The idea is to enumerate all minimal truth assignments on the axiom selector variables in $P$ and $C$.

The novel procedure, which is presented in Figure 5 at p. 28, is a straightforward variant of the All-SMT technique of Lahiri et al. (2006) of Figures 1 at p. 14 and 2 at p. 18 described in Section 2.2.4. (We assume that AnalyzeConflict adopts the Decision Scheme, so that all conflict clauses are built on decision literals only.)

The procedure works as follows. We consider a propositional CNF formula $\varphi$ on the set of axiom selector variables $P_S$. $\varphi$ is initially set to $T$ (i.e., the empty set of clauses).

One top-level instance of DPLL (namely “DPLL1”) is used to enumerate a complete set of truth assignments $\{\mu_1, \ldots, \mu_k\}$ on the axiom selector variables in $P_S$ which satisfy $\varphi$. (To this extent, the external loop, the call to BCP, and the handling of the “unknown” and “conflict” cases in Figure 5 at p. 28 are identical to those in Figure 1 at p. 14, and will not be described again here; thus we focus on the “sat” case of Steps 9-23.)

Every time that a novel assignment $\mu$ is generated, we extract the subset $\mu^+$ of the positive literals in $\mu$, and we pass $\mu^+$ and $\{-s[C_i[D_i]]\}$ to an ad-hoc “theory solver” checking if $\mu^+$ falsifies the formula $\neg s[C_i[D_i]] \land \phi_{T(p)}^{all}$. As theory solver we use the concatenation of the functions $\text{PositiveLiteralsOf}$, which selects the positive literals in $\mu$, and $\text{ComputeOneMinA}$ of Figure 4 at p. 26. (To this extent, here $\neg s[C_i[D_i]] \land \phi_{T(p)}^{all}$ plays the role of the propositional theory $T$.) Notice that we consider only the positive literals in $\mu$ because, by construction, all atoms in $P_S$ occur only negatively in $\phi_{T(p)}^{all}$, so that the negative literals cannot have any role in falsifying $\neg s[C_i[D_i]] \land \phi_{T(p)}^{all}$.

If $\mu^+ \land \neg s[C_i[D_i]] \land \phi_{T(p)}^{all} \models \bot$, then $\mu^+$ represents an nMinA for $C_i \sqsubseteq S$ $D_i$, and $\text{ComputeOneMinA}$ returns a minimal subset $\eta$ of $\mu^+$ which caused such inconsistency. By construction, $S^* \equiv \{ax_j \in S \mid s[ax_j] \in \eta\}$ is a MinA. The clause $\psi \equiv \neg \eta = \bigvee_{ax_j \in S^*} \neg s[ax_j]$ is then permanently added to $\varphi$ as a blocking clause and it is used as a conflicting clause for driving next backjumping and learning steps. Otherwise, $\text{ComputeOneMinA}$ returns $\psi \equiv \neg \mu^+$ is used as a “fake” conflicting clause for driving next backjumping and learning steps. As usual, the whole process terminates when Backtrack back-jumps to blevel zero. The set of all MinAs $\{\eta_1, \ldots, \eta_k\}$ computed in this way is returned as output.

Comparing Steps 9-23 of Figure 5 at p. 28 with Figure 2 at p. 18 right, it is straightforward to notice that $\text{ComputeAllMinAs}$ is a slight variant of the All-SMT procedure of Section 2.2.4, $\neg s[C_i[D_i]] \land \phi_{T(p)}^{all}$ being the propositional theory $T$ and the concatenation $\text{PositiveLiteralsOf}/\text{ComputeOneMinA}$ being the theory solver, with only one slight difference: $\neg \eta$ is also learned permanently; although this step is not strictly necessary (see Figure 5).

Notice that DPLL does not require that all Boolean variables in $\text{vars}$ actually occur in $\varphi$. If the idea of enumerating truth assignments for the $T$ formula puzzles the reader, however, he/she may pretend $\varphi$ is initially a valid propositional formula on all the atoms in $P_S$, e.g., $\varphi \equiv \bigwedge_{A_i \in P_S} (A_i \lor \neg A_i)$. 27

---

13. Notice that DPLL does not require that all Boolean variables in $\text{vars}$ actually occur in $\varphi$. If the idea of enumerating truth assignments for the $T$ formula puzzles the reader, however, he/she may pretend $\varphi$ is initially a valid propositional formula on all the atoms in $P_S$, e.g., $\varphi \equiv \bigwedge_{A_i \in P_S} (A_i \lor \neg A_i)$. 27
MinA-Set ComputeAllMinAs (Variable-list \( \mathcal{P}_S \), literal \( \{ \neg s_{[C \sqsubseteq D_i]} \} \))

// the formula \( \phi_{T(p_0)}^{\text{all}} \) is global, and it is already initialized
1. \( \varphi = T; \mu = \emptyset; \text{vars} = \mathcal{P}_S; \text{MinAs} = \emptyset; \)
2. while (1) {
   3. while (1) {
      4. status = BCP(\( \varphi \), \( \mu \));
      5. if (status == unknown) {
         6. if (TooManyClauses(\( \varphi \))) {
            7. DeleteSomeInactiveClauses(\( \varphi \));
         }
         8. break;
      }
      9. if (status == sat) {
         10. \( \mu^+ = \text{PositiveLiteralsOf}(\mu) \);
         11. \( \eta = \text{ComputeOneMinA}(\phi_{T(p_0)}^{\text{all}}, \mu^+, \neg s_{[C \sqsubseteq D_i]} \);\)
         12. if (\( \eta \neq \emptyset \)) { // \( \mu^+ \) contains a MinA
            13. MinAs = MinAs \( \cup \eta \);
            14. \( \psi = \neg \eta \);
            15. \( \varphi = \varphi \land \psi \); } // \( \psi \) learned permanently
         16. else { // \( \mu^+ \) contains no MinA
            17. \( \psi = \neg \mu \); }
         18. \( \psi' = \text{AnalyzeConflict}(\varphi, \mu, \psi) \);
         19. \( \varphi = \varphi \land \psi' \); // \( \psi' \) learned temporarily
         20. Backtrack(\( \psi', \mu, \varphi \));
         21. if (DecisionLevel(\( \mu \)) == 0) {
            22. return MinAs;
         }
      }
      23. }
   24. else { // status == conflict
      25. \( \psi = \text{FalsifiedClause}(\varphi, \mu) \);
      26. \( \psi' = \text{AnalyzeConflict}(\varphi, \mu, \psi) \);
      27. \( \varphi = \varphi \land \psi' \); \( \psi' \) learned temporarily
      28. Backtrack(\( \psi', \mu, \varphi \));
      29. if (DecisionLevel(\( \mu \)) == 0) {
         30. return MinAs;
      }
   }
   32. }
   33. DecideNextBranch(\( \varphi \), \( \mu \));
   34. }

Figure 5: All-SMT-based algorithm generating “all MinAs” w.r.t. the given query \( C_i \sqsubseteq_{T D_i} \).

Figure 2 at p. 18 right), it further guarantees that we do not generate redundant MinAs, and that no redundant calls to the possibly-expensive process of minimization is performed.
Therefore, the correctness, completeness and termination of this algorithm comes directly from that of the All-SMT (see Nieuwenhuis et al., 2006; Lahiri et al., 2006). As discussed in Section 2.2.4, the only drawback of clause discharging is that the same conflict clause $\psi'$—but not the MinA $\neg \eta$, which is learned permanently—can in principle be generated more than once, potentially worsening the global efficiency. As before, implementing the “lazy” clause-discharging approach represents a good compromise.

An important improvement is based on the following consideration. If an assignment $\mu \defeq \mu_1^+ \cup \mu_1^-$ does not cause the generation of a MinA, then obviously no assignment in the form $\mu' \defeq (\mu_1^+ \setminus \{s_{i_1}, ..., s_{i_k}\}) \cup (\mu^- \cup \{\neg s_{i_1}, ..., \neg s_{i_k}\})$ s.t. $\{s_{i_1}, ..., s_{i_k}\} \subseteq \mu_1^+$ can generate a MinA as well. Thus, since all the enumerated assignments $\mu$ are total, no assignment which is a superset of $\mu^- \defeq \mu_1 \setminus \mu_1^+$ has any chance of generating a MinA. Hence, we can safely improve the algorithm of Figure 5 by substituting instructions 16-17 with the following:

16. \( \text{else} \{ // \mu^+ \text{ contains no MinA} \)
17. \( \psi = -(\mu \setminus \mu^+); \} \)

which prevents the top-level Boolean search from enumerating any of the $2^{[\mu^+]_1}$ assignments extending $\mu^-$. This greatly helps pruning the top-level Boolean search space.

A few more facts are important for the overall efficiency. First, \texttt{DecideNextBranch} always assigns the new assigned variables to true. This guarantees that the first-generated assignments contain a larger number of enabled selector variables, so that it is more likely that they include new nMinAs. To this extent, notice that the first assignment is $\mu_1 \defeq \{s_{[ax_1]} \mid ax_1 \in \mathcal{P}_S\}$, and it always leads to the discovery of the first MinA. Second, the expensive steps of parsing $\phi^{\text{all}}_{\mathcal{T}(po)}$ and initializing the two DPLL instances inside \texttt{ComputeAllMinAs} and \texttt{ComputeOneMinA} (namely “DPLL2”) respectively are performed only once, before the whole process starts, and both \texttt{ComputeAllMinAs} and \texttt{ComputeOneMinA} are always invoked incrementally. Thus it is possible to perform different calls to \texttt{ComputeAllMinAs} with different axiom-variable lists and queries without reinitializing the procedure from scratch.

\textbf{Example 3.4.} We want to find all the MinAs for the subsumption \textit{Milk} $\sqsubseteq_{\mathcal{O}_{\text{milk}}} \text{Fluid}$ in the ontology $\mathcal{O}_{\text{milk}}$ (Examples 2.2 at p. 8, 3.3 at p. 23). We run the procedure in Figure 5 at p. 28 on the formula $\phi^{\text{all}}_{\mathcal{O}_{\text{milk}}(po)}$ and w.r.t. the query $\mathcal{S}_{\text{Fluid}}$. The top-level DPLL1 enumerates all the truth assignments on the variables $\mathcal{P}_{\mathcal{O}_{\text{milk}}} \defeq \{s_{[m_i]} \mid m_i \in \mathcal{O}_{\text{milk}}\}$, satisfying the formula $\varphi$ initially set to $\top$. The first truth assignment produced is $\mu_1^+ \defeq \mu_1 = \mathcal{P}_{\mathcal{O}_{\text{milk}}} = \{s_{[m_1]} \mid m_1 \in \mathcal{O}_{\text{milk}}\}$: running \texttt{ComputeOneMinA} on $\phi^{\text{all}}_{\mathcal{O}_{\text{milk}}(po)}$ and the input assumptions $\mu_1$ and $\neg \mathcal{S}_{\text{Fluid}}$ leads to the identification of the first MinA, e.g., $\mathcal{O}_{\text{milk}} \defeq \{m_0, m_2, m_4, m_5, m_6, m_9\}$. Thus the blocking clause $\psi_1 \defeq (\neg s_{[m_0]} \lor \neg s_{[m_2]} \lor \neg s_{[m_3]} \lor \neg s_{[m_5]} \lor \neg s_{[m_6]} \lor \neg s_{[m_9]})$ is added to $\varphi$, which becomes $\varphi = \psi_1$. Since $\psi_1$ contains (the negation of) no unit-propagated literal, the procedure backjumps on $\psi_1' = \psi_1$, generating another truth assignment $\mu_2$, which we suppose wlog. to be $\mathcal{P}_{\mathcal{O}_{\text{milk}}} \cup \{\neg s_{[m_5]}\} \setminus \{s_{[m_3]}\}$, so that

---

14. Here we only simulate the execution of the procedure without showing the encoding $\phi^{\text{all}}_{\mathcal{O}_{\text{milk}}(po)}$, whose significant clauses can be found in Example 3.3 at p. 23. Our purpose, in fact, is only to show the steps of the procedure in a concrete case.

15. With $m_0$ we represent all the new definition axioms introduced during the normalization of the ontology, which label complex sub-concepts with fresh concept names (see Section 2.1.2 and Section 3.2.4).
\( \mu_2 \equiv \mathcal{P}_{\text{milk}} \setminus \{ s_{[m_3]} \} \). Since \( \phi_{\text{milk}(po)}^{\text{all}} \wedge \mu_2 \wedge \{ \neg s_{\text{Milk} \subseteq \text{Fluid}} \} \) is satisfiable, ComputeOneMinA returns the empty set, and \( \psi'_2 = \psi_2 = -\mu_2 \setminus \mu_2^2 = s_{[m_3]} \) is used by DPLL1 as a fake conflicting clause, driving the backjumping.\(^{16}\) Then DPLL1 generates a third assignment \( \mu_3 \), which we suppose \( \mu_3 = \mathcal{P}_{\text{milk}} \setminus \{ s_{[m_4]} \} \cup \{ \neg s_{[m_4]} \} \) so that \( \mu_3^2 = \mathcal{P}_{\text{milk}} \setminus \{ s_{[m_4]} \} \). In this case ComputeOneMinA identifies the new MinA \( \mathcal{O}_{\text{milk}_2} = \{ m_0, m_1, m_3, m_5, m_6 \} \), producing \( \psi_3 = (\neg s_{[m_0]} \lor \neg s_{[m_1]} \lor \neg s_{[m_3]} \lor \neg s_{[m_5]} \lor \neg s_{[m_6]} \) \), which is permanently added to \( \varphi \), and \( \psi'_3 = \psi_3 \) drives the next backjumping step. Since now \( \varphi = \psi_1 \wedge \psi_3 \) and no other MinA for \( \text{Milk} \subseteq \text{Fluid} \) other than \( \mathcal{O}_{\text{milk}_1} \) and \( \mathcal{O}_{\text{milk}_2} \) exists, the procedure will proceed until termination enumerating only truth assignments which do not falsify \( \neg s_{\text{Milk} \subseteq \text{Fluid}} \wedge \phi_{\text{milk}(po)}^{\text{all}} \).

One further remark is in order. The reason why we use two nested instances of DPLL is that we must distinguish unit-propagations of negated axiom selector variables \( \neg s_{[ax]} \) on learned clauses from those performed on the clauses in \( \phi_{\mathcal{T}(po)}^{\text{all}} \): on the one hand, we want to allow the former ones because they prevent exploring the same assignments more than once; on the other hand, we want to avoid the latter ones (or to perform them in a controlled way, as explained in Section 4.2 for the theory propagation variant) because they may prevent generating some counter-model of interest.

3.2.4 Handling normalization.

The normalized TBox \( \mathcal{T} \equiv \{ ax_1, ..., ax_N \} \) can result from normalizing the non-normal TBox \( \tilde{\mathcal{T}} \equiv \{ \tilde{ax}_1, ..., \tilde{ax}_N \} \) by means of the process hinted in Section 2.1; \( |\mathcal{T}| \) is \( \mathcal{O}(|\tilde{\mathcal{T}}|) \). Each original axiom \( ax_i \) is converted into a set of normalized axioms \( \{ ax_{i1}, ..., ax_{im_i} \} \), and each normalized axiom \( ax_j \) can be reused (in the case of a definition axiom) in the conversion of several original axioms \( \tilde{ax}_j, ..., \tilde{ax}_j \). In order to handle non-normal TBoxes \( \tilde{\mathcal{T}} \), one variant of the technique of Baader et al. (2007) can be adopted: for every \( \tilde{ax}_i \), we add to \( \phi_{\mathcal{T}(po)}^{\text{all}} \) [resp. \( \phi_{\mathcal{T}}^{\text{all}} \)] the set of clauses \( \{ s_{[\tilde{ax}_i]} \rightarrow s_{[ax_{i1}]}, ..., s_{[\tilde{ax}_i]} \rightarrow s_{[ax_{im_i}]} \} \), and then we use \( \mathcal{P}_{\tilde{\mathcal{T}}} \equiv \{ s_{[\tilde{ax}_1]}, ..., s_{[\tilde{ax}_N]} \} \) as the novel set of axiom selector variables for the one-MinA and all-MinAs algorithms described above. Thus AnalyzeConflict finds conflict clauses in terms of variables in \( \mathcal{P}_{\tilde{\mathcal{T}}} \) rather than in \( \mathcal{P}_{\mathcal{T}} \). (In practice, we treat normalization as the application of a novel kind of completion rules.) Since \( \mathcal{P}_{\mathcal{T}} \) is typically smaller than \( \mathcal{P}_{\tilde{\mathcal{T}}} \), this may cause a significant reduction in the search space that the DPLL1 must explore during the all-MinAs procedure of Figure 5 at p. 28. (Notice that when one \( ax_j \) is shared by \( \tilde{ax}_{j1}, ..., \tilde{ax}_{jm_j} \), the clause set \( \{ s_{[\tilde{ax}_{j1}]} \rightarrow s_{[ax]} , ..., s_{[\tilde{ax}_{jm_j}]} \rightarrow s_{[ax]} \} \) is equivalent to \( \{ s_{[\tilde{ax}_{j1}]} \lor ... \lor s_{[\tilde{ax}_{jm_j}]} \} \rightarrow s_{[ax]} \).)

