Department of
Information Engineering
and Computer Science
DISI

UNIVERSITY
OF TRENTO - Italy

DISI - Via Sommarive, 9 - 38123 POVO, Trento - Italy
http://disi.unitn.it

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.

Journal of Artificial Intelligence Research 1 (2010) 1-80 Submitted ?/??; published ?/??

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 $\mathcal{T}$ into a *polynomial-size* Horn propositional formula $\phi^{all}_{\mathcal{T}}$; then, for each subsumption inference a_i under analysis, a sequence of sets of literals of $\phi^{all}_{\mathcal{T}}$ —corresponding to the complete sequence of minimal sets of axioms in the original ontology $\mathcal{T}$ from which a_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, Toman, & 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-

ity of Galen (Rector & Horrocks, 1997). For example, according to IHTSDO[1], Snomed-CT[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 $\mathcal{T}$, *concept subsumption* is the problem of deciding if a concept C is a subconcept of another concept D in $\mathcal{T}$ (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 $\mathcal{T}$ 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* $\mathcal{T}$ 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 ("**Min**imal **A**xiom 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.

1. *International Health Terminology Standards Development Organization*, `http://www.ihtsdo.org/`.
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 $\mathcal{T}$ into a *polynomial-size* Horn propositional formula $\phi^{all}_{\mathcal{T}}$, representing the deduction steps performed by the classification algorithms proposed by Baader et al. (2006b, 2007). $\phi^{all}_{\mathcal{T}}$ is such that

- atomic propositions label concepts and inferred subsumption relations from $\mathcal{T}$, and
- Horn clauses encode subsumption relations and applications of completion rules.

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

(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 $\phi^{all}_{\mathcal{T}}$, exploiting the *SAT under assumptions* technique by Eén and Sörensson (2004).

(ii) For each subsumption relation a_i derivable from $\mathcal{T}$, (the labelling propositions of) one minimal subset of axioms in $\mathcal{T}$ 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 $\mathcal{T}$ plus other information is implicitly used as background "theory".

Notice that (iii) requires building a *polynomial-size* formula $\phi^{all}_{\mathcal{T}}$, 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 $\phi^{all}_{\mathcal{T}}$ 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 $\mathcal{T}'$ of all potentially-relevant axioms by COI modularization; (b) compute from scratch from $\mathcal{T}'$ the new formula $\phi^{all}_{\mathcal{T}'}$; (c) perform the All-MinA enumeration (iii) on $\phi^{all}_{\mathcal{T}'}$ instead of on $\phi^{all}_{\mathcal{T}}$. Notice that typically $\mathcal{T}'$ and $\phi^{all}_{\mathcal{T}'}$ are order of magnitude smaller than $\mathcal{T}$ and $\phi^{all}_{\mathcal{T}}$ 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 $\mathcal{EL}^+$SAT on the four ontologies above, comparing the various versions of $\mathcal{EL}^+$SAT with one another and with the state-of-the-art tool CEL (Baader et al., 2006a), showing that the renewed version of $\mathcal{EL}^+$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 $\mathcal{DL}$ 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 $\mathsf{PC}_\mathcal{T}$ 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 $\mathsf{PR}_\mathcal{T}$ 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 $\mathsf{signature}()$, 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 $\mathcal{T}$, $\mathsf{signature}(\mathcal{T})$, is the set of concept and role names occurring in $\mathcal{T}$: $\mathsf{signature}(\mathcal{T}) = \mathsf{PC}_\mathcal{T} \cup \mathsf{PR}_\mathcal{T}$.

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 $\mathcal{O}_{\mathsf{med}}$. [4]

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

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

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 GCIs 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 $\mathcal{I}$ is a couple $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is the domain, i.e. a non-empty set of individuals, and $\cdot^{\mathcal{I}}$ is the interpretation function which maps each concept name C to a set $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and maps each role name r to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. In the right-most column of Table 1 at p. 7 the inductive extensions of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions are defined. An interpretation $\mathcal{I}$ is a *model* of a given TBox $\mathcal{T}$ 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 $\mathcal{T}$. A TBox $\mathcal{T}'$ is a *conservative extension* of the TBox $\mathcal{T}$ if every model of $\mathcal{T}'$ is also a model of $\mathcal{T}$, and every model of $\mathcal{T}$ can be extended to a model of $\mathcal{T}'$ 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 $\mathcal{T}$, written $X \sqsubseteq_{\mathcal{T}} Y$ (or simply $X \sqsubseteq Y$ when it is clear to which TBox we refer to), iff $X^{\mathcal{I}} \subseteq Y^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. The computation of all subsumption relations between concept names occurring in $\mathcal{T}$ is called *classification* of $\mathcal{T}$. 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.

	Syntax	Semantics
top	$\top$	$\Delta^{\mathcal{I}}$
conjunction	$X \sqcap Y$	$X^{\mathcal{I}} \cap Y^{\mathcal{I}}$
existential restriction	$\exists r.X$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \wedge y \in X^{\mathcal{I}}\}$
general concept inclusion	$X \sqsubseteq Y$	$X^{\mathcal{I}} \subseteq Y^{\mathcal{I}}$
role inclusion	$r_1 \circ \cdots \circ r_n \sqsubseteq s$	$r_1^{\mathcal{I}} \circ \cdots \circ r_k^{\mathcal{I}} \subseteq s^{\mathcal{I}}$

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):

$$(C_1 \sqcap ... \sqcap C_k) \sqsubseteq D \qquad k \geq 1 \tag{1}$$

$$C \sqsubseteq \exists r.D \tag{2}$$

$$\exists r.C \sqsubseteq D \tag{3}$$

$$r_1 \circ \cdots \circ r_n \sqsubseteq s \qquad n \geq 1 \tag{4}$$

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

- some *top-level* axioms representing the original axioms of $\mathcal{T}$ (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 $\mathcal{T}$, and the size of $\mathcal{T}'$ is linear w.r.t. that of $\mathcal{T}$ (Baader et al., 2005). We call *normal concept* of a normal TBox $\mathcal{T}'$ every non-conjunctive concept description occurring in the concept inclusions of $\mathcal{T}'$; we call $\mathsf{NC}_{\mathcal{T}'}$ the set of all the normal concepts of $\mathcal{T}'$. (I.e., the set $\mathsf{NC}_{\mathcal{T}'}$ consists in all the concepts of the form C or $\exists r.C$, with $C \in \mathsf{PC}_{\mathcal{T}'}$ and $r \in \mathsf{PR}_{\mathcal{T}'}$.)