Alternatively, a more compact solution we adopted allows for using directly and only the selector variables referring to the original axioms \( \tilde{\mathcal{T}} = \{ ax_1, ..., ax_N \} \). In such a way no extra clause is added to the encoding and a smaller number of selector variables is used. In fact, every non-normal axiom of \( \mathcal{T} \) is normalized into two sets of normal axioms: (i) a set of top-level axioms in which complex concept descriptions are substituted by newly introduced

\(^{16}\) In fact \( m_3 \equiv \text{Milk} \sqsubseteq \text{BodySubstance} \) is a key axiom included in any MinA for \( \text{Milk} \sqsubseteq \text{Fluid} \). To be a BodySubstance is necessary both in order to be a BodyFluid \( (m_3) \) and in order to be \( (\text{a Substance} \text{ first and then}) \) a Liquid \( (m_4 \text{ and } m_6) \).
Axiom Pinpointing in large $\mathcal{EL}^+$ ontologies via SAT and SMT techniques

concept names, and which keep representing the original concept inclusions, (ii) and a set of definition(s) (axioms) which represent the relations between the fresh concept names and the corresponding complex concept descriptions. For example the concept inclusion $\exists r.A \sqsubseteq \exists s.B \sqsubseteq C \cap D$ is normalized into the set of top-level normal axioms: $\{ X \subseteq C, X \subseteq D \}$ and the set of definition axioms: $\{ \exists r.A \sqsubseteq Y, \exists s.B \sqsubseteq Z, Y \cap Z \subseteq X \}$, which define Y as $\exists r.A$, Z as $\exists s.B$ and X as $Y \cap Z$.

The idea, which is inspired to the work of Plaisted and Greenbaum (1986), is to:

(i) use the same original axiom selector variable $s_{[ax]}$ for all the top-level normal axioms coming out from the normalization of $\alpha x_i$; 

(ii) associate the same unique selector variable $s_{[a0]}$ to all the description axioms introduced.

An informal explanation of this latest choice is that definition axioms play the role of labeling for complex concepts, so they take part in the deduction of a queried subsumption only in consequence of top-level axioms. Further, queries are always expressed in terms of original concept names, so we are ensured that the top-level selector variables of the original axioms are always (and firstly) involved in the search. The single selector variable $s_{[a0]}$, instead, is used to represent and enable all the axioms defining the new concept names coming out from the normalization. Thus, the presence of $s_{[a0]}$ in a MinA is not of interest, it only indicates that at-least one of the axiom included in the MinA has been normalized. Finally, notice that some definitions can be (partially) shared among many different original axioms, but the above-exposed solution is transparent w.r.t. these situations. This schema for handling normalization has been already used in Examples 3.2 at p. 22 and 3.4 at p. 29.

(Hereafter we will call $T$ the input TBox, assuming that it is in normal form, no matter if it is resulting from a normalization process or not and if we use the selector variables referring to the original axioms or to the normalized ones.)

3.3 Discussion

We first compare our all-MinAs technique for $\mathcal{EL}^+$ of Section 3.2 with that presented by Baader et al. (2007). By comparing the pinpointing formula $\Phi^{C_1 \sqsubseteq T, D_1}$ of Baader et al. (2007) (see also Section 2.1) with $\phi^{all}_{T(po)}$, and by analyzing the way they are built and used, we highlight the following differences:

(i) $\Phi^{C_1 \sqsubseteq T, D_1}$ is built only on axiom selector variables in $P_T \overset{\text{def}}{=} \{ s_{[ax]} \mid ax_j \in T \}$, whilst $\phi^{all}_{T(po)}$ is build on all selector variables in $A \overset{\text{def}}{=} \{ s_{[a]} \mid a_j \in A \}$ (i.e., of both axioms and inferred assertions);

(ii) the size of $\Phi^{C_1 \sqsubseteq T, D_1}$ and the time to compute it are worst-case exponential in $|T|$ (Baader et al., 2007), whilst the size of $\phi^{all}_{T(po)}$ and the time to compute it are worst-case polynomial in $|T|$;

(iii) the algorithm for generating $\Phi^{C_1 \sqsubseteq T, D_1}$ of (Baader et al., 2007) requires intermediate logical checks, whilst the algorithm for building $\phi^{all}_{T(po)}$ does not;

17. Notice that more than one top-level axiom can result from the normalization of an equivalence relation or from the normalization of a right-side conjunction.
(iv) each MinA is a model of $\Phi_{C_i \subseteq \mathcal{T}D_i}$, whilst it is (the projection to $\mathcal{P}_T$ of) a counter-model of $\phi_{T\{\text{po}\}}^{\text{all}}$.

Moreover, our process can reason directly in terms of (the selector variables of) the input axioms, no matter whether normal or not.

In accordance with Baader et al. (2007) and Peñaloza and Sertkaya (2010b), also our approach is not output-polynomial, because in our proposed all-MinAs procedure even the enumeration of a polynomial amount of MinAs may require exploring an exponential amount of models. In our proposed approach, however, the potential exponentiality is completely relegated to the final step of our approach, i.e. to our variant of the all-SMT search, since the construction of the SAT formula is polynomial. Thus we can build $\phi_{T\{\text{po}\}}^{\text{all}}$ once and then, for each sub-ontology $\mathcal{S} \subseteq \mathcal{T}$ of interest and for each $C_i \subseteq \mathcal{S} D_i$ of interest, run the all-SMT procedure until either it terminates or a given timeout is reached: in the latter case, we can collect the MinAs generated so far. (Notice that the fact that DPLL1 selects positive axiom selector variables first tends to anticipate the enumeration of over-constrained assignments w.r.t. to that of under-constrained ones, so that it is more likely that counter-models, and thus MinAs, are enumerated during the first part of the search. In particular it is assured that it finds one MinA in polynomial time.) With the all-MinAs algorithm of Baader et al. (2007), instead, it may take an exponential amount of time to build the pinpointing formula $\Phi_{C_i \subseteq \mathcal{T}D_i}$ before starting the enumeration of the MinAs.

We stress the fact that, once $\phi_{T\{\text{po}\}}^{\text{all}}$ is generated, in order to evaluate different sub-ontologies $\mathcal{S}$, it suffices to assume different axiom selector variables, without modifying the formula. Similarly, if we want to compute one or all MinAs for different deduced assertions, e.g. $C_1 \subseteq \mathcal{S} D_1, \ldots, C_i \subseteq \mathcal{S} D_i, \ldots$, we do not need recomputing $\phi_{T\{\text{po}\}}^{\text{all}}$ each time, we just need assuming (i.e. querying) each time a different axiom selector variable, e.g. respectively: $\neg s[C_1 \subseteq \mathcal{S} D_1], \ldots, \neg s[C_i \subseteq \mathcal{S} D_i], \ldots$. Same discourse holds for $\phi_{T}^{\text{all}}$ and $\{p[C_i], \neg p[D_i]\}$. Notice that this fact allows for a more fine-grained debugging of ontologies. In particular, it allows for testing the interactions among only some selected parts of an ontology, or for working in terms of refutable and irrefutable axioms (see, e.g., Baader et al., 2007). In fact, in many applications it is necessary to partition an ontology into trusted (i.e., irrefutable) axioms, whose correctness is established, and untrusted (i.e., refutable) ones, whose correctness is still uncertain to the designer (or to the user) of the ontology. For example, if an already well-established ontology is extended, one might view the new axioms as refutable, and trust the previously existing part of the ontology.

4. Improving the All-MinA Enumeration

As far as full All-MinA enumeration is concerned, the techniques described in Section 3.2 are still naive to some extents, and their efficiency is not yet satisfactory. (See the empirical results in the short version of this paper (Sebastiani & Vescovi, 2009b).) In particular, most real-world problems are too large to be entirely enumerated by our approach, so we must try to reduce the search space as much as possible. In this section we describe some important novel enhancements which greatly improve the performances of our technique. For the sake of clarity, in the following we use the full ontology $\mathcal{T}$, and hence the full list of
axiom selection variables $\mathcal{P}_T$, although the same techniques can be always applied to $\phi_{T^{(po)}}^\text{all}$ (or $\phi_T^\text{all}$) considering some sub-ontology $\mathcal{S}/\mathcal{P}_S$ instead of $\mathcal{T}/\mathcal{P}_T$.

4.1 Cone-of-influence Modularization

The most important improvement we have introduced is a SAT-based form of modularization, which we call Cone-of-influence Modularization for its analogy with the cone-of-influence reduction used in model checking (see, e.g., Clarke et al., 1999). This technique is similar in aim—but different in content and results—to that proposed by Baader and Suntisrivaraporn (2008), Suntisrivaraporn (2009) which we have described in Section 2.1.5: the idea is to isolate the set $\mathcal{P}_T^{C_i\subseteq D_i}$ of the selector variables in $\phi_{T^{(po)}}^\text{all}$ labeling all the axioms in $\mathcal{T}$ which might have a role in the inference of a given query $C_i \sqsubseteq \mathcal{T} D_i$, so that we can restrict our All-SMT process to these selector variables only.

**Definition 5.** Let $\mathcal{T}$ be an $\mathcal{EL}^+$ TBox in normal form, $C_i \sqsubseteq \mathcal{T} D_i$ be an existing subsumption relation in $\mathcal{T}$, $\phi_{T^{(po)}}^\text{all}$ be the encoding of $\mathcal{T}$ in Definition 4 and $\mathcal{P}_T \overset{\text{def}}{=} \{s_{[ax]} \mid ax_i \in \mathcal{T}\}$ the set of the axiom selector variables for $\mathcal{T}$. Then the set of selector variables $\mathcal{P}_A^{C_i\subseteq D_i}$ and the set of rule clauses $\phi_T^{C_i\subseteq D_i}$ are defined inductively as follows:

**base:** $s_{[C_i \subseteq D_i]} \in \mathcal{P}_A^{C_i\subseteq D_i}$

**induction:**

- if $s_{[a]} \in \mathcal{P}_A^{C_i\subseteq D_i}$, then every clause $(\bigwedge_j s_{[a_j]} \rightarrow s_{[a]}$ in $\phi_{T^{(po)}}^\text{all}$ is also in $\phi_T^{C_i\subseteq D_i}$;
- if $(\bigwedge_{j=1}^n s_{[a_j]} \rightarrow s_{[a]}$ is in $\phi_T^{C_i\subseteq D_i}$, then $s_{[a]} \in \mathcal{P}_A^{C_i\subseteq D_i}$ for every $j \in \mathcal{1}, ..., \mathcal{n}$. 

$\mathcal{P}_A^{C_i\subseteq D_i}$ and $\phi_T^{C_i\subseteq D_i}$ are called cone-of-influence (COI) and COI formula respectively. Furthermore, we define COI Module’s Assumptions for $C_i \sqsubseteq \mathcal{T} D_i$ w.r.t. $\phi_{T^{(po)}}^\text{all}$, namely $\mathcal{P}_T^{C_i\subseteq D_i}$, the set of axiom selector variables $\mathcal{P}_A^{C_i\subseteq D_i} \overset{\text{def}}{=} \mathcal{P}_A^{C_i\subseteq D_i} \cap \mathcal{P}_T$. Finally, the COI Module for $C_i \sqsubseteq \mathcal{T} D_i$ w.r.t. $\phi_{T^{(po)}}^\text{all}$, namely $\mathcal{M}_{C_i\subseteq D_i}^{\text{coi}}$, is the set $\mathcal{M}_{C_i\subseteq D_i}^{\text{coi}} \subseteq \mathcal{T}$ of axioms such that $\mathcal{M}_{C_i\subseteq D_i}^{\text{coi}} \overset{\text{def}}{=} \{ax_i \mid s_{[ax]} \in \mathcal{P}_T^{C_i\subseteq D_i}\}$.

**Example 4.1.** We compute the COI module $\mathcal{M}_{\text{Pericarditis} \sqsubseteq \text{HeartDisease}}^{\text{coi}}$ w.r.t. the encoding $\phi_{\text{med}(po)}^\text{all}$ from Example 3.2 of the sample ontology $\mathcal{O}_{\text{med}}$ of Examples 2.1 and 2.3.

$$\phi_{\text{med}(po)}^\text{all} \overset{\text{def}}{=} s_{[a_{[7]}]} \land s_{[a_{[7]}]} \rightarrow s_{[b_{[7]}]} \land s_{[a_{[7]}]} \land s_{[a_{[7]}]} \rightarrow s_{[b_{[7]}]} \land s_{[b_{[7]}]} \land s_{[a_{[7]}]} \rightarrow s_{[b_{[7]}]} \land s_{[b_{[7]}]} \land s_{[a_{[7]}]} \rightarrow s_{[b_{[7]}]}$$

The query Pericarditis $\sqsubseteq \mathcal{O}_{\text{med}}$ HeartDisease corresponds to the assumption $b_9$. We underline the selector variables involved by the COI for $b_9$: (i) positive literals are wavy
underlined, (ii) assertion selector variables are singly underlined and (iii) axiom selector variables are doubly underlined. The set of all the axiom/assumption selector variables marked (ii) and (iii) represent the COI $\mathcal{P}^{\text{Pericarditis}}_{\text{HeartDisease}}$, the set of clauses containing underlined variables represent the COI formula $\phi^{\text{Pericarditis}}_{\text{HeartDisease}}$, and the set of the axioms represented by the selector variables marked (iii) represent the module

$$\mathcal{M}^{\text{COI}}_{\text{Pericarditis} \sqsubseteq \text{HeartDisease}} = \{a_0, a_3, a_6, a_7, a_8, a_{11}\}.$$  

The module is inductively defined backward from the positive literal $s_{[a_9]}$. Due to the first clause in row (e) also the literals $s_{[b_6]}$, $s_{[b_5]}$ are involved in the cone of influence and $a_8$ is added to the initially-empty module; due to $s_{[b_6]}$ (second clause of row (d)), also $s_{[b_7]}$ is part of the cone and $a_0$ of the module while, similarly, the other axioms $a_6$ and $a_7$ are part of the module consequently to the clauses implying $s_{[b_6]}$ (first clause of row (c)). Finally, also axioms $a_3$ and $a_{11}$ are incorporated due to the clause involving $s_{[b_7]}$ (first clause of row (d)).

Notice that, $a_9$ is not included in the COI module for $\text{Pericarditis} \sqsubseteq \text{HeartDisease}$, unlike in the reachability-based module for $\text{Pericarditis}$ (see Example 2.5 for the computation of $\mathcal{M}^{\text{reach \ Pericarditis}}_{\text{Pericarditis}}$).

The following result is a straightforward consequence of Definition 5 and Theorem 3, and it is formally proved in Appendix A.

**Theorem 6.** Given an $\mathcal{EL}^+$ TBox $T$ in normal form and the formula $\phi^{\text{all}}_{T,(po)}$ as defined in Definition 4, for every pair of concept names $C, D$ in $\mathcal{PC}_T$, the following facts hold:

(i) $C \sqsubseteq T D$ if and only if $C \sqsubseteq \mathcal{M}^{\text{COI}}_{C \sqsubseteq D}$;

(ii) if $S$ is a MinA for $C \sqsubseteq T D$, then $S \subseteq \mathcal{M}^{\text{COI}}_{C \sqsubseteq D}$,

where $\mathcal{M}^{\text{COI}}_{C \sqsubseteq D}$ is the COI module for $C \sqsubseteq D$ w.r.t. $\phi^{\text{all}}_{T,(po)}$, as defined in Definition 5.