Example 2.2. The following set of axioms, that we call $\mathcal{O}_{\mathsf{milk}}$, is adapted from a fragment of the ontology not-Galen, and represents some facts and relationships concerning the concept Milk:

$$\begin{aligned}
\mathsf{BodyFluid} &\sqsubseteq \mathsf{Fluid} && m_1\\
\mathsf{Liquid} &\sqsubseteq \mathsf{Fluid} && m_2\\
\mathsf{BodyFluid} &\equiv \mathsf{BodySubstance} \sqcap \exists \mathsf{hasPhysicalState.liquidState} && m_3\\
\mathsf{BodySubstance} &\sqsubseteq \mathsf{Substance} && m_4\\
\mathsf{Milk} &\sqsubseteq \mathsf{BodySubstance} && m_5\\
\mathsf{Milk} &\sqsubseteq \exists \mathsf{hasPhysicalState.liquidState} && m_6\\
\mathsf{Milk} &\sqsubseteq \exists \mathsf{isActedOnSpecificallyBy.}(\mathsf{Secretion} \sqcap \exists \mathsf{isFunctionOf.Breast}) && m_7\\
\mathsf{SecretedSubstance} &\equiv \mathsf{Substance} \sqcap \exists \mathsf{isActedOnBy.Secretion} && m_8\\
\mathsf{Liquid} &\equiv \mathsf{Substance} \sqcap \exists \mathsf{hasPhysicalState.liquidState} && m_9\\
\mathsf{liquidState} &\equiv \mathsf{PhysicalState} \sqcap \exists \mathsf{hasState.liquid} && m_{10}\\
\mathsf{isActedOnSpecificallyBy} &\sqsubseteq_r \mathsf{isActedOnBy} && m_{11}
\end{aligned}$$

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:

$$\begin{aligned}
\mathsf{BodyFluid} &\sqsubseteq \mathsf{BodySubstance} \sqcap \exists \mathsf{hasPhysicalState.liquidState} && m_{3a}\\
\mathsf{BodySubstance} \sqcap \exists \mathsf{hasPhysicalState.liquidState} &\sqsubseteq \mathsf{BodyFluid} && m_{3b}.
\end{aligned}$$

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

$$\begin{aligned}
\mathsf{BodyFluid} &\sqsubseteq \mathsf{BodySubstance}\\
\mathsf{BodyFluid} &\sqsubseteq \mathsf{N}\\
\mathsf{N} &\sqsubseteq \exists \mathsf{hasPhysicalState.liquidState}\\
\mathsf{BodySubstance} \sqcap \mathsf{N} &\sqsubseteq \mathsf{BodyFluid}\\
\exists \mathsf{hasPhysicalState.liquidState} &\sqsubseteq \mathsf{N}.
\end{aligned}$$

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 (Secretion $\sqcap$ $\exists$isFunctionOf.Breast). The definition of M is then split into two axioms:

$$\begin{aligned}
\mathsf{Milk} &\sqsubseteq \exists \mathsf{isActedOnSpecificallyBy.M}\\
\mathsf{M} &\sqsubseteq \mathsf{Secretion}\\
\mathsf{M} &\sqsubseteq \exists \mathsf{isFunctionOf.Breast}.
\end{aligned}$$

$\diamond$

Subsumption assertions ($\ldots \in \mathcal{A}$)	**TBox's axioms** ($\ldots \in \mathcal{T}$)	... added to $\mathcal{A}$
$X \sqsubseteq C_1$, $X \sqsubseteq C_2$, $\ldots$, $X \sqsubseteq C_k$ $\quad k \geq 1$	$C_1 \sqcap \cdots \sqcap C_k \sqsubseteq D$	$X \sqsubseteq D$
$X \sqsubseteq C$	$C \sqsubseteq \exists r.D$	$X \sqsubseteq \exists r.D$
$X \sqsubseteq \exists r.E$, $E \sqsubseteq C$	$\exists r.C \sqsubseteq D$	$X \sqsubseteq D$
$X \sqsubseteq \exists r.D$	$r \sqsubseteq s$	$X \sqsubseteq \exists s.D$
$X \sqsubseteq \exists r_1.E_1$, $\ldots$, $E_{n-1} \sqsubseteq \exists r_n.D$ $\quad n \geq 2$	$r_1 \circ \cdots \circ r_n \sqsubseteq s$	$X \sqsubseteq \exists s.D$

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

2.1.3 Concept Subsumption in $\mathcal{EL}^+$.

Given a normalized TBox $\mathcal{T}$ over the set of primitive concepts $\mathsf{PC}_\mathcal{T}$ and the set of primitive roles $\mathsf{PR}_\mathcal{T}$, the subsumption algorithm for $\mathcal{EL}^+$ proposed by Baader et al. (2007) generates and extends a set $\mathcal{A}$ 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 $\mathcal{T}$. The algorithm starts with the initial set $\mathcal{A} = \{a_i \in \mathcal{T} \mid a_i \text{ is a GCI}\} \cup \{C \sqsubseteq C,\ C \sqsubseteq \top \mid C \in \mathsf{PC}_\mathcal{T}\}$ and extends $\mathcal{A}$ 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 $\mathcal{A}$.) 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 $\mathcal{O}_{\mathsf{med}}$ (Example 2.1 at p. 5) from its original axioms. Once $\mathcal{O}_{\mathsf{med}}$ is turned into normal form its full classification $\mathcal{A}_{\mathcal{O}_{\mathsf{med}}}$ is the following:

$\mathsf{Appendix} \sqsubseteq \mathsf{BodyPart}$	a'_1	$\mathsf{Appendix} \sqsubseteq \exists \mathsf{partOf}.\mathsf{Intestine}$	a''_1
$\mathsf{Endocardium} \sqsubseteq \mathsf{Tissue}$	a'_2	$\mathsf{Endocardium} \sqsubseteq \exists \mathsf{partOf}.\mathsf{HeartValve}$	a''_2
$\mathsf{Pericardium} \sqsubseteq \mathsf{Tissue}$	a'_3	$\mathsf{Pericardium} \sqsubseteq \exists \mathsf{containedIn}.\mathsf{Heart}$	a''_3
$\mathsf{Appendicitis} \sqsubseteq \mathsf{Inflammation}$	a'_4	$\mathsf{Appendicitis} \sqsubseteq \exists \mathsf{hasLoc}.\mathsf{Appendix}$	a''_4
$\mathsf{Endocarditis} \sqsubseteq \mathsf{Inflammation}$	a'_5	$\mathsf{Endocarditis} \sqsubseteq \exists \mathsf{hasLoc}.\mathsf{Endocardium}$	a''_5
$\mathsf{Pericarditis} \sqsubseteq \mathsf{Inflammation}$	a'_6	$\mathsf{Pericarditis} \sqsubseteq \exists \mathsf{hasLoc}.\mathsf{Pericardium}$	a''_6
$\mathsf{Inflammation} \sqsubseteq \mathsf{Disease}$	a'_7	$\mathsf{Inflammation} \sqsubseteq \exists \mathsf{actsOn}.\mathsf{Tissue}$	a''_7
$\mathsf{Disease} \sqcup \mathsf{New} \sqsubseteq \mathsf{HeartDisease}$	a'_8	$\exists \mathsf{hasLoc}.\mathsf{Heart} \sqsubseteq \mathsf{New}$	a_0
$\mathsf{HeartDisease} \sqsubseteq \exists \mathsf{hasState}.\mathsf{NeedsTreat}.$	a_9		
$\mathsf{Appendicitis} \sqsubseteq \mathsf{Disease}$	b_1 $r_1(a'_4, a'_7)$	$\mathsf{Appendicitis} \sqsubseteq \exists \mathsf{actsOn}.\mathsf{Tissue}$	b_2 $r_2(a'_4, a''_7)$
$\mathsf{Endocarditis} \sqsubseteq \mathsf{Disease}$	b_3 $r_1(a'_5, a'_7)$	$\mathsf{Endocarditis} \sqsubseteq \exists \mathsf{actsOn}.\mathsf{Tissue}$	b_4 $r_2(a'_5, a''_7)$
$\mathsf{Pericarditis} \sqsubseteq \mathsf{Disease}$	b_5 $r_1(a'_6, a'_7)$	$\mathsf{Pericarditis} \sqsubseteq \exists \mathsf{actsOn}.\mathsf{Tissue}$	b_6 $r_2(a'_6, a''_7)$
$\mathsf{Pericarditis} \sqsubseteq \exists \mathsf{hasLoc}.\mathsf{Heart}$	b_7 $r_5(a''_6, a''_3, a_{11})$	$\mathsf{Pericarditis} \sqsubseteq \mathsf{New}$	b_8 $r_3(b_7, a_0)$
$\mathsf{Pericarditis} \sqsubseteq \mathsf{HeartDisease}$	b_9 $r_1(b_5, b_8, a'_8)$	$\mathsf{Pericarditis} \sqsubseteq \exists \mathsf{hasState}.\mathsf{NeedsTreat}.$	b_{10} $r_2(b_9, a_9)$

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 $a_{10} \stackrel{\text{def}}{=} \mathsf{partOf} \circ \mathsf{partOf} \sqsubseteq \mathsf{partOf}$ and $a_{11} \stackrel{\text{def}}{=} \mathsf{hasLocation} \circ \mathsf{containedIn} \sqsubseteq \mathsf{hasLocation}$ compose the normalization of $\mathcal{O}_{\mathsf{med}}$. We labeled a'_i (and a''_i) the normal-form top-level

axiom(s) resulting from the normalization of the original axiom a_i (see Example 2.1 at p. 5), while we labeled a_0 the new definition axiom introduced in order to normalize a_8. Next we labeled b_j every other subsumption relation (assertion) inferred through the classification algorithm above exposed, with $j = 1, 2, \ldots 10$ following the inference order of the algorithm. In particular, for each new assertion inferred we show the index of the completion rule of Table 2 at p. 9 applied (r_1 for the first rule, r_2 for the second, and so on and so forth) and the label of its necessary premises in between parenthesis. For instance, the new assertion b_9 is inferred applying a ternary instance of the first completion rule of Table 2 at p. 9, where the premises of the rule are the other assertions b_5, b_8 and the axiom a'_8. Finally, notice that three premises are necessary in order to infer b_8 via the third completion rule r_3, but the second assertion is the trivial inclusion $\mathsf{Heart} \sqsubseteq \mathsf{Heart}$. $\diamond$

Baader et al. (2005) proved the soundness and the completeness of the algorithm together with the fact that the algorithm terminates after polynomially-many rule applications, each of which can be performed in polynomial time. Intuitively, since the number of concept and role names is linear in the size of the input TBox, the algorithm cannot add to $\mathcal{A}$ more than the cardinality of $\mathsf{PC}_\mathcal{T} \times \mathsf{PC}_\mathcal{T} \times \mathsf{PR}_\mathcal{T}$ assertions. Thus, since no rule removes assertions from $\mathcal{A}$, the algorithm stops after at most a polynomial number of rule applications. Moreover, every rule application can be performed in polynomial time.

Once a complete classification of the normalized TBox is computed and stored in some ad-hoc data structure, if $C, D \in \mathsf{PC}_\mathcal{T}$, then $C \sqsubseteq_\mathcal{T} D$ iff the pair C, D can be retrieved from the latter structure. The problem of computing $X \sqsubseteq_\mathcal{T} Y$ s.t. $X, Y \notin \mathsf{PC}_\mathcal{T}$ can be reduced to that of computing $C \sqsubseteq_{\mathcal{T} \cup \{C \sqsubseteq X, Y \sqsubseteq D\}} D$, s.t. C and D are two new concept names.

2.1.4 Axiom Pinpointing in $\mathcal{EL}^+$.

We recall one important definition (Baader et al., 2007).

Definition 1 (nMinA, MinA)**.** Consider the subsumption relation $C_i \sqsubseteq_\mathcal{T} D_i$, with $C_i, D_i \in \mathsf{PC}_\mathcal{T}$. If $C_i \sqsubseteq_\mathcal{S} D_i$ for some set $\mathcal{S} \subseteq \mathcal{T}$ of axioms, then $\mathcal{S}$ is called an *axiom set (nMinA)* for $C_i \sqsubseteq D_i$ w.r.t. $\mathcal{T}$. If $C_i \not\sqsubseteq_{\mathcal{S}'} D_i$ for every $\mathcal{S}'$ s.t. $\mathcal{S}' \subset \mathcal{S}$, then $\mathcal{S}$ is called a *minimal axiom set (MinA)* for $C_i \sqsubseteq D_i$ w.r.t. $\mathcal{T}$.

Example 2.4. In the ontology $\mathcal{O}_{\mathsf{milk}}$ of Example 2.2 at p. 8, a MinA for $\mathsf{Milk} \sqsubseteq \mathsf{SecretedSubstance}$ w.r.t. $\mathcal{O}_{\mathsf{milk}}$ is given by the original axioms $\{m_4, m_5, m_7, m_8, m_{11}\}$.[5] In particular, m_4 and m_5 are necessary to infer $\mathsf{Milk} \sqsubseteq \mathsf{Substance}$, while from (the normalization of) m_7 and m_{11}, it follows that $\mathsf{Milk} \sqsubseteq \exists \mathsf{isActedOnBy}.\mathsf{Secretion}$. Finally $\mathsf{Milk} \sqsubseteq \mathsf{SecretedSubstance}$ can be inferred from the two previous premises and the definition of $\mathsf{SecretedSubstance}$ in m_8. $\diamond$