Notice that point (i) of Theorem 6 is a direct consequence of point (ii), but we distinguished them because (i) states that the COI module preserves the subsumption relation for which the module is computed, whilst (ii) states that the COI module contains all the possible MinAs responsible of such a subsumption relation.

The computation of the COI Module can be performed straightforwardly through the simple queue-based algorithm of Figure 6. ComputeCOIModule traverses breadth-first the COI $\mathcal{P}^{\text{A}}_{A} \sqsubseteq D_i$ and the COI formula $\phi^{\text{COI}}_{T \sqsubseteq D_i}$ starting from the positive selector variable $s_{[C \sqsubseteq D_i]}$, mirroring their inductive definition in Definition 5.

Importantly, Step 5 of the algorithm in Figure 6 can be performed efficiently if we exploit the two-watched-literals technique (Moskewicz et al., 2001) (see Section 2.2) implemented in all the modern state-of-the-art SAT solvers. In fact, since all the clauses in $\phi^{\text{all}}_{T,(po)}$ are definite Horn clauses (see Section 3.2) they are all implications having exactly one positive literal (that is $s_{[a_i]}$ in our exposition). Therefore, at loading/parsing time, we can force the only positive literal of each clause to be one of its two watched literals. This ensures that at Step 5, through the two-watched-literal scheme, we can obtain the set $\phi_{a_i}$ of all the clauses in which the literal $s_{[a_i]}$ appears positively in linear time w.r.t. the cardinality of $\phi_{a_i}$ itself.

**Proposition 7.** Given the Horn propositional formula $\phi^{\text{all}}_{T,(po)}$, the set of assumptions $\mathcal{P}_T$ and the query $s_{[C \sqsubseteq D_i]}$, the algorithm of Figure 6 executes in linear time w.r.t. the number of clauses of the COI subformula $\phi^{\text{COI}}_{T \sqsubseteq D_i}$. 

R. Sebastiani & M. Vescovi
Axiom Pinpointing in large $\mathcal{EL}^+$ ontologies via SAT and SMT techniques

variable-list ComputeCOIModule (formula $\phi_{T(\text{po})}^{\text{all}}$), variable-list $P_T$, literal $s_{[C_i \sqsubseteq D_i]}$

1. $Q = \emptyset$; $P_T^{C_i \sqsubseteq D_i} = \emptyset$;
2. enqueue $s_{[C_i \sqsubseteq D_i]}$ in $Q$; mark $s_{[C_i \sqsubseteq D_i]}$ as reached;
3. while $Q$ is not empty {
    4. dequeue $s_{[a_i]}$ from $Q$;
    5. let $\phi_{a_i}$ be the set of all the clauses of the form $s_{[a_i]} \lor \bigvee_j \neg s_{[a_j]}$;
    6. for each clause $c \in \phi_{a_i}$ {
        7. for each $\neg s_{[a_j]}$ occurring in $c$ {
            8. if ($s_{[a_j]}$ is not reached)
                9. enqueue $s_{[a_j]}$ in $Q$; mark $s_{[a_j]}$ as reached;
            10. if ($s_{[a_j]} \in P_T$)
                11. $P_T^{C_i \sqsubseteq D_i} = P_T^{C_i \sqsubseteq D_i} \cup \{s_{[a_j]}\}$;
                12. 
                13. 
                14. }
            15. }
4. return $P_T^{C_i \sqsubseteq D_i}$

Figure 6: Schema of the SAT-based COI Modularization algorithm.

Since the COI Modularization algorithm is computationally cheap and, most important, the modules computed are typically orders of magnitude smaller than $T$, this technique improves dramatically the performance of our approach (see Section 5).

Remark 3. Importantly, this technique works directly on the propositional input formula $\phi_{T(\text{po})}^{\text{all}}$, with no need for re-computing anything from $T$, or for performing any other form of $\mathcal{EL}^+$ reasoning. From the perspective of our approach, this is a point in favor of our COI modularization since we can solve every query working directly on the SAT formula regardless the original ontology. Furthermore, with exactly the same procedure we can obtain the COI module’s assertions $P_S^{C_i \sqsubseteq D_i}$ for any desired sub-ontology $S$ of $T$: once $P_A^{C_i \sqsubseteq D_i}$ has been computed from $\phi_{T(\text{po})}^{\text{all}}$, we have simply $P_S^{C_i \sqsubseteq D_i} \equiv P_A^{C_i \sqsubseteq D_i} \cap P_S$. ☐

Although similar in aim, COI Modularization and Reachability-based Modularization (see Section 2.1.5) present substantial differences.

First, differently from Reachability-based Modularization, in which modules are extracted from the original axioms of $T$ proceeding forward on the basis of syntactic interactions between the signatures of the axioms (starting from the given signature), COI Modularization extracts modules in a SAT-based manner proceeding backward according to the propositional interactions between the clauses of the encoding $\phi_{T(\text{po})}^{\text{all}}$ (starting from the query selector variable). Therefore, even if COI Modularization relies directly on the propositional encoding instead of reasoning in terms of description logic, it is a semantic modularization technique. In fact, the SAT formula $\phi_{T(\text{po})}^{\text{all}}$ includes (and hides) the semantics of $T$ handled during the construction of the formula.

Second, Reachability-based Modularization does not depend on the specific set of completion rules adopted, whilst the COI technique does. In fact, reachability-based modules
are computed before the classification independently from the completion rules, while COI modules are computed after the classification — and thus exploiting the information which is produced by the classification — on the basis of the encoding $\phi_{T,(po)}^{all}$, whose definition depends on the specific set of completion rules adopted. A different COI modules would be produced if a different set of completion rules was used.

Third, we state that our approach is more precise than the reachability-based one, i.e., that it extracts smaller or equal modules. We have the following fact.

**Proposition 8.** Given an $\mathcal{EL}^+ TBox T$ in normal form and the formula $\phi_{T,(po)}^{all}$ as defined in Definition 4, for every pair of concept names $C, D$ in $PC_T$ we have that $M^{reach}_{C \subseteq D} \subseteq M^{c,o,i}_{C \subseteq D}$, where $M^{reach}_{C \subseteq D}$ and $M^{c,o,i}_{C \subseteq D}$ are, respectively, the reachability-based module (Definition 2) and COI module (Definition 5) for $C \subseteq D$.

In fact, whilst Reachability-based Modularization computes a module “dependent” only from $C$, COI Modularization computes a module “dependent” from both $C$ and $D$ (i.e. dependent from the subsumption relation $C \sqsubseteq D$). For example, the reachability-based module computed for the query $C \sqsubseteq D$ is the same computed for the query $C \sqsubseteq E$, because in both cases it includes everything is syntactically reachable from $C$: more precisely, it also includes in the module everything else is syntactically reachable either from $D$ or from $E$. (This fact makes us conjecture that some kind of combination of a forward reachability from $C$ and a backward one from $D$ might be used to refine the reachability-based technique.)

**Example 4.2.** The following example shows in the ontology not-Galen the differences between the reachability-based module $M^{reach}_{\text{Lithium}}$ and the COI module $M^{c,o,i}_{\text{Lithium} \subseteq \text{ChemicalSubstance}}$ for the existing subsumption relation $\text{Lithium} \sqsubseteq \text{not-Galen ChemicalSubstance}$.

<table>
<thead>
<tr>
<th>Lithium $\sqsubseteq$ ElementalChemical</th>
<th>hasFeature $\sqsubseteq$ FeatureStateAttribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lithium $\sqsubseteq$ Metal</td>
<td>$\exists r$. FeatureStateAttribute</td>
</tr>
<tr>
<td>ElementalChemical $\sqsubseteq$</td>
<td>$\exists r$. ModifierAttribute</td>
</tr>
<tr>
<td>Metal $\sqsubseteq$ ComplexChemicals</td>
<td>$\exists r$. DomainAttribute</td>
</tr>
<tr>
<td>ChemicalSubstance $\sqsubseteq$</td>
<td>$\exists r$. Attribute</td>
</tr>
<tr>
<td>Substance $\sqsubseteq$ GeneralisedSubstance</td>
<td>$\exists r$. infinitelyDivisible mass</td>
</tr>
<tr>
<td>GeneralisedSubstance $\sqsubseteq$</td>
<td>$\exists r$. mass</td>
</tr>
<tr>
<td>DomainCategory $\sqsubseteq$ TopCategory</td>
<td>$\exists r$. CountabilityStatus</td>
</tr>
<tr>
<td>Substance $\sqsubseteq$ $\exists r$. hasCountability.infinatelyDivisible</td>
<td>$\exists r$. AbstractStatus</td>
</tr>
<tr>
<td>StructuralAppearanceModifier $\sqsubseteq$</td>
<td>$\exists r$. Status</td>
</tr>
<tr>
<td>StructuralModifierAttribute $\sqsubseteq$</td>
<td>$\exists r$. Aspect</td>
</tr>
</tbody>
</table>

While $M^{reach}_{\text{Lithium}}$ consists of all the 24 listed axioms of not-Galen, $M^{c,o,i}_{\text{Lithium} \subseteq \text{ChemicalSubstance}}$ consists of only the first 5 axioms (listed above the separation line).

### 4.2 Theory Propagation

Another important improvement to the $\text{ComputeAllMinAs}$ procedure of Figure 5 we have introduced in the new version of $\mathcal{EL}^+ SAT$ is that of exploiting early pruning and theory propagation, two well-known techniques from SMT which we briefly described in Section 2.2.4. (In (Sebastiani & Vescovi, 2009b) we already suggested the potential applicability of these techniques).
techniques, but they were not implemented in the first version of \(\mathcal{EL}^+\text{SAT}\)). The idea is that the “theory solver” ComputeOneMinA can be invoked also when status = unknown, that is, when \(\mu\) is a partial truth assignments on \(\mathcal{P}_S\). If \(\mu^+\) is enough to falsify the “theory” \(\neg s_{[c1 \sqsubseteq D_1]} \land \phi^{all}_{T(p_o)}\), then ComputeOneMinA computes and returns one MinA, without exploring all possible extensions of \(\mu\). If \(\mu^+\) does not falsify \(\neg s_{[c1 \sqsubseteq D_1]} \land \phi^{all}_{T(p_o)}\), then it may be the case that inside ComputeOneMinA \(\mu^+\) causes the unit-propagation on \(\phi^{all}_{T(p_o)}\) of one (or more) literal \(\neg s_{[axj]}\) s.t. \(s_{[axj]} \in \mathcal{P}_S\) and \(s_{[axj]}\) is unassigned. If this is the case, then \(\mu^+ \land s_{[axj]}\) represents a non-minimal truth assignment falsifying \(\neg s_{[c1 \sqsubseteq D_1]} \land \phi^{all}_{T(p_o)}\). Consequently, (i) the main procedure can unit-propagate \(\neg s_{[axj]}\), and (ii) \(\mu^+ \land s_{[axj]}\) can be minimized by ComputeOneMinA to compute one novel MinA. Notice that such MinA is necessarily novel, because the negation of all previous MinAs have been learned permanently, so that, if \(\mu^+ \land s_{[axj]}\) contained a previous MinA, then \(\neg s_{[axj]}\) would have been unit-propagated before invoking ComputeOneMinA.

**Example 4.3.** We consider a slight variant of Example 3.4 at p. 29. We want to find all the MinAs for the subsumption Milk \(\sqsubseteq\text{Fluid}\) in the ontology \(\mathcal{O}_{\text{milk}}\) of Example 2.2 at p. 8, using the encoding \(\phi^{all}_{\mathcal{O}_{\text{milk}}(po)}\) of Example 3.3 at p. 23. The query above is labeled by \(s_{[n_2]}\). Suppose ComputeAllMinAs enumerates the partial assignment \(\mu = \mu^+ \overset{\text{def}}{=} \{s_{[m_0]}, s_{[m_1]}, s_{[n_2]}, s_{[n_3]}, s_{[n_4]}, s_{[m_5]}\}\), and ComputeOneMinA is invoked on \(\phi^{all}_{\mathcal{O}_{\text{milk}}(po)}\), \(\mu^+\) and \(\neg s_{[n_2]}\). This causes the unit-propagation on \(\phi^{all}_{\mathcal{O}_{\text{milk}}(po)}\) of \(s_{[n_1]}\), \(\neg s_{[n_3]}\), \(\neg s_{[n_2]}\), and hence \(\neg s_{[m_6]}\). Thus \(\mu \cup \{s_{[m_0]}\}\) invalidates \(\neg s_{[n_2]} \land \phi^{all}_{\mathcal{O}_{\text{milk}}(po)}\), and is then minimized by ComputeOneMinA into the set \(\eta = \{s_{[m_0]}, s_{[m_1]}, s_{[n_3]}, s_{[n_5]}, s_{[m_6]}\}\) (corresponding to the MinA \(\mathcal{O}_{\text{milk}}^*\) in Example 3.4 at p. 29). Then ComputeAllMinAs learns permanently the clause \(\neg \eta\) and adds \(\neg s_{[m_6]}\) to \(\mu\) by unit-propagation, and the search proceeds.

### 4.3 Working on Smaller Ontologies: \(\mathcal{EL}^+\text{SAT} \times 2\)

We have already noticed that COI modules are typically much smaller, up to orders of magnitude smaller, than the original ontologies. This suggest our last enhancement, which separates the modularization and the enumeration of all MinAs into two distinct executions of \(\mathcal{EL}^+\text{SAT}\) on two different formulas. We first exploit COI modularization to extract a query-dependent sub-ontology on which, subsequently, we run our complete \(\mathcal{EL}^+\text{SAT}\) approach (i.e. both the encoding and the all-MinAs search phases).

Given a query \(C_i \sqsubseteq D_i\), the process, which we call \(\mathcal{EL}^+\text{SAT} \times 2\), consists in three phases:

1. after loading the formula \(\phi^{all}_{T(p_o)}\), extract the query-dependent sub-ontology \(\mathcal{T}_i \overset{\text{def}}{=} \mathcal{M}^{\mathcal{C}_{C_i \sqsubseteq D_i}}\) by invoking ComputeCOIModule;

2. build from scratch a new classification formula \(\phi^{all}_{T_i(p_o)}\) from the sub-ontology \(\mathcal{T}_i\);

3. enumerate all MinAs applying ComputeAllMinAs to \(\phi^{all}_{T_i(p_o)}\) rather than to \(\phi^{all}_{T(p_o)}\).

Implementation-wise, it is convenient to run two distinct instances of \(\mathcal{EL}^+\text{SAT}\):
(i) a first instance implementing Step 1., which loads only once $\phi^{all}_{T_{(po)}}$, and then, for every query $C_i \subseteq T \ D_i$, given as input, performs COI Modularization, producing $T_i \overset{\text{def}}{=} \mathcal{M}^{\phi^{all}_{T_{(po)}}}_{\mathcal{C}_i} T_i$;

(ii) a second “dynamic” instance of $\mathcal{E}\mathcal{L}^{+}\text{SAT}$ implementing Steps 2. and 3. for each sub-ontology $T_i$, producing $\phi^{all}_{T_i \ (po)}$ and then using it to enumerate all MinAs for $C_i \subseteq D_i$.

We remark a few facts. First, with the first $\mathcal{E}\mathcal{L}^{+}\text{SAT}$ instance (i), once all $T_{(po)}$ has been loaded, a COI module $T_i$ can be computed in negligible time for an unlimited number of queries $C_i \subseteq D_i$, with no need for reloading $\phi^{all}_{T_{(po)}}$. Second, since each sub-ontology $T_i$ is typically much smaller than the original ontology $T$ (e.g., up to a hundred axioms w.r.t. up to hundreds of thousands axioms), the time taken by the second instance of $\mathcal{E}\mathcal{L}^{+}\text{SAT}$ (ii) to produce the new formulas $\phi^{all}_{T_i \ (po)}$ is typically negligible. Third, overall, $\mathcal{E}\mathcal{L}^{+}\text{SAT} \\times 2$ allows for a faster enumeration of the MinAs. In fact, while phases 1. and 2. are expected to take negligible time even for really huge input ontologies, in phase 3. the second instance of $\mathcal{E}\mathcal{L}^{+}\text{SAT}$ runs over the classification of $\approx 10/100$ axioms instead of $\approx 10,000/100,000$ axioms. This can strongly reduce the overhead of every internal call to $\text{ComputeOneMinA}$ during the MinAs enumeration, leading to a drastic enhancement overall.