Baader et al. (2007) proposed a technique for computing all MinAs for $C_i \sqsubseteq_\mathcal{T} D_i$ w.r.t. $\mathcal{T}$: during the classification of $\mathcal{T}$, a *pinpointing formula* (namely $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$) is built, which is a monotone propositional formula [6] on the set of variables $\mathcal{P}_\mathcal{T} \stackrel{\text{def}}{=} \{s_{[ax_j]} \mid ax_j \in \mathcal{T}\}$ s.t., for every $\mathcal{O} \subseteq \mathcal{T}$, $\mathcal{O}$ is a MinA for $C_i \sqsubseteq_\mathcal{T} D_i$ iff $\{s_{[ax_i]} \mid ax_i \in \mathcal{O}\}$ is a minimal valuation of $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$. In a nutshell, the process of building $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$ works as follows. Every axiom $ax_j \in \mathcal{T}$ is encoded with a propositional variable $s_{[ax_j]}$ and every deduced assertion

5. This is the only MinA for this subsumption in $\mathcal{O}_{\mathsf{milk}}$.
6. A *monotone* propositional formula is a propositional formula whose only connectives are $\wedge$ and $\vee$.

$a_j \in \mathcal{A}$ is encoded into a monotone propositional formula. Consider each application of a completion rule r during the classification of $\mathcal{T}$, and let ϕ_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 ϕ_r. Otherwise, if the assertion in the consequence of r is already in $\mathcal{A}$ and it is labeled with ψ, then its label is updated with $\psi^* = \psi \vee \phi_r$, unless $\phi_r \models \psi$.

The Baader et al.'s (2007) all-MinAs algorithm consists thus in (i) building $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$ and (ii) computing all minimal valuations of $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$. This algorithm, however, has serious limitations in terms of complexity: first, the algorithm for generating $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$ requires intermediate logical checks, each of them involving the solution of an NP-complete problem; second, the size of $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$ 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 Suntisrivaraporn (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 & Suntisrivaraporn, 2008) for a detailed description.

Furthermore, Suntisrivaraporn, 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* (Suntisrivaraporn, 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 (Suntisrivaraporn, 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 $a_i \stackrel{\text{def}}{=} C_i \sqsubseteq D_i$ 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.) $\mathcal{M}_{a_i}$ 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 (Σ-reachable symbols, axioms, reachability-based module)**.** Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ and a signature $\Sigma \subseteq \mathsf{signature}(\mathcal{T})$, the set of the Σ-*reachable symbols* of $\mathcal{T}$ is recursively defined as follows:

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

(ii) for every axiom $\hat{C} \sqsubseteq \hat{D}$ of $\mathcal{T}$, if all the symbols in $\mathsf{signature}(\hat{C})$ are Σ-reachable, then all symbols $y \in \mathsf{signature}(\hat{D})$ are Σ-reachable.

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

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

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

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

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 techiniques is fully independent from the completion rules used in the classification of $\mathcal{T}$. In the following

we will refer to this technique either calling it reachability-based modularization or (more generically) syntactic modularization.

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

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 $\mathcal{M}_C^{\mathsf{reach}}$,

(ii) applying the axiom pinpointing algorithm to $\mathcal{M}_C^{\mathsf{reach}}$ 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 modularizatin 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 $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$. We call a *propositional atom* (also called propositional *variable*) every primitive proposition in $\mathcal{P}$, and a *propositional literal* every propositional atom (*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 μ as a conjunction of literals $\bigwedge_i l_i$ (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 $\{\mu_1, ..., \mu_k\}$ is a *complete* set of partial truth assignments satisfying φ iff (i) $\mu_i \models \varphi$ for every i and (ii) every total assignment μ s.t. $\mu \models \varphi$ 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: $\bigwedge_i \bigvee_j l_{ij}$. Each disjunction of literals $\bigvee_j l_{ij}$ is called a *clause*. Notationally, we often write clauses as implications: "$(\bigwedge_i l_i) \rightarrow (\bigvee_j l_j)$" for "$\bigvee_i \neg l_i \vee \bigvee_j l_j$"; also, if η is a conjunction of literals $\bigwedge_i l_i$, we write $\neg\eta$ for the clause $\bigvee_i \neg l_i$, 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 φ) {
1.          μ = ∅; vars = Atoms(φ);
2.          Preprocess(φ, μ);
3.          while (1) {
4.              while (1) {
5.                  status = BCP(φ, μ);
6.                  if (status == unknown) {
7.                      if (TooManyClauses(φ)) {
8.                          DeleteSomeInactiveClauses(φ); }
9.                      break; }
10.                 if (status == sat) {
11.                     return sat; }
12.                 else { // status == conflict
13.                     ψ = FalsifiedClause(φ, μ);
14.                     ψ' = AnalyzeConflict(φ, μ, ψ);
15.                     φ = φ ∧ ψ'; // ψ' learned temporarily
16.                     Backtrack(ψ',μ,φ);
17.                     if (DecisionLevel(μ) == 0) {
18.                         return unsat; }
19.             }   }
20              DecideNextBranch(φ, μ);
21.     }   }
```

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 ϕ is always satisfiable, since it is satisfied by both the assignments $\mu_\top$ and $\mu_\bot$ which assign all variables to true and to false respectively. Notice also that, for every subset $\{p_i\}_i$ of propositional variables in ϕ, $\phi \wedge \bigwedge_i p_i$ and $\phi \wedge \bigwedge_i \neg p_i$ are satisfied by $\mu_\top$ and $\mu_\bot$ respectively. Thus, in order to falsify a definite Horn formula ϕ, 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 φ is in CNF; the assignment μ is initially empty, and it is updated in a stack-based manner. DPLL considers all and only the Boolean variables in $vars \stackrel{\text{def}}{=} Atoms(\varphi)$ (if not specified otherwise).

Initially, `Preprocess` simplifies φ as much as possible, and updates μ accordingly (see e.g., Eén & Biere, 2005; Biere, 2008). In the main loop, after the inner loop, `DecideNextBranch` chooses an unassigned literal l from φ according to some heuristic criterion, and adds it to μ. (This operation is called *decision*, l is called *decision literal* and the number of decision literals in μ 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 φ and μ accordingly; this step is repeated until either (i) μ falsifies some clause in φ, or (ii) no more Boolean variable in $vars$ can be assigned (s.t. μ satisfies φ), 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 ψ, `AnalyzeConflict` detects the subset η of μ which caused the falsification of ψ (*conflict set*), and returns $\psi' \stackrel{\text{def}}{=} \neg\eta$. (This process is called *conflict analysis*, and is described in more details below.) ψ' is then temporarily added to φ (this process is called *learning*). Then `Backtrack` uses ψ' to decide the decision level to backtrack to (namely `blevel`), and then it backtracks to `blevel`, popping out of μ all literals whose decision level is strigtly greater than `blevel`, and updating φ accordingly (this process is called *backjumping*). If `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 φ, then l is added to μ, 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 φ is

a Horn formula, then one run of BCP is sufficient to decide its satisfiability: if $\texttt{BCP}(\varphi, \{\})$ returns conflict, then φ is unsatisfiable; otherwise φ is satisfiable because, since all unit clauses have been removed from φ, all remaining clauses contain at least one negative literal, so that assigning all unassigned literals to false would satisfy φ.

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 nth decision and the literals derived by unit-propagation after it are labeled with n; each non-decision literal l in μ is also tagged by a link to the clause ψ_l causing its unit-propagation (called the *antecedent clause* of l). When a clause ψ is falsified by the current assignment —in which case we say that a *conflict* occurs and ψ is the *conflicting clause*— a *conflict clause* ψ' is computed from ψ s.t. ψ' contains only one literal l_u whose negation has been assigned at the last decision level. ψ' is computed starting from $\psi' = \psi$ by iteratively resolving ψ' with the antecedent clause ψ_l of some literal l in ψ' (typically the last-assigned literal in ψ', see Zhang & Malik, 2002), until some stop criterion is met. E.g., with the *1st-UIP Scheme* the last-assigned literal in ψ' is the one always picked, and the process stops as soon as ψ' contains only one literal l_u assigned at the last decision level; with the *Decision Scheme*, ψ' 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 ψ' 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 ψ' 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 $\eta \stackrel{\text{def}}{=} \neg\psi'$: as soon as all but one literals in η are in μ, then the remaining literal is assigned to false by BCP on ψ'. (To this extent, ψ' 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 φ 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 conflic clause ψ' to φ; 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 $|Atoms(\varphi)|$ 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 learnead 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 necessaily 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 φ *under a set of assumptions* $\mathcal{L} \stackrel{def}{=} \{l_1, ..., l_k\}$. (From a purely-logical viewpoint, this corresponds to check the satisfiability of $\bigwedge_{l_i \in \mathcal{L}} l_i \wedge \varphi$.) This works as follows: $l_1, ..., l_k$ are initially assigned to true, they are tagged as decision literals and added to μ, then the decision level is reset to 0 and `DPLL` enters the external loop. If $\bigwedge_{l_i \in \mathcal{L}} l_i \wedge \varphi$ 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 $\bigvee_{l_j \in \mathcal{L}'} \neg l_j$ s.t. $\mathcal{L}'$ is the (possibly much smaller) subset of $\mathcal{L}$ which actually caused the inconsistency revealed by the SAT solver (i.e., s.t. $\bigwedge_{l_j \in \mathcal{L}'} l_j \wedge \varphi$ 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 $\mathcal{L}$.

This technique is very useful in some situations. First, sometimes one needs checking the satisfiability of a (possibly very big) formula φ under many different sets of assumptions $\mathcal{L}_1, ..., \mathcal{L}_N$. If this is the case, instead of running `DPLL` on $\bigwedge_{l_i \in \mathcal{L}_j} l_i \wedge \varphi$ for every $\mathcal{L}_j$ —which means parsing the formulas and initializing `DPLL` from scratch each time— it is sufficient to parse φ and initialize `DPLL` only once, and run the search under the different sets of assumptions $\mathcal{L}_1, ..., \mathcal{L}_N$. This is particularly important when parsing and initialization times are relevant w.r.t. solving times. In particular, if φ is a Horn formula, solving φ 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 ψ_i of φ can be substituted by the corresponding clauses $s_i \rightarrow \psi_i$, all s_is being fresh propositional variables, which are initially assumed to be true (i.e., $\mathcal{L} = \{s_i \mid \psi_i \in \varphi\}$). If φ is unsatisfiable, then the final conflict clause will be of the form $\bigvee_{s_k \in \mathcal{L}'} \neg s_k$, s.t. $\{\psi_k \mid s_k \in \mathcal{L}'\}$ is the actual subset of clauses which caused the inconsistency of φ. 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 μ satisfying a propositional formula φ which is consistent w.r.t.some propositional theory T.[8] To this extent, most SMT solvers are based on the *lazy SMT* paradighm, where a CDCL SAT solver is used to enumerate propositional truth assignments

8. From a more standard SMT perspective, given an input quantifier-free first-order formula φ_{fo} and some decidable first-order theory T_{fo}, the problem consists in searching for a set of literals μ_{fo} in φ_{fo} whose conjunction tautologically entails φ_{fo} and is consistent in T_{fo}. Thus, the propositional formula φ is the *Boolean abstraction* of the input quantifier-free first-order formula φ_{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 φ_{fo} and are valid in the first-order theory

```
...                                                  ...
if (status == sat) {                                 if (status == sat) {
    η = T-solver(μ);                                     η = T-solver(μ);
    if (η == ∅) // μ consistent w.r.t. T                 if (η != ∅) // μ inconsistent w.r.t. T
        { return sat; }                                      { ψ = ¬η; }
    else { // μ inconsistent w.r.t. T                    else // μ consistent w.r.t. T
        ψ = ¬η;                                              { ψ = ¬μ; }
        ψ' = AnalyzeConflict(φ, μ, ψ);                   ψ' = AnalyzeConflict(φ, μ, ψ);
        φ = φ ∧ ψ'; //ψ' learned temporarily             φ = φ ∧ ψ'; //ψ' learned temporarily
        Backtrack(ψ',μ,φ);                               Backtrack(ψ',μ,φ);
        if (DecisionLevel(μ) == 0)                       if (DecisionLevel(μ) == 0)
            return unsat;                                    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.)

$\mu_1, \mu_2, ...$ satisfying φ. Consider Figure 2 at p. 18 left. Every time a satisfying assignment μ is generated, μ is fed to a *theory solver* ($\mathcal{T}$*-solver*) which checks its consistency w.r.t. T; if μ is consistent w.r.t. T, then the whole procedure returns `sat`; if μ is inconsistent w.r.t. T, then the theory solver returns (the negation η of) one clause ψ valid in T (called *theory lemma*) which is violated by μ, 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 φ is consistent w.r.t. T— or no more truth assignment can be enumerated — so that φ is inconsistent w.r.t. T. Notice that the efficiency of the whole process strongly benefits from the fact that $\mathcal{T}$*-solver* returns *minimal* T-inconsistent subsets η.