5. An Extensive Experimental Evaluation

We have implemented in C++ the procedures of Sections 3 and 4 into a tool —hereafter referred as $\mathcal{E}\mathcal{L}^{+}\text{SAT}$— which includes a customized versions the SAT solver MiniSat2.0 070721 (Déï & Sörensson, 2004). We have performed a very extensive experimental evaluation on the five ontologies introduced in Section 1: SNOMED-CT’09, NCI, GENEONTOLOGY, NOT-GALEN and FULL-GALEN. The latter two ontologies are derived from GALEN: NOT-GALEN is a stripped-down version of GALEN with no role functionality that has been widely used for benchmarking several standard DL-reasoners, and FULL-GALEN represents the full GALEN medical ontology, excluding role inverses and functionalities. In Table 3 we present some numerical informations on the structure and size of these ontologies.

For all these ontologies, we have compared different versions of $\mathcal{E}\mathcal{L}^{+}\text{SAT}$ and the other $\mathcal{E}\mathcal{L}^{+}$-specific tool CEL (Baader et al., 2006a), version v.1.1.2, which implements also the Reachability-based Modularization of Suntisirivaraporn (2009). All tests have been run on a biprocessor dual-core machine Intel Xeon 3.00GHz with 4GB RAM on Linux RedHat 2.6.9-11, except for SNOMED-CT which has been processed by both $\mathcal{E}\mathcal{L}^{+}\text{SAT}$ and CEL tools on a Intel Xeon 2.66 GHz machine with 16GB RAM on Debian Linux 2.6.18-6-amd64. For each query, we have set a timeout of 1000 seconds.

The whole evaluation was performed in two steps.

In the short version of this paper (Sebastiani & Vescovi, 2009b) we presented, implemented and evaluated only the procedures of Section 3, without the optimizations of Section 4, and presented a preliminary evaluation. There we reported the CPU times of the formula encoding and parsing phases, the size of the encoded formulas, the CPU times

18. and, possibly, for every sub-ontology $\mathcal{S} \subseteq T$.

19. SNOMED-CT’09 is courtesy of IHTSDO http://www.ihtsdo.org/, whilst the other four ontologies are available at http://lat.inf.tu-dresden.de/~meng/toyont.html.

taken to check subsumptions and to compute single MinAs, and some preliminary results on all-MinAs problems. As a result, the amounts of CPU time required by subsumption checks and single-MinA computation were negligible (< 0.02s on average); unfortunately, for the All-MinA problems, for no sample \( \mathcal{EL}^+ \)SAT could conclude the full enumeration within a timeout of 1000s. (In order not to further enlarge this experimental section, we refer the reader to (Sebastiani & Vescovi, 2009b) for these data.)

In this paper we have focused on the All-MinA problems, and we have extended \( \mathcal{EL}^+ \)SAT with the optimizations presented in Section 4. For all the five ontologies, we have compared different versions of \( \mathcal{EL}^+ \)SAT and CEL.

Note for reviewers. The tarballs containing the tool \( \mathcal{EL}^+ \)SAT, the encodings and the data of this empirical evaluation are available from http://disi.unitn.it/~rseba/elsat/.

5.1 Evaluation criteria

To perform an accurate performance comparison between \( \mathcal{EL}^+ \)SAT in its various versions and CEL is very problematic for some reasons. First, the most-recent version of CEL is implemented as a Java plug-in for the Protege ontology editor and knowledge acquisition system,\(^{21}\) and as such it is practically impossible to have a performance comparison against it. Second, the available stand-alone version of CEL, version v.1.1.2, stops as soon as it reports 10 MinAs.\(^{22}\) Third, all the MinAs found by CEL are reported only after the whole search is terminated, so that it is not possible to interrupt the execution of CEL at the expiration of the timeout and obtain the partial results; thus, in order to make a comparison possible, we chose to run CEL without a timeout. Notice that, in this case, it is also impossible to know how many of the MinAs returned have been actually enumerated within the timeout.

Thus, we adopt the following criterium for comparing \( \mathcal{EL}^+ \)SAT versions and CEL.

Definition 6. Let A and B be either one \( \mathcal{EL}^+ \)SAT version or CEL. We say that a tool A **performs better** than a tool B iff one of the following alternative facts happens:

(i) both A and B terminate without being stopped by the timeout or by the 10 MinAs limit for CEL, and A takes less time, or

(ii) A terminates within the timeout/limit and B does not, or

\(^{21}\) http://protege.stanford.edu/

\(^{22}\) We have reported this problem to the authors long ago, but so far they were not able to provide us with a fixed stand-alone version.
(iii) neither terminates within the timeout/limit and A enumerates more MinA’s than B, or
(iv) neither terminates within the timeout/limit, they enumerate the same amount of MinA’s,
    and A takes less time in doing it.

Notice that in case (i) A and B return the same MinAs, whilst in case (ii) A returns a
non-strict superset of MinAs than B does. We reckon that the criterium of Definition 6 is
not accurate in some situations, since the notion of “non-termination” for $EL^+SAT$ (1000s
timeout) and $CEL$ (10 MinAs reached) differ. Unfortunately, the 10-MinA limit of $CEL$
and the fact that it is not possible to evaluate the actual number of MinAs evaluated by $CEL$
within a given amount of time prevented us from conceiving a more fine-grained criterium.

We also reckon that the results depend on the values of the two constants used. Of
course, we are forced to take the 10MinA limit as is. As far as the 1000s timeout is con-
cerned, we thought this value, which we have adopted in many other papers, is a reasonable
compromise. (We will further discuss the latter issue later on.)

Overall, we believe that the values reported can provide at least a coarse-grained idea
of the relative performance of the two tools.

5.2 Test description and results

For each ontology $T$ we run two groups of 50 test pinpointing queries each, extracted among
the subsumption relations which are deducible from $T$, so that there exists at least one
MinA for every query. A first group of queries, which we call random, has been generated
by picking 50 random queries among all the possible existing and non-trivial ones; this is
done in order to understand the behavior of the tool on a normal debugging circumstance.
A second group of queries, which we call selected, has been chosen by considering the 50
subsumptions $C_i \subseteq D_i$ whose selector variable $s_{[C_i \subseteq D_i]}$ appears positively more frequently
in $\phi_{T_{(p_o)}}^\text{eff}$; this is done in order to understand the behavior of the tool on these “potentially
harder” queries, that is, queries potentially involving the highest number of different MinAs.

In what follows we present and discuss the results of this evaluation, reporting both
the CPU times taken by the tools to compute the generated MinAs and the number of
MinAs found. We report also some data concerning the size of the modules obtained by
applying the COI Modularization for $EL^+SAT$ (see Section 4.1) and the Reachability-based
Modularization for $CEL$. It is worth noticing that our modularization technique required
negligible time ($\approx 10^{-2}$ secs.) in all the test cases we performed.

In the evaluation we executed the following four different versions of $EL^+SAT$, which
increasingly introduce the optimizations exposed in Section 4:

- **BASIC (b)**: the first prototype version used in the feasibility evaluation presented by Sebas-
tiani and Vescovi (2009b) (without the optimizations of Section 4);
- **COI (c)**: a second version implementing the COI Modularization introduced in Section 4.1;
- **COI&TP (t)**: the COI version enhanced with Theory Propagation (Section 4.2) and other
  minor optimizations;
- **$EL^+SAT \times 2 (\times 2)$**: a final version implementing the “two-instances” approach of Section 4.3,
  which runs two instances of $EL^+SAT$ COI&TP.
Table 4: Number of instances for which each tool has been the best performer.

<table>
<thead>
<tr>
<th></th>
<th>SNOMED'09 ran.</th>
<th>SNOMED'09 sel.</th>
<th>FULLGALEN ran.</th>
<th>FULLGALEN sel.</th>
<th>NCI ran.</th>
<th>NCI sel.</th>
<th>GENEONT. ran.</th>
<th>GENEONT. sel.</th>
<th>NOTGALEN ran.</th>
<th>NOTGALEN sel.</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\mathcal{EL}^+)SAT BASIC</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(\mathcal{EL}^+)SAT coi</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(\mathcal{EL}^+)SAT coi&amp;TP</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(\mathcal{EL}^+)SAT x2</td>
<td>37</td>
<td>24</td>
<td>32</td>
<td>3</td>
<td>49</td>
<td>49</td>
<td>50</td>
<td>50</td>
<td>35</td>
<td>30</td>
<td>359</td>
</tr>
<tr>
<td>CEL</td>
<td>13</td>
<td>22</td>
<td>18</td>
<td>47</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>15</td>
<td>20</td>
<td>136</td>
</tr>
<tr>
<td>Total</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>500</td>
</tr>
</tbody>
</table>

In Tables 6–15—which for lack of space we have moved into Appendix, Section B, Pages ix–xviii—we report in detail the results of each single query for the random/selected groups of the SNOMED-CT’09, FULL-GALEN, NCI, GENEONTOLOGY, NOT-GALEN respectively. In these tables we compare the detailed results of the four representative variants of \(\mathcal{EL}^+\)SAT and those of CEL. For each query and variant/tool we expose the results of modularization, the CPU times required by the tools and the number of MinAs resulting from the search.

We remark that the results are consistent: whenever completing the enumeration, the numbers of MinAs pinpointed by the different versions of \(\mathcal{EL}^+\)SAT and/or by CEL are always the same. Terminating versions always discover a greater or equal number of MinAs w.r.t. non-terminating versions (or w.r.t. CEL, when it stops at 10 MinAs).

The results of Tables 6-15 are first briefly summarized in Table 4 in terms of best performers, as defined in Section 5.1. A more detailed summary is then presented in Table 5, in terms of numbers of problems terminated and numbers of MinAs found (Table 5(a)) and statistics on CPU Times (Table 5(b)).

Table 4, which reports the number of instances in each category for which each tool was the best performer, is self-explanatory. (We remark that the “\# of MinAs found” columns in Table 4 refer to the total amount of MinAs accumulated over the 50 tests.) Table 5 deserves some explanation.

Consider Table 5(a). In the left-most block of the table we count, for every variant, the number of \(\mathcal{EL}^+\)SAT computations terminating the enumeration before the timeout; for CEL we indicate the number of instances exhaustively terminating the search (i.e. not stopping at 10 MinAs). In the central block we count the total amount of MinAs generated before termination. In the right-most block we present the average size of the modules extracted by \(\mathcal{EL}^+\)SAT and CEL, respectively.

In Table 5(b) we summarize time performances. For every value we compute both the 50th and the 90th percentiles. In the left-most block we report the statistics concerning either the time taken by \(\mathcal{EL}^+\)SAT to discover the first MinA or the time between each pair of consecutive MinA detections. In the central block we report, instead, the statistics on total elapsed time. Finally, in the right-most block, we report the statistics concerning the modularization procedures of \(\mathcal{EL}^+\)SAT and CEL, respectively. (Since with CEL it is not possible to directly distinguish between the time taken by the modularization procedure and the time required by the following search, we measured such times by running in CEL separate modularization queries.) Notice that the time statistics for CEL include also problems terminated after the timeout or because of the detection of 10 MinAs.
<table>
<thead>
<tr>
<th>test</th>
<th># terminated (b)</th>
<th># MinAs found (b)</th>
<th>Module size avg. (b)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(\mathcal{L}^+)SAT</td>
<td>CEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(c) (T) (×2)</td>
<td>(b) (c) (T) (×2)</td>
</tr>
<tr>
<td>SNOMED'09</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random</td>
<td>0</td>
<td>23</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>selected</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FULLGalen</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random</td>
<td>0</td>
<td>10</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>selected</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NCI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random</td>
<td>0</td>
<td>40</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>selected</td>
<td>0</td>
<td>29</td>
</tr>
<tr>
<td>GENEONT.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random</td>
<td>0</td>
<td>49</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>selected</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>NOTGalen</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random</td>
<td>0</td>
<td>17</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>selected</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TOTAL</td>
<td>0</td>
<td>185</td>
<td>296</td>
</tr>
</tbody>
</table>

(a) Number of terminated problems and generated MinAs.

<table>
<thead>
<tr>
<th>test</th>
<th>Time between MinAs (s)</th>
<th>Total pinpointing time (s)</th>
<th>Modulariz. (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(\mathcal{L}^+)SAT</td>
<td>CEL</td>
<td>(\mathcal{L}^+)SAT</td>
</tr>
<tr>
<td></td>
<td>(b) (c) (T) (×2)</td>
<td>(b) (c) (T) (×2)</td>
<td>(b) (c) (T) (×2)</td>
</tr>
<tr>
<td>SNOMED'09</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random 50th</td>
<td>7.4</td>
<td>0.5</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>selected 50th</td>
<td>7.2</td>
<td>0.6</td>
</tr>
<tr>
<td>random 90th</td>
<td>83.3</td>
<td>13.2</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>selected 90th</td>
<td>11.1</td>
<td>31.3</td>
</tr>
<tr>
<td>FULLGalen</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random 50th</td>
<td>3.9</td>
<td>0.4</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>selected 50th</td>
<td>3.7</td>
<td>0.3</td>
</tr>
<tr>
<td>random 90th</td>
<td>17.6</td>
<td>0.6</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>selected 90th</td>
<td>3.8</td>
<td>0.4</td>
</tr>
<tr>
<td>NCI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random 50th</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>selected 50th</td>
<td>0.3</td>
<td>0.0</td>
</tr>
<tr>
<td>random 90th</td>
<td>9.1</td>
<td>6.9</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>selected 90th</td>
<td>3.1</td>
<td>1.0</td>
</tr>
<tr>
<td>GENEONT.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random 50th</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>selected 50th</td>
<td>0.3</td>
<td>0.0</td>
</tr>
<tr>
<td>random 90th</td>
<td>8.5</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>selected 90th</td>
<td>0.3</td>
<td>0.0</td>
</tr>
<tr>
<td>NOTGalen</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random 50th</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>selected 50th</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>random 90th</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>selected 90th</td>
<td>0.2</td>
<td>0.0</td>
</tr>
</tbody>
</table>

(b) CPU times in pinpointing and modularization.

Table 5: Summary results of \(\mathcal{L}^+\)SAT (all versions) and CEL on all the test problems.
5.3 Analysis of the Results

Here we discuss the results of our extensive experimental evaluation. First, we analyze the general performance of $\mathcal{EL}^+$SAT, comparing its different variants. Second, we evaluate the most-enhanced version of $\mathcal{EL}^+$SAT in comparison with CEL.

5.3.1 Analysis of $\mathcal{EL}^+$SAT Performances

As a general fact (see the last “Total” row in Table 5(a)) from all the data in the tables we can immediately notice that each enhancement in $\mathcal{EL}^+$SAT improves its overall performances, increasing both the total number of terminating test cases and the total number of MinAs found within the timeout. In particular, from Table 4 we notice that, with the exception of only 5 instances out of 500, $\mathcal{EL}^+$SAT $\times 2$ is always the best performer among the $\mathcal{EL}^+$SAT versions. (The very-rare exceptions — e.g., the 24th, 36th, 38th and 42nd SNOMED’09 selected instances in Table 7 in Appendix — can be easily justified by pure chance, due to different enumeration orders induced to the SAT solvers.) The improvements are typically much more evident when the test case includes a (relatively) large number of MinAs. (In these cases, e.g., techniques like Theory Propagation yield more benefits and have their computational cost amortized.)

More specifically, looking at Tables 4 and 5 (plus Tables 6-15 in Appendix) we notice the following facts concerning the four $\mathcal{EL}^+$SAT versions:

- **COI Modularization** dramatically improves $\mathcal{EL}^+$SAT performances, requiring negligible overhead time. This fact can be easily observed by comparing the results of the (b) and (c) versions in Table 5(a). In particular, due to the number of axioms in the input ontologies (see Table 3) no problem can be completely enumerated by the basic version, whilst in the coi version COI modularization allows for completing 185 enumerations, generating 416 more MinAs overall.

- **Theory propagation** significantly increases the number of generated MinAs. As can be seen comparing the (c) and (t) variants of $\mathcal{EL}^+$SAT in Table 5(a), theory propagation allows for discovering 364 more MinAs overall the five benchmark ontologies, (275 of which in SNOMED-CT), also leading to 24 complete enumerations more. In fact early-discovering MinAs speeds up the search by introducing earlier the respective blocking clauses).