This technique is further improved by means of *early pruning* and *theory-propagation*: the theory solver is invoked also on *partial* assignments μ which do not satisfy φ yet; if μ is found T-inconsistent, then a theory lemma is produced and the SAT solver backtracks without further expanding μ; if instead μ is found T-consistent, and if the theory solver is able to perform some deduction in the form $T \wedge \mu' \models l$, s.t. $\mu' \subseteq \mu$ and l is a literal representing a truth assignment to some unassigned atom in φ, then l is unit-propagated by the SAT solver and the theory lemma $\mu \rightarrow l$ 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 $\{\mu_i\}_i$ of (possibly-partial) T-consistent truth assignment satisfying φ. This technique modifies the lazy SMT schema as follows (see Figure 2 at p. 18 right): whenever a (possibly-partial) propositional model μ is generated and μ 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 μ_{fo}.

in T by $\mathcal{T}$-*solver*, then $\psi \stackrel{\text{def}}{=} \neg\mu$ 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 $\mathcal{T}$-*solver*] subsume all $\mathcal{T}$-consistent [resp. $\mathcal{T}$-inconsistent] total truth assignments propositionally satisfying φ.

As with the pure SAT case of Figure 1 at p. 14, it is not necessary to add permanently to φ each conflict clause ψ'; rather, it is sufficient to keep it only as long as it is active, preserving the correctness, completeness and termination of the enumarating algorithm (see, e.g., (Zhang et al., 2001; Nieuwenhuis et al., 2006; Lahiri et al., 2006)), while avoinding the blowup in the number of learned clauses.[9] 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 $\mathcal{T}$ 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 $\phi_{\mathcal{T}}$ representing the classification of the input ontology $\mathcal{T}$.

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

$$\mathcal{EL}^+2sat(a_i) \stackrel{\text{def}}{=} \begin{cases} (p_{[C_1]} \wedge ... \wedge p_{[C_k]}) \rightarrow p_{[D]} & \text{if } a_i \text{ is of type } (C_1 \sqcap ... \sqcap C_k) \sqsubseteq D, k \geq 1; \quad (5) \\ p_{[C]} \rightarrow p_{[\exists r.D]} & \text{if } a_i \text{ is of type } C \sqsubseteq \exists r.D \ ; \quad (6) \\ p_{[\exists r.C]} \rightarrow p_{[D]} & \text{if } a_i \text{ is of type } \exists r.C \sqsubseteq D \ . \quad (7) \end{cases}$$

Then we define the following CNF Horn propositional formula:

$$\phi_{\mathcal{T}} \stackrel{\text{def}}{=} \bigwedge_{a_i \in \mathcal{A}} \mathcal{EL}^+2sat(a_i). \quad (8)$$

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

9. Here we assume that the output models/conflict sets are produced in output one-by-one and then possibly deleted, otherwise we need al least the possibly-exponential space to store all them.

Since the clauses (5)-(7) are definite Horn clauses, $\phi_{\mathcal{T}}$ is a definite Horn formula. Thus, $\phi_{\mathcal{T}}$ 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 $\mathcal{T}$ in normal form, for every pair of concept names C, D in $\mathsf{PC}_{\mathcal{T}}$, $C \sqsubseteq_{\mathcal{T}} D$ if and only if the Horn propositional formula $\phi_{\mathcal{T}} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable.*

In practice, in order to build $\phi_{\mathcal{T}}$, 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 $\mathcal{T}$ or every deduced assertion a_i of the form (1)-(3) which is added to $\mathcal{A}$, we add to $\phi_{\mathcal{T}}$ 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 $|\mathcal{A}| \leq |\mathsf{PC}_{\mathcal{T}}|^2 \cdot |\mathsf{PR}_{\mathcal{T}}|$ (see Section 2.1.3), $\phi_{\mathcal{T}}$ is worst-case polynomial in size w.r.t. $|\mathcal{T}|$ and can be generated in polynomial time. $\diamond$

Once $\phi_{\mathcal{T}}$ 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 $\phi_{\mathcal{T}}$ is parsed and `DPLL` is initialized, each subsumption query $C_i \sqsubseteq_{\mathcal{T}} D_i$ corresponds to solving $\phi_{\mathcal{T}}$ under the assumption list $\mathcal{L}_i \stackrel{\text{def}}{=} \{p_{[C_i]}, \neg p_{[D_i]}\}$. 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 $C_i \sqsubseteq_{\mathcal{T}} D_i$ then $\phi_{\mathcal{T}}$ contains the clause $p_{[C_i]} \rightarrow p_{[D_i]}$, so that `BCP` stops as soon as $p_{[C_i]}$ and $\neg p_{[D_i]}$ are unit-propagated.

Example 3.1. Consider the classification $\mathcal{A}_{\mathcal{O}_{\mathsf{med}}}$ in Example 2.3 at p. 9.[10] Then we have:

$$
\begin{array}{lrcll}
\phi_{\mathcal{O}_{\mathsf{med}}} \stackrel{\text{def}}{=} & p_{[\mathsf{Appendix}]} \rightarrow p_{[\mathsf{BodyPart}]} & a'_1 & \wedge & p_{[\mathsf{Appendix}]} \rightarrow p_{[\exists \mathsf{partOf.Intestine}]} \quad a''_1 \\
\wedge & p_{[\mathsf{Endocardium}]} \rightarrow p_{[\mathsf{Tissue}]} & a'_2 & \wedge & p_{[\mathsf{Endocardium}]} \rightarrow p_{[\exists \mathsf{partOf.HeartValve}]} \quad a''_2 \\
\wedge & p_{[\mathsf{Pericardium}]} \rightarrow p_{[\mathsf{Tissue}]} & a'_3 & \wedge & p_{[\mathsf{Pericardium}]} \rightarrow p_{[\exists \mathsf{containedIn.Heart}]} \quad a''_3 \\
\wedge & p_{[\mathsf{Appendicitis}]} \rightarrow p_{[\mathsf{Inflammation}]} & a'_4 & \wedge & p_{[\mathsf{Appendicitis}]} \rightarrow p_{[\exists \mathsf{hasLocation.Appendix}]} \quad a''_4 \\
\wedge & p_{[\mathsf{Endocarditis}]} \rightarrow p_{[\mathsf{Inflammation}]} & a'_5 & \wedge & p_{[\mathsf{Endocarditis}]} \rightarrow p_{[\exists \mathsf{hasLocation.Endocardium}]} \quad a''_5 \\
\wedge & p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\mathsf{Inflammation}]} & a'_6 & \wedge & p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\exists \mathsf{hasLocation.Pericardium}]} \quad a''_6 \\
\wedge & p_{[\mathsf{Inflammation}]} \rightarrow p_{[\mathsf{Disease}]} & a'_7 & \wedge & p_{[\mathsf{Inflammation}]} \rightarrow p_{[\exists \mathsf{actsOn.Tissue}]} \quad a''_7 \\
\wedge & p_{[\mathsf{Disease}]} \wedge p_{[\mathsf{New}]} \rightarrow p_{[\mathsf{HeartDisease}]} & a'_8 & \wedge & p_{[\exists \mathsf{hasLocation.Heart}]} \rightarrow p_{[\mathsf{New}]} \quad a_0 \\
\wedge & p_{[\mathsf{HeartDisease}]} \rightarrow p_{[\exists \mathsf{hasState.NeedsTreatement}]} & a_9 & & \\
\wedge & p_{[\mathsf{Appendicitis}]} \rightarrow p_{[\mathsf{Disease}]} & b_1 & \wedge & p_{[\mathsf{Appendicitis}]} \rightarrow p_{[\exists \mathsf{actsOn.Tissue}]} \quad b_2 \\
\wedge & p_{[\mathsf{Endocarditis}]} \rightarrow p_{[\mathsf{Disease}]} & b_3 & \wedge & p_{[\mathsf{Endocarditis}]} \rightarrow p_{[\exists \mathsf{actsOn.Tissue}]} \quad b_4 \\
\wedge & p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\mathsf{Disease}]} & b_5 & \wedge & p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\exists \mathsf{actsOn.Tissue}]} \quad b_6 \\
\wedge & p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\exists \mathsf{hasLocation.Heart}]} & b_7 & \wedge & p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\mathsf{New}]} \quad b_8 \\
\wedge & p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\mathsf{HeartDisease}]} & b_9 & \wedge & p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\exists \mathsf{hasState.NeedsTreatement}]} \quad b_{10}
\end{array}
$$

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

10. In this section and in the following ones, with a small abuse of notation, we use the names $\mathcal{O}_{\mathsf{med}}$ and $\mathcal{O}_{\mathsf{milk}}$ to identify both the original ontology and its normalized version.

relations deduced from the axioms of $\mathcal{O}_{\mathsf{med}}$. We use the same labeling of Example 2.3 at p. 9.

For instance, performing concept subsumption on the query $\mathsf{Pericarditis} \sqsubseteq \mathsf{HeartDisease}$ corresponds to solve $\phi_{\mathcal{O}_{\mathsf{med}}}$ under the assumption list $\{p_{[\mathsf{Pericarditis}]}, \neg p_{[\mathsf{HeartDisease}]}\}$, which is clearly unsatisfiable because clause b_9 of $\phi_{\mathcal{O}_{\mathsf{med}}}$ is falsified by the two assumed literals.

Instead, if the query is $\mathsf{Appendicitis} \sqsubseteq \mathsf{HeartDisease}$, this corresponds to solve $\phi_{\mathcal{O}_{\mathsf{med}}}$ under the assumptions $\{p_{[\mathsf{Appendicitis}]}, \neg p_{[\mathsf{HeartDisease}]}\}$. This leads to the unit-propagation in $\phi_{\mathcal{O}_{\mathsf{med}}}$ of $p_{[\mathsf{Inflammation}]}$ and $p_{[\exists \mathsf{hasLocation.Appendix}]}$ from a'_4 and a''_4 respectively, then to the unit-propagation of $p_{[\mathsf{Disease}]}$ and $p_{[\exists \mathsf{actsOn.Tissue}]}$ 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 $\phi_{\mathcal{O}_{\mathsf{med}}}$ is a Horn formula, we can conclude that it is satisfiable, and that $\mathsf{Appendicitis} \sqsubseteq \mathsf{HeartDisease}$ is not a subsumption relation deducible from $\mathcal{O}_{\mathsf{med}}$. $\diamond$

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 $\phi^{all}_{\mathcal{T}}$ representing the complete classification DAG of the input normalized ontology $\mathcal{T}$.[11] The size of $\phi^{all}_{\mathcal{T}}$ is polynomial w.r.t. that of $\mathcal{T}$.

3.2.1 Building the formula $\phi^{all}_{\mathcal{T}}$.

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 $C_1 \sqcap ... \sqcap C_k \sqsubseteq D$ in $\mathcal{T}$ can be rewritten into the set $\{C_1 \sqcap C_2 \sqsubseteq C_{1:2}, C_{1:2} \sqcap C_3 \sqsubseteq C_{1:3}, ..., C_{1:k-1} \sqcap C_k \sqsubseteq D\}$, and each RI axiom of the form $r_1 \circ \cdots \circ r_n \sqsubseteq s$ can be rewritten into the set $\{r_1 \circ r_2 \sqsubseteq r_{1:2}, r_{1:2} \circ r_3 \sqsubseteq r_{1:3}, ..., r_{1:n-1} \circ r_n \sqsubseteq s\}$, each $C_{1:i}$ and $r_{1:j}$ being a fresh concept name and a fresh role name respectively.[12]

Definition 4 ($\phi^{all}_{\mathcal{T}}$, $\phi_{\mathcal{T}(so)}$, $\phi^{all}_{\mathcal{T}(po)}$)**.** 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 $\{p_{[X]} \mid X \in \mathsf{NC}_{\mathcal{T}}\}$. We introduce the set of propositional variables $\mathcal{P}_{\mathcal{A}} \stackrel{\text{def}}{=} \{s_{[a_i]} \mid a_i \in \mathcal{A}\}$, that we call *assertion [resp. axiom] selector variables*. Then we define $\phi^{all}_{\mathcal{T}} \stackrel{\text{def}}{=} \phi_{\mathcal{T}(so)} \wedge \phi^{all}_{\mathcal{T}(po)}$, where

- $\phi_{\mathcal{T}(so)}$ is the conjunction of all the clauses:

$$\{s_{[a_i]} \rightarrow \mathcal{EL}^+2sat(a_i) \mid a_i \in \mathcal{A}\} \tag{9}$$

  that we call *assertion clauses*, and

- $\phi^{all}_{\mathcal{T}(po)}$ is the conjunction of all the clauses:

$$\{(s_{[a_i]} \wedge s_{[a_{i'}]} \wedge s_{[a_j]}) \rightarrow s_{[a_k]} \mid a_i, a_{i'}, a_k \in \mathcal{A},\ a_j \in \mathcal{T} \text{ and } r(a_i, a_{i'}, a_j, a_k)\} \tag{10}$$

  that we call *rule clauses*. With $r(a_i, a_{i'}, a_j, a_k)$ we mean that r is one of the completion rules of the classification algorithm of Section 2.1 and $a_i, a_{i'}, a_j, a_k$ are valid instances

11. Here "complete" means "including also the rule applications generating already-generated assertions".
12. Notice, however, that this is just a simplyfying 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 $\top$.) $\diamond$

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

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

The result in Theorem 1 at p. 20 extends straightforwardly to $\phi^{all}_{\mathcal{T}}$, as described in the following.