- The ($\times 2$) enhancement also much improves the performances under every perspective (number of discovered MinAs, number of terminated problems and CPU times): w.r.t. the (t) version, the ($\times 2$) one terminates the enumeration on 87 more problems and returns more than 671 more MinAs, almost doubling the number of MinAs pinpointed for SNOMED-CT. Importantly, the CPU time required in the first two phases of the approach (modularization and encoding of the sub-ontology) is negligible ($\leq 10^{-1}$) for every query and ontology (SNOMED-CT and FULL-GALEN included).

- The sub-ontologies (modules) extracted are orders of magnitude smaller than the original ones. This is one key factor of the drastic improvements in performances from version (b) to version ($\times 2$). In particular, with $\mathcal{EL}^+$SAT $\times 2$, every single internal call to the $\mathcal{T}$-solver during the enumeration is performed on a SAT formula.
that is many orders of magnitude smaller. This acts on reducing the overheads of every single search, leading globally to a much faster enumeration.

Overall, looking at the performances of the most-enhanced version, $\mathcal{E}\mathcal{L}^+\text{SAT} \times 2$, we notice a few facts:

- $\mathcal{E}\mathcal{L}^+\text{SAT} \times 2$ cuts down to almost zero the CPU time required to find all the existing MinAs in the very majority of the test cases. (See column “($\times 2$)” in Tables 6-13.) Moreover, whichever the ontology or the query is, $\mathcal{E}\mathcal{L}^+\text{SAT} \times 2$ allows for computing some MinAs in negligible time.

- The performances reached by $\mathcal{E}\mathcal{L}^+\text{SAT} \times 2$ for an “average” query (represented by the random test suites) are quite satisfactory. Comparing the 50th and 90th percentiles of Table 5(b) we can notice that, even if there are some really challenging queries, the greatest part of the queries are easily affordable by $\mathcal{E}\mathcal{L}^+\text{SAT} \times 2$.

5.3.2 Comparing $\mathcal{E}\mathcal{L}^+\text{SAT} \times 2$ with CEL

Comparing the data out of the 500 samples on $\mathcal{E}\mathcal{L}^+\text{SAT} \times 2$ and CEL in Tables 4 and 5 (plus Tables 6-15 in Appendix B, p. ix-xviii) we notice the following facts:

- $\mathcal{E}\mathcal{L}^+\text{SAT} \times 2$ is the best performer on 359 samples, whilst CEL is the best performer on 136 samples (see Table 4). These results are heterogeneous: $\mathcal{E}\mathcal{L}^+\text{SAT} \times 2$ performs worse than CEL on FULL-GALEN, better on SNOMED’09 and NOT-GALEN, and drastically better on NCI and GENEONTOLOGY. The results vary a lot also in terms of the cases (i)-(iv) of our comparison criterium of Section 5.1.

- $\mathcal{E}\mathcal{L}^+\text{SAT} \times 2$ terminates within the timeout on 296 samples, whilst CEL terminates within the timeout/10MinA limit on 379 samples (see Table 5(a)). However (see last two columns in Tables 6-15) even when not terminating within the timeout, $\mathcal{E}\mathcal{L}^+\text{SAT} \times 2$ only on 6 samples (respectively 1, 4, and 1 in Tables 6, 7, and 8); vice versa, $\mathcal{E}\mathcal{L}^+\text{SAT} \times 2$ reports more MinAs than CEL on 102 samples (respectively samples 9, 25, 5, 16, 4, 43, on Tables 6, 7, 10, 11, 12, 13). Overall (Table 5(a)) $\mathcal{E}\mathcal{L}^+\text{SAT} \times 2$ finds 3433 MinAs against the 1927 MinAs found by CEL.

- When both $\mathcal{E}\mathcal{L}^+\text{SAT} \times 2$ and CEL terminate (see the topmost partition of Tables 6-13) $\mathcal{E}\mathcal{L}^+\text{SAT} \times 2$ always outperforms CEL in time, by almost-always detecting all MinAs in negligible time ($\leq 0.1s$) whilst CEL takes, e.g., $\approx 20s$ with SNOMED-CT and up to $\approx 900s$ with FULL-GALEN.

Remark 4. In Section 5.1 we noticed that the results can be sensitive to the values of the two constants used. Thus we might conjecture that lowering the timeout would have shifted results in favour of CEL. We claim this is not really the case. For instance, by looking at the “($\times 2$)” and “CEL” columns in Tables 6-13 we notice that $\mathcal{E}\mathcal{L}^+\text{SAT} \times 2$ times exceed 100s only on 26 instances, whilst CEL exceeds it with 72 instances (most of which in FULL-GALEN).
To sum up, typically $\mathcal{EL}^+\text{SAT} \times 2$ takes less time than CEL to enumerate the MinAs, and then it takes more time than CEL to show there is no other MinA left afterwards. This fact mirrors the differences in the enumeration techniques performed by the two tools we have described in previous sections; in particular, it reflects the fact that, thanks to the strategy of selecting positive axiom selector variables (see Section 3.3) and to Theory Propagation (Section 4.2), $\mathcal{EL}^+\text{SAT} \times 2$ tends to find the MinAs at the early stage of the Boolean search, and then to spend most time to complete the search to certify there is no MinA left.

With both tools, modularization plays an essential role. In particular, comparing the two modularization techniques, we notice the following facts.

- The data on module size (see the last two columns of Table 5(a) and the first two columns of Tables 6-13) confirm empirically what formally proved in Proposition 8: COI Modularization of $\mathcal{EL}^+\text{SAT}$ is more effective than CEL’s syntactic modularization, always producing smaller modules. Notice that this size gap is up to 2-3 order magnitude with FULL-GALEN (Tables 8, 9).  
- COI Modularization of $\mathcal{EL}^+\text{SAT}$ is much faster than the Reachability-based Modularization of CEL, even when the module extracted are comparable in size (see the last two columns of Table 5(b)). In particular, the cost of COI Modularization is linear w.r.t. the size of the cone of influence for the query (see Section 4.1), whilst that of Reachability-based Modularization seems to grow with the size of the whole input ontology.

6. Related Work.

In this section we survey the main related work, from different perspectives.

**Concept Subsumption, Classification and Axiom Pinpointing in $\mathcal{EL}^+$.** Apart from the preliminary version of this paper (Sebastiani & Vescovi, 2009b), the most closely-related work to our own is that of Baader and colleagues (see e.g. Baader et al., 2006b, 2007; Baader & Suntisrivaraporn, 2008), who thoroughly investigated the problems of concept subsumption, classification and axiom pinpointing in $\mathcal{EL}^+$, and developed efficient algorithms for debugging real-world $\mathcal{EL}^+$ ontologies, which were implemented and tested with success on large real-world ontologies, including SNOMED-CT.

**Black-Box Debugging Techniques for General Ontologies.** Axiom pinpointing has increasingly caught attention in the Semantic Web research community, as a fundamental method for debugging and repair ontologies in a variety of logics (see, e.g., Kalyanpur, Parsia, Horridge, & Sirin, 2007; Suntisrivaraporn et al., 2008; Horridge, Parsia, & Sattler, 2008; Peñaloza & Sertkaya, 2010a; Moodley, Meyer, & Varzinczak, 2011; Horridge, 2011; Bail, Horridge, Parsia, & Sattler, 2011). In this domain axiom pinpointing —here usually called justification finding— is mainly solved via black-box approaches, which use blind search algorithms, that is, algorithms which are not driven by properties of the specific logic/ontology (Schlobach & Cornet, 2003; Kalyanpur, Parsia, Sirin, & Hendler, 2005; Kalyanpur et al., 2007; Suntisrivaraporn et al., 2008; Horridge et al., 2008). More recent works (e.g., Moodley et al., 2011; Horridge, 2011; Bail et al., 2011), explored more sophisticated structure-aware approaches also in the case of black-box debugging of general
ontologies, in order to cope with multiple undesired entailments and with multiple justifications, improving the debugging support by identifying the best/minimal repair strategies.

Modularization. In order to cope with the huge dimension of many ontologies such as SNOMED-CT, and to better support the different ontology-manipulation tasks, a lot of effort has been spent in devising efficient and effective module-extraction procedures, to be combined with the various reasoning services (e.g., Grau et al., 2007; Konev et al., 2008b, 2008a; Suntisrivarporn et al., 2008; Del Vescovo, Parsia, Sattler, & Schneider, 2010, 2011), so that modularization has become a prominent research problem.

Reasoning on Other Tractable Extensions of $\mathcal{EL}$. Beside the logic $\mathcal{EL}^+$ (Baader et al., 2006b), many other extension of $\mathcal{EL}$ or tractable fragments of harder logics have been studied by members of the Description Logic community, with the aim of defining a maximal subset of logical constructors whose practical applications remain tractable (Baader et al., 2005; Baader, Brandt, & Lutz, 2008; Kazakov, 2009; Magka, Kazakov, & Horrocks, 2010; Kazakov, Kroetzsch, & Simancik, 2012). A witness of the relevance acquired by the $\mathcal{EL}$ family of description logics is the introduction in the new W3C standard OWL 2 of the specific OWL 2 EL profile 23 and the concurrent creation of many reasoners for handling and manipulating OWL EL ontologies, such as CEL (Baader et al., 2006a; Suntisrivarporn, 2009; Mendez & Suntisrivarporn, 2009), Snorocket (Lawley & Bousquet, 2010), jCEL (Mendez, Ecke, & Turhan, 2011) and ELK (Kazakov et al., 2012; Kazakov & Krötzsch, 2013).

SAT- and SMT-based Reasoning Techniques for Description Logics. The idea of implementing automated reasoners for description logics based on nested calls to a SAT solver was introduced by Giunchiglia and Sebastiani (1996, 2000). Reasoning techniques for the description Logics $\mathcal{ALC}$ and $\mathcal{ALCQ}$, which are based instead on direct encodings into SAT and SMT, have been presented by Sebastiani and Vescovi (2009a, 2009b) and Haarslev, Sebastiani, and Vescovi (2011) respectively.

Unsatisfiable-core Extraction, All-SMT and COI Reduction. The idea of computing single MinAs by exploiting the conflict-analysis techniques of modern CDCL SAT solvers was inspired by the technique for unsatisfiable-core extraction in SAT by Lynce and Silva (2004). The idea for All-MinA enumeration was inspired by the All-SMT technique for computing predicate abstractions in Model Checking by Lahiri et al. (2006). The idea of COI Modularization was inspired by the COI Reduction technique widely used in Symbolic Model Checking (see e.g. Clarke et al., 1999).

7. Conclusions

We have presented a novel approach for axiom pinpointing in the logic $\mathcal{EL}^+$ and its sublogics, which is based on a SAT encoding of the classification of the input ontology, and which exploits an ad-hoc combination of modern SAT and SMT techniques. This approach also strongly benefits from a novel SAT-based modularization technique we have also proposed here.

We have implemented this novel approach into a tool, $\mathcal{EL}^+$SAT, and we have showed in an extensive empirical evaluation that $\mathcal{EL}^+$SAT is extremely efficient in exhaustively enu-

23. http://www.w3.org/TR/owl2-profiles/
Axiom Pinpointing in large $\mathcal{EL}^+$ ontologies via SAT and SMT techniques

merating the minimal sets of axioms leading to (typically undesired) subsumption relations, even with huge medical ontologies like SNOMED-CT.

References


Axiom Pinpointing in large $\mathcal{EL}^+$ ontologies via SAT and SMT techniques


Appendix A. Appendix: Proofs

In this section we define and prove formally all the results stated along this work

**Theorem 1.** Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form, for every pair of concept names $C, D$ in $\mathcal{PC}_\mathcal{T}$, $C \sqsubseteq T D$ if and only if the Horn propositional formula $\phi_\mathcal{T} \land p[C] \land \neg p[D]$ is unsatisfiable.

**Proof.** (Only if.) By definition the clause $p[C] \rightarrow p[D]$, that is $\mathcal{EL}^+2sat(C \sqsubseteq D)$, is in $\phi_\mathcal{T}$ if and only if $C \sqsubseteq D$ belongs to $\mathcal{A}$ that is, since $\mathcal{A}$ is the classification of $\mathcal{T}$, if and only if $C \sqsubseteq T D$. Thus, if $C \sqsubseteq T D$ then $\phi_\mathcal{T} \land p[C] \land \neg p[D]$ is unsatisfiable.

(If.) Vice versa, if $\phi_\mathcal{T} \land p[C] \land \neg p[D]$ is unsatisfiable, then it follows that $\phi_\mathcal{T} \models p[C] \rightarrow p[D]$. Thus, there must be a resolution derivation $P$ of $p[C] \rightarrow p[D]$ from some subset of clauses $\psi_1, \ldots, \psi_n$ in $\phi_\mathcal{T}$. Since $\phi_\mathcal{T}$ is a definite Horn formula, every resolution step can be written in the form

$$
\frac{(\bigwedge_i p[x_i]) \rightarrow p[x_k] \quad (p[x_k] \land \bigwedge_j p[x_j]) \rightarrow p[x_n]}{(\bigwedge_i p[x_i] \land \bigwedge_j p[x_j]) \rightarrow p[x_n]}
$$

s.t. all $X$’s are concepts. Each corresponding derivation step

$$
\frac{\prod_i X_i \subseteq X_k \quad (X_k \cap \prod_j X_j) \subseteq X_n}{\prod_i X_i \cap \prod_j X_j \subseteq X_n}
$$

is valid in $\mathcal{EL}^+$, $^{24}$ Moreover, by definition, each $\psi_i$ is $\mathcal{EL}^+2SAT(a_i)$ for some $a_i$ which is either in $\mathcal{T}$ or has been derived from $\mathcal{T}$. Hence, there is a valid derivation of $C \sqsubseteq D$ from $\mathcal{T}$ in $\mathcal{EL}^+$, so that $C \sqsubseteq T D$. \qed

**Proposition 2.** The size of the formula $\phi_\mathcal{T}^{all}$ defined in Definition 4 is worst-case polynomial in the size of the TBox $\mathcal{T}$.

**Proof.** Under the assumption that conjunctions and role compositions are binary, every completion rule instantiation (Table 2) has at most three antecedents: one axiom and two assertions. Thus, the number of different rule instantiations $r(a_i, a_j, a_j, a_k)$ on the axioms and assertions in $\mathcal{A}$ is upper-bounded by $|\mathcal{A}|^2 \cdot |\mathcal{T}|$. Recalling that $|\mathcal{A}| \leq |\mathcal{PC}_\mathcal{T}|^2 \cdot |\mathcal{PR}_\mathcal{T}|$ and hence $|\mathcal{A}|$ is polynomial in $|\mathcal{T}|$, we have the thesis. \qed

**Lemma 9.** A literal $\ell$ can be derived by unit-propagation from $\phi_\mathcal{T}^{all} \land \bigwedge_{ax \in S} s_{[ax]}$ [resp. $\phi_\mathcal{T}'^{all} \land \bigwedge_{ax \in S} s_{[ax]}$] if and only if $\ell$ is a positive occurrence of a selector variable $s_{[ax]}$ for some subsumption relation $a_k$ which is deducible from $S$.

**Proof.** (If.) If the subsumption relation $a_k$ is deducible from $S$ then there is a chain of rule applications $r_1, \ldots, r_N$ which allows for deriving $a_k$ starting from a set of premises $S' \subseteq S$, and such that $r_N = r(a_i, a_j, a_j, a_k)$ for some rule $r$. We reason by induction on the number $N$ of rules applied in order to deduce $a_k$ from $S'$.

$^{24}$ Notice that this statement is purely based on $\mathcal{EL}^+$ semantic, meaning “if an interpretation satisfies the premises, it satisfies also the conclusion”, and does not depend on the set of derivation rules used.
Base: If \( N = 1 \) then \( a_i, a_{i'} \) and \( a_j \) must all be axioms in \( S' \). By construction, the rule clause (10) \( c_{r_N} = (s_{[a_i]} \land s_{[a_{i'}]} \land s_{[a_j]}) \rightarrow s_{[a_k]} \) is included in \( \phi_T^{all} \), and thus \( s_{[a_k]} \) is derived by unit-propagation from \( \phi_T^{all} \land \bigwedge_{ax \in S} s_{[ax]} \) where all the axiom selector variables relative to the axioms of \( S \) are propagated.

Step: We assume that \( a_k \) has been derived from the rule applications \( r_1, \ldots, r_{N-1}, r_N \). So, by construction, \( \phi_T^{all} \) includes the corresponding set of rule clauses (10) \( c_{r_1}, \ldots, c_{r_N} \), where \( c_{r_N} = (s_{[a_i]} \land s_{[a_{i'}]} \land s_{[a_j]}) \rightarrow s_{[a_k]} \) and \( a_i, a_{i'}, a_j \) are either axioms of \( S' \) or subsumption relations consequence of one of the rules among \( r_1, \ldots, r_{N-1} \). In both cases all the selector variables \( s_{[a_i]}, s_{[a_{i'}]} \) and \( s_{[a_j]} \) are positively unit-propagated by inductive hypothesis. Thus \( s_{[a_k]} \) is also positively unit-propagation from \( c_{r_N} \).

Notice also that, since \( \phi_T^{all} \) is a subformula of \( \phi_T^{all} \), every literal that is unit-propagated in the first is also unit-propagated in the second.

(Only if.) We reason by induction on the number of unit-propagation steps performed on \( \phi_T^{all} \land \bigwedge_{ax \in S} s_{[ax]} \):

Base: By definition, all the axiom selector variables \( s_{[ax]} \) such that \( ax \in S \) are initially unit-propagated. Since they are axioms of \( S \) they are trivially deducible from \( S \).

Step: Let \( L \) be the set of all literals derived by \( N \) applications of unit-propagation steps and assume that all literals in \( L \) are in the form \( s_{[ax]} \). Since only assertion clauses (9) contain concept variables \( p(X) \) (for some concept \( X \)) and each assertion clause contains at least two of them, then: (i) no concept variables can be unit-propagated, and (ii) no assertion clause causes unit-propagations, but only rule clauses (10) (all included in the subformula \( \phi_T^{all} \)) do. Thus only positive selection variables \( s_{[a_k]} \) can be propagated, and only from rule clauses (10) in the form \( (s_{[a_i]} \land s_{[a_{i'}]} \land s_{[a_j]}) \rightarrow s_{[a_k]} \) such that \( \{s_{[a_i]}, s_{[a_{i'}]}, s_{[a_j]}\} \subseteq L \) and \( r(a_i, a_{i'}, a_j, a_k) \) is a rule application. Since, by inductive hypothesis, \( a_i, a_{i'}, a_j \) are derived from \( S \), so that \( a_k \) is also derived from \( S \) by means of \( r \).

We remark that the literals which can be unit-propagated from \( \phi_T^{all} \land \bigwedge_{ax \in S} s_{[ax]} \) are all and only those which can be unit-propagated from \( \phi_T^{all} \land \bigwedge_{ax \in S} s_{[ax]} \) because, as above stated at point (ii), assertion clauses (9) play no role in unit-propagations.

**Theorem 3.** Given an \( \mathcal{EL}^+ \) TBox \( T \) in normal form, for every \( S \subseteq T \) and for every pair of concept names \( C, D \) in \( PC_T \), \( C \sqsubseteq S \) \( D \) if and only if the Horn propositional formula \( \phi_T^{all} \land \bigwedge_{ax \in S} s_{[ax]} \land \neg s_{[C \sqsubseteq D]} \) is unsatisfiable.

*Proof.* Due to the soundness and completeness of the set of classification rules reported in Section 2.1, \( C \sqsubseteq S \) \( D \) if and only if there exists a sequence of rule applications \( r_1, \ldots, r_N \) generating \( C \sqsubseteq D \) from \( S \). If and only if this is the case, by definition, \( \phi_T^{all} \) contains all the rule clauses of type (10) corresponding to all the rule applications \( r_1, \ldots, r_N \).
Thus, on the one hand, if $C \subseteq D$, then $\bigwedge_{ax_i \in S} s_{[ax_i]}$ and all the clauses of type (10) corresponding to all the rule applications $r_1, \ldots, r_N$, from Lemma 9, force $s_{[C \subseteq D]}$ to be true, which falsifies the unit clause $\neg s_{[C \subseteq D]}$. Thus $\phi_T^{\mathrm{all}}(\text{po}) \wedge \bigwedge_{ax_i \in S} s_{[ax_i]} \wedge \neg s_{[C \subseteq D]}$ is unsatisfiable.

(If.) On the other hand suppose $\phi_T^{\mathrm{all}}(\text{po}) \wedge \bigwedge_{ax_i \in S} s_{[ax_i]} \wedge \neg s_{[C \subseteq D]}$ is unsatisfiable. Let $\phi_T^*$ be the result of assigning in $\phi_T^{\mathrm{all}}(\text{po})$ all $s_{[ax_i]}$ with $ax_i$ in $S$ to $\top$ and unit-propagating the values. (Notice that $\phi_T^*$ is satisfiable since $\phi_T^{\mathrm{all}}(\text{po}) \wedge \bigwedge_{ax_i \in S}$ is satisfiable.) By Lemma 9 and by the definition of $\phi_T^{\mathrm{all}}(\text{po})$ and $\phi_T^*$, all and only the variables $s_{[a_j]}$ s.t. $a_j$ can be derived from $S$ are unit-propagated in this process. Thus, by contradiction, if $C \subseteq D$ could not be derived from $S$, then $s_{[C \subseteq D]}$ would not be unit-propagated in this process, so that $\phi_T^* \wedge \neg s_{[C \subseteq D]}$ would be satisfiable; hence $\phi_T^{\mathrm{all}}(\text{po}) \wedge \bigwedge_{ax_i \in S} s_{[ax_i]} \wedge \neg s_{[C \subseteq D]}$ would be satisfiable, violating the hypothesis.

**Theorem 4.** Given an $\mathcal{EL}^+$ TBox $T$ in normal form, for every $S \subseteq T$ and for every pair of concept names $C, D$ in $\mathcal{PC}_T$, $C \subseteq D$ if and only if the Horn propositional formula $\phi_T^{\mathrm{all}}(\text{po}) \wedge \bigwedge_{ax_i \in S} s_{[ax_i]} \wedge p[C] \wedge \neg p[D]$ is unsatisfiable.

**Proof.**

(Only if.) Suppose $C \subseteq D$. By Theorem 3, since $\phi_T^{\mathrm{all}}(\text{po}) \equiv \phi_T(\text{po})$, we have that $\phi_T^{\mathrm{all}} \wedge \bigwedge_{ax_i \in S} s_{[ax_i]} \models s_{[C \subseteq D]}$. Moreover, if $C \subseteq S$ then $C \subseteq T$; thus, by Definition 4 and due to the soundness and completeness of the classification rules, $\phi_T^{\mathrm{all}}$ contains the assertion clause

$$s_{[C \subseteq D]} \rightarrow (p[C] \rightarrow p[D]).$$

Thus $\phi_T^{\mathrm{all}} \wedge \bigwedge_{ax_i \in S} s_{[ax_i]} \wedge p[C] \wedge \neg p[D]$ is unsatisfiable.

(If.) Suppose $\phi_T^{\mathrm{all}} \wedge \bigwedge_{ax_i \in S} s_{[ax_i]} \wedge p[C] \wedge \neg p[D]$ is unsatisfiable. Since $\phi_T^{\mathrm{all}}$ is a definite Horn formula, $\phi_T^{\mathrm{all}} \wedge \bigwedge_{ax_i \in S}$ is satisfiable. Let $\phi_T^*$ be the result of assigning in $\phi_T^{\mathrm{all}}$ all $s_{[ax_i]}$ with $ax_i$ in $S$ to true and of unit-propagating the values. Hence $\phi_T^* \wedge p[C] \wedge \neg p[D]$ is unsatisfiable and $\phi_T^*$ is satisfiable.

By Lemma 9 and by the definition of $\phi_T^{\mathrm{all}}$ and $\phi_T^*$, all and only the variables $s_{[a_j]}$ such that $a_j$ can be derived from $S$ are unit-propagated in the process of building $\phi_T^*$. Thus $\phi_T^*$ consists only on clauses in the forms:

(i) $\mathcal{EL}^+2\text{SAT}(a_j)$ such that $a_j$ is derived from $S$ (assertion clauses (9) from $\phi_T(\text{so})$),

(ii) $(s_{[a_j]} \rightarrow \mathcal{EL}^+2\text{SAT}(a_j))$ such that $a_j$ is not derived from $S$ (assertion clauses (9) from $\phi_T(\text{so})$), and

(iii) $(\bigwedge_j s_{[a_j]} \rightarrow s_{[a_k]}$ such that $a_j, a_k$ are not derived from $S$ (rule clauses (10) from $\phi_T^{\mathrm{all}}(\text{po})$).


Since $\phi^*_T \land p[C] \land \neg p[D]$ is unsatisfiable, then $\phi^*_T \models p[C] \rightarrow p[D]$. Thus, like in the proof of Theorem 1, there must be a resolution derivation $P$ of $(p[C] \rightarrow p[D])$ from some subset of clauses $\psi_1, \ldots, \psi_n$ in $\phi^*_T$.

We notice that all $\psi_i$’s must be in form (i). In fact, a resolution derivation involving as leaves also clauses in the form (ii) and (iii) could only infer clauses containing at least one literal in the form $\neg s_{[a_i]}$, since there is no way of resolving away such literals against clauses (i)-(iii) without reintroducing new ones. Thus, since $\phi^*_T$ is a definite Horn formula, every resolution step in $P$ can be written in the form

$$
\frac{(\bigland_i p[\{X_i\}]) \rightarrow p[\{X_n\}]}{\bigland_i p[\{X_i\}] \land \bigland_j p[\{X_j\}] \rightarrow p[\{X_n\}]}
$$

s.t. all $X$’s are concepts. Each corresponding derivation step

$$
\frac{(\bigland_i X_i) \subseteq X_k}{(\bigland_i X_i \cap \bigland_j X_j) \subseteq X_n}
$$

is valid in $\mathcal{EL}^+$. Moreover, each $\psi_i$ is $\mathcal{EL}^+2SAT(a_i)$ for some $a_i$ which is either in $S$ or has been derived from $S$. Hence, there is a valid derivation of $C \subseteq D$ from $S$ in $\mathcal{EL}^+$, so that $C \subseteq S D$.

Notice that from the latter fact we can also conclude that the clause $p[C] \rightarrow p[D]$ belongs to $\phi^*_S$ and therefore, by Definition 4, that (13) belongs to $\phi^S_{all}$ (more precisely $\phi^S_{all}(S_{\{a_i\}})$). Thus (13) belongs also to $\phi^T_{all}$. 

The following corollary is obtained straightforwardly from Theorem 4, stating $S \overset{\text{def}}{=} T$.

**Corollary 5.** Given an $\mathcal{EL}^+$ TBox $T$ in normal form, for every pair of concept names $C, D$ in $\mathcal{PC}_T$, $C \subseteq T D$ if and only if the Horn propositional formula $\phi^T_{all} \land \bigland_{a_i \in T} s_{[a_i]} \land p[C] \land \neg p[D]$ [resp. $\phi^T_{all}(po) \land \bigland_{a_{x_i} \in T} s_{[a_{x_i}]} \land \neg s_{[C \subseteq D]}$] is unsatisfiable.

We finally prove some facts concerning the Cone-of-influence Modularization exposed in Section 4.1. First we prove that it preserves pinpointing, including in the computed module all the existing MinAs for the given subsumption query.

**Theorem 6.** Given an $\mathcal{EL}^+$ TBox $T$ in normal form and the formula $\phi^T_{all}(po)$ as defined in Definition 4, for every pair of concept names $C, D$ in $\mathcal{PC}_T$, the following facts hold:

(i) $C \subseteq T D$ if and only if $C \subseteq M_{C \subseteq D}^{\phi_{all}}$;

(ii) if $S$ is a MinA for $C \subseteq T D$, then $S \subseteq M_{C \subseteq D}^{\phi_{all}}$.

25. Notice that, by Lemma 9 and by Definition 3, all and only the clauses of type (i) are exactly $\phi_S$, since they are the propositional encoding of all the subsumption relations deducible by $S$. Intuitively, $\phi_S \land p[C] \land \neg p[D]$ is the unsatisfiable subpart of $\phi^*_T \land p[C] \land \neg p[D]$, because there is no way of resolving away the literals $\neg s_{[a_j]}$ from the other clauses of $\phi^*_T$, which are all of type (ii) and (iii). By Theorem 1, from $\phi_S \land p[C] \land \neg p[D]$ unsatisfiable it follows $C \subseteq S D$.

26. Notice that this statement is purely based on $\mathcal{EL}^+$ semantic, meaning “if an interpretation satisfies the premises, it satisfies also the conclusion”, and does not depend on the set of derivation rules used.
where $\mathcal{M}_{\subseteq D}^{C,i}$ is the cone-of-influence module for $C \subseteq D$ w.r.t. $\phi_{T,(p0)}^{all}$, as defined in Definition 5.

Proof.

(i) By Theorem 3 and by Definition 5 of $\mathcal{M}_{\subseteq D}^{C,i}$ w.r.t. $\phi_{T,(p0)}^{all}$, in place of fact (i), we can equivalently prove that the Horn propositional formula $\psi_{\bar{C}D}^{T,(p0)} \wedge \Lambda_{ax_i \in T} s_{[ax_i]} \wedge \neg s_{[C \subseteq D]}$, namely $\psi_{\bar{C}D}^{T,(p0)}$, is unsatisfiable if and only if the Horn propositional formula $\phi_{T,(p0)}^{all} \wedge \Lambda_{ax_i \in \mathcal{M}_{\subseteq D}^{C,i}} s_{[ax_i]} \wedge \neg s_{[C \subseteq D]}$, namely $\psi_{\bar{C}D}^{C,i}$, is unsatisfiable. On the one hand, since $\psi_{\bar{C}D}^{T,(p0)} = \psi_{\bar{C}D}^{C,i} \wedge \Lambda_{ax_i \in \mathcal{M}_{\subseteq D}^{C,i}} s_{[ax_i]}$, if $\psi_{\bar{C}D}^{C,i}$ is unsatisfiable then $\psi_{\bar{C}D}^{T,(p0)}$ is also trivially unsatisfiable. On the other hand, we consider any minimal subset $\mathcal{S}$ of $T$ such that $\phi_{T,(p0)}^{all} \wedge \Lambda_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge \neg s_{[C \subseteq D]}$ is unsatisfiable and we refer to the following proof of fact (ii).

(ii) Point (ii) is again consequence of Theorem 3. Let $\mathcal{S}$ be any minimal subset of $T$ such that $C \subseteq \mathcal{S}$ and $C \subseteq D$ for every $\mathcal{S}' \subseteq \mathcal{S}$. By Theorem 3 we have that $\phi_{T,(p0)}^{all} \wedge \Lambda_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge \neg s_{[C \subseteq D]}$ is unsatisfiable. Since $\phi_{T,(p0)}^{all}$ is trivially satisfiable, it follows that $\phi_{T,(p0)}^{all} \wedge \Lambda_{ax_i \in \mathcal{S}} s_{[ax_i]} \models s_{[C \subseteq D]}$. But since, by Theorem 3, $\phi_{T,(p0)}^{all} \wedge \Lambda_{ax_i \in \mathcal{S} \setminus \{ax_j\}} s_{[ax_i]} \wedge \neg s_{[C \subseteq D]}$ is instead satisfiable for every $ax_j \in \mathcal{S}$, then every $s_{[ax_j]}$ is included in the subformula of $\phi_{T,(p0)}^{all}$ responsible for the deduction of $s_{[C \subseteq D]}$ (i.e. $s_{[ax_j]}$ is included in $\phi_{C \subseteq D}$), so that $s_{[ax_j]} \in \mathcal{P}_{\bar{C}D}^{T,(p0)}$, for every $ax_j \in \mathcal{S}$. Accordingly, by Definition 5, $ax_j \in \mathcal{M}_{\subseteq D}^{C,i}$ for every $ax_j \in \mathcal{S}$ and we have the thesis $\mathcal{S} \subseteq \mathcal{M}_{\subseteq D}^{C,i}$.

Second, we prove that the COI module can be computed in linear time w.r.t. the coi subformula.

**Proposition 7.** Given the Horn propositional formula $\phi_{T,(p0)}^{all}$, the set of assumptions $\mathcal{P}_{\bar{C}D}$ and the query $s_{[C \subseteq D]}$, the algorithm of Figure 6 executes in linear time w.r.t. the number of clauses of the COI subformula $\phi_{C \subseteq D(i)}$. 

Proof. In the algorithm of Figure 6 we assured that each selector variable is enqueued in $Q$ (and thus subsequently handled at Step 4) at most once, when it is reached for the first time. Thus it is easy to see that the algorithm executes a number of external iterations which is linear w.r.t. the number of selector variables included in $Q$, that is exactly the size of $\mathcal{P}_{\bar{C}D}$. In each iteration, once dequeued the selector variable $s_{[a]}$ at Step 4, the algorithm: (i) gets the clauses $\phi_{a}$ (Step 5), (ii) executes a fixed number of instructions for every literal of every clause $c \in \phi_{a}$ (Steps 6-13). Since (i) is performed in linear time w.r.t. the number of clauses in $\phi_{a}$ by exploiting the two-watched-literals scheme, the complexity of the algorithm is dominated by the operations (ii). Thus, since wlog. we can assume that every clause has at-most four literals, the modularization algorithm of Figure 6 terminates in linear time w.r.t. the number of clauses handled by the algorithm, that is $O(|\phi_{C \subseteq D(i)}|)$ (Definition 5).
Third we prove that, assuming a TBox in normal form, Cone-of-influence Modularization produces smaller modules than Reachability-based Modularization.

**Proposition 8.** Given an $\mathcal{EL}^+$ TBox $T$ in normal form and the formula $\phi_{T,T_{(po)}}^{\text{all}}$ as defined in Definition 4, for every pair of concept names $C, D$ in $\mathcal{PC}_T$ it holds $M_{C;D}^{c,o,i} \subseteq M_{C}^{c,\text{reach}}$, where $M_{C;D}^{c,\text{reach}}$ and $M_{C;D}^{c,o,i}$ are, respectively, the reachability-based module (Definition 2) and cone-of-influence module (Definition 5) for $C \sqsubseteq D$.

**Proof.** We remark that the cone-of-influence module $M_{C;D}^{c,o,i}$ for $C \sqsubseteq T D$ is defined w.r.t. the formula $\phi_{T,T_{(po)}}^{\text{all}}$, thus it is defined w.r.t. a sound and complete set of completion rules for the classification problem. Let us consider the set of completion rules of Table 2 and, for sake of clarity, the generic concept $X$ so that we prove the stronger fact $M_{X \subseteq D}^{c,o,i} \subseteq M_{X}^{\text{reach}}$ for every pair of concept names $X \in \mathcal{PC}_T$ and $D \in \mathcal{NC}_T$.

If a completion rule of Table 2 can be applied starting from a set of assertions (first column) and one axiom (second column), namely $ax$, deducing a new subsumption relation $X \subseteq D$ (third column), then such a completion rule is encoded into $\phi_{T,T_{(po)}}^{\text{all}}$ and, accordingly, the axiom $ax$ becomes part of the cone-of-influence module $M_{X \subseteq D}^{c,o,i}$ for $X \subseteq D$. Wlog. we limit our proof to axiom $ax$ but the same considerations apply to all the other premises of the rule application when they are axioms of $T$. In particular we prove that $ax \in M_{X \subseteq D}^{c,o,i}$ implies $ax \in M_{X}^{\text{reach}}$.

Notice the following two facts:

(i) : given a subsumption relation $a$, the cone-of-influence module $M_{a}^{c,o,i}$ for $a$ is fully included in any cone-of-influence module $M_{a'}^{c,o,i}$, with $a'$ any subsumption relation such that $s_{[a]}$ is in the cone of influence $P_{\mathcal{T}}^{\mathcal{C}}$ for $a'$.

(ii) : the reachability-based module for a given concept $X$ includes all the reachability-based modules for any other concept that is $X$-reachable.

This has been said consider the five completion rules of Table 2:

1. Suppose that the first completion rule is applied on the previously deduced subsumptions $X \subseteq C_1, \ldots, X \subseteq C_k$ (with $k \geq 1$), and on the axiom $ax \overset{\text{def}}{=} C_1 \sqcap \cdots \sqcap C_k \subseteq D$. Thus, $ax \in M_{X \subseteq D}^{c,o,i}$. Since Reachability-based Modularization preserves subsumption (Property 1), then the symbols $C_1, \ldots, C_k$ must be $X$-reachable and, therefore, $ax$ is a $X$-reachable axiom by Definition 2.

2. The same arguments of the previous point can be spent for the second completion rule with the premise $X \subseteq C$, the symbol $C$ and the axiom $ax \overset{\text{def}}{=} C \subseteq \exists r.D$.

3. Suppose that the third completion rule is applied on the previously deduced subsumptions $X \subseteq \exists r.E$ and $E \subseteq C$, and on the axiom $ax = \exists r.C \subseteq D$. Thus, $ax \in M_{X \subseteq D}^{c,o,i}$. Accordingly, $r$ and $E$ must be $X$-reachable and everything that is $E$-reachable is also $X$-reachable. Henceforth $C$ is in turn $X$-reachable. It follows that $ax$ is a $X$-reachable axiom by Definition 2.
4. The same arguments of point 1 can be spent for the forth completion rule with the premise \( X \subseteq \exists r. C \), the symbol \( r \) and the axiom \( ax \overset{\text{def}}{=} r \subseteq s \).

5. Suppose that the fifth completion rule is applied on the previously deduced subsumptions \( X \subseteq \exists r_1.E_1, \ldots, E_{n-1} \subseteq \exists r_n.D \) (with \( n \geq 2 \)), and on the axiom \( ax \overset{\text{def}}{=} r_1 \circ \cdots \circ r_n \). Thus, \( ax \in M_{X \subseteq D}^{\text{e.o.i.}} \). Then \( E_1, r_1 \) must be \( X \)-reachable and everything that is \( E_1/r_1 \)-reachable is also \( X \)-reachable, in particular \( E_2 \) and \( r_2 \) are \( X \)-reachable, and so on and so forth. So, it follows that also the symbols \( E_1, \ldots, E_{n-1}, D \) and \( r_1, \ldots, r_n \) are \( X \)-reachable. Thus, by Definition 2, \( ax \) is a \( X \)-reachable axiom.

Concluding, we proved for every possible completion rule applied in building \( \phi_{\mathcal{T}(po)}^{\text{HI}} \) that if \( ax \in M_{X \subseteq D}^{\text{e.o.i.}} \) then \( ax \) is \( X \)-reachable, i.e. that every \( ax \) included in a cone-of-influence module is included in the respective reachability-based module. Thus, by facts (i) and (ii), we proved \( M_{X \subseteq D}^{\text{e.o.i.}} \subseteq M_{X \subseteq D}^{\text{reach}} \). \( \square \)
Appendix B. Further Experimental Evaluation Data

In Tables 6-15, we compare the detailed results of the four representative variants of $\mathcal{EL}^+$ SAT and those of CEL. For each query and variant/tool we expose the size of the modules extracted respectively through the COI Modularization for $\mathcal{EL}^+$ SAT and the Reachability-based Modularization for CEL. In the central part of the tables we report, for each $\mathcal{EL}^+$ SAT version, the total search time until the last MinA found is returned, for CEL, we report the termination time of the search. In these columns we tagged with a "*" each case where the MinAs’ enumeration was completed within the timeout; due to the problems highlighted in Section 5.1, with CEL this also means that it has completely terminated (i.e., it has not stopped due to the 10-MinAs limit). In all these tables, we highlight in bold the best performer for each entry. In the times shown we avoid considering the encoding time for $\mathcal{EL}^+$ SAT (since it is done offline once forever) and the loading times of the ontology/encoding both for CEL and $\mathcal{EL}^+$ SAT (since they are done once for the whole test session). 27

The execution of $\mathcal{EL}^+$ SAT × 2 deserves a separate mention. Since we must evaluate the overall performance of the approach, we show the total CPU time arising from all the three phases required by $\mathcal{EL}^+$ SAT × 2 (see Section 4.3). For the first instance of $\mathcal{EL}^+$ SAT involved in the approach only modularization time (phase 1.) must be considered, since $\phi_{T}^{\text{all}} / \phi_{T(po)}^{\text{all}}$ is loaded once and then exploited for every modularization query, without reloading. Instead, all the CPU times required by the second instance of $\mathcal{EL}^+$ SAT must be taken into account, because they refer in each query to a different sub-ontology. The second $\mathcal{EL}^+$ SAT instance must: (i) encode the extracted sub-ontology (module), (ii) load the new encoding, (iii) perform the MinAs’ search. (Notice that phase 2. is measured by the time spent in (i), while the time spent in (ii) and (iii) both falls into phase 3.) The times reported in the column (×2) consist in the sum of these four CPU times (modularization time and (i)-(iii) times). In order to demonstrate that phases 1. and 2. are negligible in practice, as predicted in Section 4.3, we report separately in the column titled “enc.” the sum of the time required in those two phases (i.e. modularization and re-encoding times).

In order to have a better view of the results we split the tables vertically in groups, depending on whether the MinAs search has been completed by $\mathcal{EL}^+$ SAT × 2 and/or by CEL. Thus, e.g., the problems which have been completely solved both by $\mathcal{EL}^+$ SAT × 2 and by CEL (with both $\mathcal{EL}^+$ SAT × 2 and CEL times marked with “*”) are exposed in the top-most part of the tables, whilst in the lower-most part we report the problems which (eventually) neither $\mathcal{EL}^+$ SAT nor CEL completely solved. 28 Notice that when CEL either stops due to the limit of 10 MinAs or runs over the 1000 sec. time limit we can not consider the problem as “completely solved” by CEL.

27. These times for $\mathcal{EL}^+$ SAT can be found in the preliminary empirical evaluation of (Sebastiani & Vescovi, 2009b), whilst for CEL they are on average: 25.1, 4.0, 3.4, 1.4 and 0.4 sec. for SNOMED’09, FULLGALEN, NCI, GENEONT. and NOTGALEN respectively.

28. In order to make our empirical evaluation reproducible, we report on the left side of each line the index of the problem within the test suite.
Table 6: Results of the 50 random test queries for SNOMED-CT, with best performance.
<table>
<thead>
<tr>
<th>Module size</th>
<th>MinAs search time (sec.)</th>
<th>#MinAs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ΣL⁺SAT</strong></td>
<td>CEL (b) (c) (t) enc. (x2)</td>
<td>CEL (b) (c) (t) (x2)</td>
</tr>
<tr>
<td>8</td>
<td>143 156 19.2 1.3 0.7 0.0 0.0 10 0.0</td>
<td>20.9* 2 2 2 2 2</td>
</tr>
<tr>
<td>11</td>
<td>119 130 25.8 1.6 0.9 0.0 0.0 11 0.0</td>
<td>21.4* 3 3 3 3 3</td>
</tr>
<tr>
<td>20</td>
<td>156 164 6.7 0.7 0.7 0.0 0.0 0.0</td>
<td>21.0* 1 1 1 1 1</td>
</tr>
<tr>
<td>22</td>
<td>136 178 6.6 0.8 0.6 0.0 0.0 0.0</td>
<td>21.3* 1 1 1 1 1</td>
</tr>
<tr>
<td>29</td>
<td>199 214 6.7 0.7 0.6 0.0 0.0 0.0</td>
<td>21.3* 1 1 1 1 1</td>
</tr>
<tr>
<td>30</td>
<td>190 198 6.6 0.7 0.6 0.0 0.0 0.0</td>
<td>21.4* 1 1 1 1 1</td>
</tr>
<tr>
<td>41</td>
<td>143 151 6.6 0.7 0.7 0.0 0.0</td>
<td>21.2* 1 1 1 1 1</td>
</tr>
<tr>
<td>43</td>
<td>171 177 19.8 1.6 1.0 0.1 0.1 20 0.0</td>
<td>21.9* 2 2 2 2 2</td>
</tr>
<tr>
<td>45</td>
<td>123 134 6.5 0.6 0.5 0.0 0.0 0.0</td>
<td>20.6* 1 1 1 1 1</td>
</tr>
<tr>
<td>47</td>
<td>127 139 6.5 0.6 0.5 0.0 0.0 0.0</td>
<td>20.7* 1 1 1 1 1</td>
</tr>
<tr>
<td>53</td>
<td>136 142 38.9 42.6 9.4 0.0 0.1 24.1* 3 6 6 6 6</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>138 151 6.5 0.6 0.5 0.0 0.0 0.0</td>
<td>20.7* 1 1 1 1 1</td>
</tr>
<tr>
<td>63</td>
<td>134 147 6.5 0.6 0.5 0.0 0.0 0.0</td>
<td>20.7* 1 1 1 1 1</td>
</tr>
<tr>
<td>65</td>
<td>120 126 58.3 129.8 23.0 0.0 0.0</td>
<td>22.6* 3 4 4 4 4</td>
</tr>
<tr>
<td>67</td>
<td>86 94 6.6 0.7 0.6 0.0 0.0 0.0</td>
<td>20.8* 1 1 1 1 1</td>
</tr>
<tr>
<td>69</td>
<td>146 152 19.3 1.1 0.7 0.0 0.0</td>
<td>21.0* 2 2 2 2 2</td>
</tr>
</tbody>
</table>

Table 7: Results of the 50 selected test queries for SNOMED-CT, with best performance.
Table 8: Results of the 50 random test queries for FULL-GALEN, with best performance.
<table>
<thead>
<tr>
<th>Module size</th>
<th>MinAs search time (sec.)</th>
<th>#MinAs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(\mathcal{E}\mathcal{L}^{+}\mathcal{S}\mathcal{AT}) ((\mathcal{E}\mathcal{L}^{+}\mathcal{S}\mathcal{AT})) enc. ((\times 2))</td>
</tr>
<tr>
<td>8</td>
<td>147</td>
<td>14789</td>
</tr>
<tr>
<td>9</td>
<td>147</td>
<td>14789</td>
</tr>
<tr>
<td>14</td>
<td>135</td>
<td>14789</td>
</tr>
<tr>
<td>1</td>
<td>242</td>
<td>14835</td>
</tr>
<tr>
<td>2</td>
<td>234</td>
<td>14838</td>
</tr>
<tr>
<td>3</td>
<td>230</td>
<td>14836</td>
</tr>
<tr>
<td>4</td>
<td>230</td>
<td>14835</td>
</tr>
<tr>
<td>5</td>
<td>157</td>
<td>14789</td>
</tr>
<tr>
<td>6</td>
<td>161</td>
<td>14789</td>
</tr>
<tr>
<td>7</td>
<td>146</td>
<td>14789</td>
</tr>
<tr>
<td>8</td>
<td>184</td>
<td>14789</td>
</tr>
<tr>
<td>9</td>
<td>167</td>
<td>14789</td>
</tr>
<tr>
<td>10</td>
<td>167</td>
<td>14789</td>
</tr>
<tr>
<td>11</td>
<td>199</td>
<td>14794</td>
</tr>
<tr>
<td>12</td>
<td>146</td>
<td>14789</td>
</tr>
<tr>
<td>13</td>
<td>198</td>
<td>14789</td>
</tr>
<tr>
<td>14</td>
<td>151</td>
<td>14789</td>
</tr>
<tr>
<td>15</td>
<td>200</td>
<td>14789</td>
</tr>
<tr>
<td>16</td>
<td>200</td>
<td>14789</td>
</tr>
<tr>
<td>17</td>
<td>161</td>
<td>14789</td>
</tr>
<tr>
<td>18</td>
<td>116</td>
<td>14789</td>
</tr>
<tr>
<td>19</td>
<td>206</td>
<td>14789</td>
</tr>
<tr>
<td>20</td>
<td>161</td>
<td>14789</td>
</tr>
<tr>
<td>21</td>
<td>163</td>
<td>14789</td>
</tr>
<tr>
<td>22</td>
<td>153</td>
<td>14789</td>
</tr>
<tr>
<td>23</td>
<td>197</td>
<td>14789</td>
</tr>
<tr>
<td>24</td>
<td>197</td>
<td>14789</td>
</tr>
<tr>
<td>25</td>
<td>197</td>
<td>14789</td>
</tr>
<tr>
<td>26</td>
<td>193</td>
<td>14789</td>
</tr>
<tr>
<td>27</td>
<td>185</td>
<td>14789</td>
</tr>
<tr>
<td>28</td>
<td>185</td>
<td>14789</td>
</tr>
<tr>
<td>29</td>
<td>193</td>
<td>14789</td>
</tr>
<tr>
<td>30</td>
<td>205</td>
<td>14943</td>
</tr>
<tr>
<td>31</td>
<td>180</td>
<td>14789</td>
</tr>
<tr>
<td>32</td>
<td>178</td>
<td>14789</td>
</tr>
<tr>
<td>33</td>
<td>164</td>
<td>14790</td>
</tr>
<tr>
<td>34</td>
<td>133</td>
<td>14789</td>
</tr>
<tr>
<td>35</td>
<td>173</td>
<td>14789</td>
</tr>
<tr>
<td>36</td>
<td>163</td>
<td>14789</td>
</tr>
<tr>
<td>37</td>
<td>119</td>
<td>14789</td>
</tr>
<tr>
<td>38</td>
<td>114</td>
<td>14789</td>
</tr>
<tr>
<td>39</td>
<td>158</td>
<td>14789</td>
</tr>
<tr>
<td>40</td>
<td>168</td>
<td>14789</td>
</tr>
<tr>
<td>41</td>
<td>142</td>
<td>14789</td>
</tr>
<tr>
<td>42</td>
<td>182</td>
<td>14789</td>
</tr>
<tr>
<td>43</td>
<td>204</td>
<td>14874</td>
</tr>
<tr>
<td>44</td>
<td>204</td>
<td>14874</td>
</tr>
<tr>
<td>50</td>
<td>215</td>
<td>14789</td>
</tr>
</tbody>
</table>

Table 9: Results of the 50 selected test queries for FULL-GALEN, with best performance.

xii
<table>
<thead>
<tr>
<th>Module size</th>
<th>MinAs search time (sec.)</th>
<th>#MinAs</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{L}^+ )SAT</td>
<td>CEL</td>
<td>( \mathcal{L}^+ )SAT</td>
</tr>
<tr>
<td>(B)</td>
<td>(c)</td>
<td>(T)</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>0.2</td>
</tr>
<tr>
<td>3</td>
<td>42</td>
<td>0.2</td>
</tr>
<tr>
<td>3</td>
<td>17</td>
<td>0.2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0.2</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>0.6</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>0.2</td>
</tr>
<tr>
<td>5</td>
<td>19</td>
<td>0.2</td>
</tr>
<tr>
<td>5</td>
<td>91</td>
<td>0.2</td>
</tr>
<tr>
<td>2</td>
<td>29</td>
<td>0.2</td>
</tr>
<tr>
<td>9</td>
<td>42</td>
<td>0.9</td>
</tr>
<tr>
<td>5</td>
<td>39</td>
<td>0.2</td>
</tr>
<tr>
<td>4</td>
<td>158</td>
<td>0.2</td>
</tr>
<tr>
<td>15</td>
<td>270</td>
<td>0.2</td>
</tr>
<tr>
<td>15</td>
<td>66</td>
<td>0.2</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>0.2</td>
</tr>
<tr>
<td>8</td>
<td>39</td>
<td>0.6</td>
</tr>
<tr>
<td>3</td>
<td>26</td>
<td>0.2</td>
</tr>
<tr>
<td>10</td>
<td>49</td>
<td>1.3</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>0.2</td>
</tr>
<tr>
<td>2</td>
<td>35</td>
<td>0.2</td>
</tr>
<tr>
<td>12</td>
<td>24</td>
<td>0.4</td>
</tr>
<tr>
<td>11</td>
<td>67</td>
<td>74.2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.2</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0.2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>0.2</td>
</tr>
<tr>
<td>6</td>
<td>25</td>
<td>0.6</td>
</tr>
<tr>
<td>8</td>
<td>17</td>
<td>0.2</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>0.4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.2</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>0.2</td>
</tr>
<tr>
<td>9</td>
<td>54</td>
<td>1.1</td>
</tr>
<tr>
<td>7</td>
<td>54</td>
<td>0.4</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>0.2</td>
</tr>
<tr>
<td>6</td>
<td>31</td>
<td>0.4</td>
</tr>
<tr>
<td>19</td>
<td>56</td>
<td>12.6</td>
</tr>
<tr>
<td>24</td>
<td>34</td>
<td>7.8</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>66.6</td>
</tr>
<tr>
<td>21</td>
<td>22</td>
<td>78.7</td>
</tr>
<tr>
<td>26</td>
<td>67</td>
<td>624.8</td>
</tr>
<tr>
<td>22</td>
<td>54</td>
<td>54.0</td>
</tr>
<tr>
<td>31</td>
<td>57</td>
<td>14.0</td>
</tr>
<tr>
<td>32</td>
<td>51</td>
<td>229.9</td>
</tr>
<tr>
<td>38</td>
<td>50</td>
<td>67.7</td>
</tr>
<tr>
<td>33</td>
<td>57</td>
<td>335.5</td>
</tr>
</tbody>
</table>

Table 10: Results of the 50 random test queries for NCI, with best performance.
<table>
<thead>
<tr>
<th>Module size</th>
<th>MinAs search time (sec.)</th>
<th>#MinAs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEL</td>
<td>CEL</td>
<td>CEL</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>57</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>51</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>26</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>19</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>36</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>36</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>39</td>
</tr>
<tr>
<td>12</td>
<td>14</td>
<td>43</td>
</tr>
<tr>
<td>13</td>
<td>15</td>
<td>44</td>
</tr>
<tr>
<td>14</td>
<td>16</td>
<td>22</td>
</tr>
<tr>
<td>15</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>12</td>
<td>35</td>
</tr>
<tr>
<td>17</td>
<td>10</td>
<td>32</td>
</tr>
<tr>
<td>18</td>
<td>11</td>
<td>57</td>
</tr>
<tr>
<td>19</td>
<td>12</td>
<td>38</td>
</tr>
<tr>
<td>20</td>
<td>11</td>
<td>54</td>
</tr>
<tr>
<td>21</td>
<td>11</td>
<td>45</td>
</tr>
<tr>
<td>22</td>
<td>14</td>
<td>45</td>
</tr>
<tr>
<td>23</td>
<td>13</td>
<td>58</td>
</tr>
<tr>
<td>24</td>
<td>15</td>
<td>37</td>
</tr>
<tr>
<td>25</td>
<td>13</td>
<td>58</td>
</tr>
<tr>
<td>26</td>
<td>9</td>
<td>60</td>
</tr>
<tr>
<td>27</td>
<td>8</td>
<td>43</td>
</tr>
<tr>
<td>28</td>
<td>16</td>
<td>56</td>
</tr>
<tr>
<td>29</td>
<td>11</td>
<td>40</td>
</tr>
<tr>
<td>30</td>
<td>13</td>
<td>21</td>
</tr>
<tr>
<td>31</td>
<td>10</td>
<td>69</td>
</tr>
<tr>
<td>32</td>
<td>17</td>
<td>66</td>
</tr>
<tr>
<td>33</td>
<td>22</td>
<td>35</td>
</tr>
<tr>
<td>34</td>
<td>23</td>
<td>44</td>
</tr>
<tr>
<td>35</td>
<td>22</td>
<td>24</td>
</tr>
<tr>
<td>36</td>
<td>26</td>
<td>38</td>
</tr>
<tr>
<td>37</td>
<td>20</td>
<td>68</td>
</tr>
<tr>
<td>38</td>
<td>26</td>
<td>65</td>
</tr>
<tr>
<td>39</td>
<td>19</td>
<td>68</td>
</tr>
<tr>
<td>40</td>
<td>38</td>
<td>75</td>
</tr>
<tr>
<td>41</td>
<td>28</td>
<td>63</td>
</tr>
<tr>
<td>42</td>
<td>31</td>
<td>68</td>
</tr>
<tr>
<td>43</td>
<td>38</td>
<td>91</td>
</tr>
<tr>
<td>44</td>
<td>27</td>
<td>66</td>
</tr>
<tr>
<td>45</td>
<td>29</td>
<td>68</td>
</tr>
<tr>
<td>46</td>
<td>43</td>
<td>55</td>
</tr>
<tr>
<td>47</td>
<td>32</td>
<td>91</td>
</tr>
<tr>
<td>48</td>
<td>34</td>
<td>46</td>
</tr>
<tr>
<td>49</td>
<td>31</td>
<td>43</td>
</tr>
<tr>
<td>50</td>
<td>40</td>
<td>52</td>
</tr>
</tbody>
</table>

Table 11: Results of the 50 selected test queries for NCI, with best performance.
<table>
<thead>
<tr>
<th>Module size</th>
<th>$\mathcal{EL}^+$ SAT CEL</th>
<th>$\mathcal{EL}^+$ SAT CEL</th>
<th>$\mathcal{EL}^+$ SAT CEL</th>
<th>$\mathcal{EL}^+$ SAT CEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3 8</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>2</td>
<td>3 18</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>3</td>
<td>2 6</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>4</td>
<td>7 35</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>5</td>
<td>9 37</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>6</td>
<td>5 23</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>7</td>
<td>8 11</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>8</td>
<td>9 3</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>9</td>
<td>10 2</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>10</td>
<td>11 36</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>11</td>
<td>4 7</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>12</td>
<td>6 19</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>13</td>
<td>12 24</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>14</td>
<td>4 9</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>15</td>
<td>2 8</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>16</td>
<td>3 16</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>17</td>
<td>6 28</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>18</td>
<td>9 55</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>19</td>
<td>9 42</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>20</td>
<td>7 15</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>21</td>
<td>3 31</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>22</td>
<td>7 55</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>23</td>
<td>2 7</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>24</td>
<td>4 2</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>25</td>
<td>4 12</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>26</td>
<td>3 10</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>27</td>
<td>9 26</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>28</td>
<td>7 8</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>29</td>
<td>10 16</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>30</td>
<td>5 55</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>31</td>
<td>2 20</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>32</td>
<td>5 8</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>33</td>
<td>3 20</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>34</td>
<td>6 2</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>35</td>
<td>2 4</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>36</td>
<td>2 26</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>37</td>
<td>6 10</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>38</td>
<td>8 35</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>39</td>
<td>3 27</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>40</td>
<td>2 5</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>41</td>
<td>4 30</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>42</td>
<td>3 28</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>43</td>
<td>10 17</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>44</td>
<td>5 6</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>45</td>
<td>8 21</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>46</td>
<td>4 13</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>47</td>
<td>15 18</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>48</td>
<td>13 14</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>49</td>
<td>13 22</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
<tr>
<td>50</td>
<td>22 27</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
</tr>
</tbody>
</table>

Table 12: Results of the 50 random test queries for GeneOntology, with best performance.
<table>
<thead>
<tr>
<th>Module size</th>
<th>MinAs search time (sec.)</th>
<th>#MinAs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ε⁺⁺SAT</strong></td>
<td><strong>CEL</strong></td>
<td><strong>ε⁺⁺SAT</strong></td>
</tr>
<tr>
<td>(u)</td>
<td>(c)</td>
<td>(v)</td>
</tr>
<tr>
<td>16</td>
<td>25</td>
<td>141.0</td>
</tr>
<tr>
<td>15</td>
<td>23</td>
<td>123.2</td>
</tr>
<tr>
<td>14</td>
<td>23</td>
<td>289.6</td>
</tr>
<tr>
<td>13</td>
<td>41</td>
<td>677.5</td>
</tr>
<tr>
<td>12</td>
<td>25</td>
<td>23.1</td>
</tr>
<tr>
<td>11</td>
<td>28</td>
<td>2.3</td>
</tr>
<tr>
<td>10</td>
<td>27</td>
<td>50.7</td>
</tr>
<tr>
<td>9</td>
<td>33</td>
<td>161.9</td>
</tr>
</tbody>
</table>

Table 13: Results of the 50 selected test queries for GeneOntology, with best performance.
Table 14: Results of the 50 random test queries for not-Galen, with best performance.
<table>
<thead>
<tr>
<th>Module size (b)</th>
<th>CEL</th>
<th><code>\(\Sigma\)</code> SAT</th>
<th><code>\(\Sigma\)</code> SAT enc.</th>
<th><code>\(\Sigma\)</code> SAT (x2)</th>
<th>CEL</th>
<th><code>\(\Sigma\)</code> SAT</th>
<th><code>\(\Sigma\)</code> SAT enc.</th>
<th><code>\(\Sigma\)</code> SAT (x2)</th>
<th>CEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>97</td>
<td>0.3 0.0 0.0 0.0 0.0* 0.6*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>96</td>
<td>0.3 0.0 0.0 0.0 0.0* 0.8*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>116</td>
<td>0.3 0.0 0.0 0.0 0.0* 0.7*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>106</td>
<td>0.3 0.0 0.0 0.0 0.0* 0.6*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>105</td>
<td>0.3 0.0 0.0 0.0 0.0* 0.6*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>116</td>
<td>0.3 0.0 0.0 0.0 0.0* 1.0*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>106</td>
<td>0.2 0.0 0.0 0.0 0.0* 1.0*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>105</td>
<td>0.2 0.0 0.0 0.0 0.0* 0.6*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>110</td>
<td>0.2 0.0 0.0 0.0 0.0* 0.6*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>133</td>
<td>0.2 0.0 0.0 0.0 0.0* 1.1*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>112</td>
<td>0.2 0.0 0.0 0.0 0.0* 0.6*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>97</td>
<td>0.2 0.0 0.0 0.0 0.0* 0.6*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>112</td>
<td>0.2 0.0 0.0 0.0 0.0* 0.6*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>115</td>
<td>0.2 0.0 0.0 0.0 0.0* 0.7*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>106</td>
<td>0.3 0.0 0.0 0.0 0.0* 0.8*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>109</td>
<td>0.2 0.0 0.0 0.0 0.0* 0.6*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>113</td>
<td>0.3 0.0 0.0 0.0 0.0* 0.7*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>116</td>
<td>0.2 0.0 0.0 0.0 0.0* 0.6*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>297</td>
<td>0.1 0.0 0.0 0.0 0.0* 1.4*</td>
<td>1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>180</td>
<td>0.1 0.0 0.0 0.0 0.0* 0.6*</td>
<td>1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>183</td>
<td>0.1 0.0 0.0 0.0 0.0* 0.6*</td>
<td>1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>232</td>
<td>0.1 0.0 0.0 0.0 0.0* 1.1*</td>
<td>1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>215</td>
<td>0.1 0.0 0.0 0.0 0.0* 0.8*</td>
<td>1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>277</td>
<td>0.1 0.0 0.0 0.0 0.0* 1.2*</td>
<td>1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>111</td>
<td>0.3 0.0 0.0 0.0 0.0* 0.7*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>111</td>
<td>0.2 0.0 0.0 0.0 0.0* 0.6*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>110</td>
<td>0.1 0.0 0.0 0.0 0.0* 0.4*</td>
<td>1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>239</td>
<td>0.1 0.0 0.0 0.0 0.0* 0.9*</td>
<td>1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>130</td>
<td>0.3 0.0 0.0 0.0 0.0* 0.7*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>130</td>
<td>0.3 0.0 0.0 0.0 0.0* 0.7*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>130</td>
<td>0.2 0.0 0.0 0.0 0.0* 0.7*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>130</td>
<td>0.2 0.0 0.0 0.0 0.0* 1.1*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>130</td>
<td>0.2 0.0 0.0 0.0 0.0* 1.0*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>130</td>
<td>0.2 0.0 0.0 0.0 0.0* 0.7*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>130</td>
<td>0.2 0.0 0.0 0.0 0.0* 0.7*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>130</td>
<td>0.3 0.0 0.0 0.0 0.0* 0.7*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>132</td>
<td>0.3 0.0 0.0 0.0 0.0* 0.6*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>127</td>
<td>0.3 0.0 0.0 0.0 0.0* 0.7*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>113</td>
<td>0.2 0.0 0.0 0.0 0.0* 0.6*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>116</td>
<td>0.2 0.0 0.0 0.0 0.0* 1.0*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>110</td>
<td>0.2 0.0 0.0 0.0 0.0* 0.7*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>119</td>
<td>0.2 0.0 0.0 0.0 0.0* 0.6*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>113</td>
<td>0.2 0.0 0.0 0.0 0.0* 0.6*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>113</td>
<td>0.2 0.0 0.0 0.0 0.0* 0.6*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>113</td>
<td>0.3 0.0 0.0 0.0 0.0* 0.6*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>113</td>
<td>0.3 0.0 0.0 0.0 0.0* 0.7*</td>
<td>2 2 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 15: Results of the 50 selected test queries for NOT-GALEN, with best performance.