Theorem 3. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form, for every $\mathcal{S} \subseteq \mathcal{T}$ and for every pair of concept names C, D in $\mathsf{PC}_{\mathcal{T}}$, $C \sqsubseteq_{\mathcal{S}} D$ if and only if the Horn propositional formula $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge \neg s_{[C \sqsubseteq D]}$ is unsatisfiable.*

Theorem 4. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form, for every $\mathcal{S} \subseteq \mathcal{T}$ and for every pair of concept names C, D in $\mathsf{PC}_{\mathcal{T}}$, $C \sqsubseteq_{\mathcal{S}} D$ if and only if the Horn propositional formula $\phi^{all}_{\mathcal{T}} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable.*

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

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

Notice that, in general, there may be more than one way of deducing $C \sqsubseteq D$ from $\mathcal{S}$. 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 $\phi^{all}_{\mathcal{T}}$ is generated, it is possible to reason in terms of every subset $\mathcal{S}$ of $\mathcal{T}$ by "selecting" all and only the axioms we are interested in. This requires no new formula generation or computation on $\mathcal{S}$ or $\mathcal{T}$. Rather, it is sufficient to restrict the list of the assumptions for each query on $\phi^{all}_{\mathcal{T}}$ to the set of the selector variables of the axioms of $\mathcal{S}$ and to the selector variable of the query. $\diamond$

Example 3.2. Consider the ontology $\mathcal{O}_{\mathsf{med}}$ in Examples 2.1 at p. 5 and 2.3 at p. 9. Then $\phi^{all}_{\mathcal{O}_{\mathsf{med}}} \stackrel{\text{def}}{=} \phi_{\mathcal{O}_{\mathsf{med}}(so)} \wedge \phi^{all}_{\mathcal{O}_{\mathsf{med}}(po)}$:

$$
\begin{aligned}
\phi_{\mathcal{O}_{\mathsf{med}}(so)} \stackrel{\text{def}}{=}\ & s_{[a_1]} \rightarrow (p_{[\mathsf{Appendix}]} \rightarrow p_{[\mathsf{BodyPart}]}) && \wedge\ s_{[a_1]} \rightarrow (p_{[\mathsf{Appendix}]} \rightarrow p_{[\exists \mathsf{partOf.Intestine}]})\\
\wedge\ & s_{[a_2]} \rightarrow (p_{[\mathsf{Endocardium}]} \rightarrow p_{[\mathsf{Tissue}]}) && \wedge\ s_{[a_2]} \rightarrow (p_{[\mathsf{Endocardium}]} \rightarrow p_{[\exists \mathsf{partOf.HeartValve}]})\\
\wedge\ & s_{[a_3]} \rightarrow (p_{[\mathsf{Pericardium}]} \rightarrow p_{[\mathsf{Tissue}]}) && \wedge\ s_{[a_3]} \rightarrow (p_{[\mathsf{Pericardium}]} \rightarrow p_{[\exists \mathsf{containedIn.Heart}]})\\
\wedge\ & s_{[a_4]} \rightarrow (p_{[\mathsf{Appendicitis}]} \rightarrow p_{[\mathsf{Inflammation}]}) && \wedge\ s_{[a_5]} \rightarrow (p_{[\mathsf{Appendicitis}]} \rightarrow p_{[\exists \mathsf{hasLocation.Appendix}]})\\
\wedge\ & s_{[a_5]} \rightarrow (p_{[\mathsf{Endocarditis}]} \rightarrow p_{[\mathsf{Inflammation}]}) && \wedge\ s_{[a_5]} \rightarrow (p_{[\mathsf{Endocarditis}]} \rightarrow p_{[\exists \mathsf{hasLocation.Endocardium}]})\\
\wedge\ & s_{[a_6]} \rightarrow (p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\mathsf{Inflammation}]}) && \wedge\ s_{[a_6]} \rightarrow (p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\exists \mathsf{hasLocation.Pericardium}]})\\
\wedge\ & s_{[a_7]} \rightarrow (p_{[\mathsf{Inflammation}]} \rightarrow p_{[\mathsf{Disease}]}) && \wedge\ s_{[a_7]} \rightarrow (p_{[\mathsf{Inflammation}]} \rightarrow p_{[\exists \mathsf{actsOn.Tissue}]})\\
\wedge\ & s_{[a_8]} \rightarrow (p_{[\mathsf{Disease}]} \wedge p_{[\mathsf{New}]} \rightarrow p_{[\mathsf{HeartDisease}]}) && \wedge\ s_{[a_0]} \rightarrow (p_{[\exists \mathsf{hasLocation.Heart}]} \rightarrow p_{[\mathsf{New}]})\\
\wedge\ & s_{[a_9]} \rightarrow (p_{[\mathsf{HeartDisease}]} \rightarrow p_{[\exists \mathsf{hasState.NeedsTreatement}]}) &&\\
\wedge\ & s_{[b_1]} \rightarrow (p_{[\mathsf{Appendicitis}]} \rightarrow p_{[\mathsf{Disease}]}) && \wedge\ s_{[b_2]} \rightarrow (p_{[\mathsf{Appendicitis}]} \rightarrow p_{[\exists \mathsf{actsOn.Tissue}]})\\
\wedge\ & s_{[b_3]} \rightarrow (p_{[\mathsf{Endocarditis}]} \rightarrow p_{[\mathsf{Disease}]}) && \wedge\ s_{[b_4]} \rightarrow (p_{[\mathsf{Endocarditis}]} \rightarrow p_{[\exists \mathsf{actsOn.Tissue}]})\\
\wedge\ & s_{[b_5]} \rightarrow (p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\mathsf{Disease}]}) && \wedge\ s_{[b_6]} \rightarrow (p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\exists \mathsf{actsOn.Tissue}]})\\
\wedge\ & s_{[b_7]} \rightarrow (p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\exists \mathsf{hasLocation.Heart}]}) && \wedge\ s_{[b_8]} \rightarrow (p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\mathsf{New}]})\\
\wedge\ & s_{[b_9]} \rightarrow (p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\mathsf{HeartDisease}]}) && \wedge\ s_{[b_{10}]} \rightarrow (p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\exists \mathsf{hasState.NeedsTreatement}]})
\end{aligned}
$$

$$
\begin{aligned}
\phi^{all}_{\mathcal{O}_{\mathsf{med}}(po)} \stackrel{\text{def}}{=}\ & s_{[a_4]} \wedge s_{[a_7]} \rightarrow s_{[b_1]} && \wedge\ s_{[a_4]} \wedge s_{[a_7]} \rightarrow s_{[b_2]}\\
\wedge\ & s_{[a_5]} \wedge s_{[a_7]} \rightarrow s_{[b_3]} && \wedge\ s_{[a_5]} \wedge s_{[a_7]} \rightarrow s_{[b_4]}\\
\wedge\ & s_{[a_6]} \wedge s_{[a_7]} \rightarrow s_{[b_5]} && \wedge\ s_{[a_6]} \wedge s_{[a_7]} \rightarrow s_{[b_6]}\\
\wedge\ & s_{[a_6]} \wedge s_{[a_3]} \wedge s_{[a_{11}]} \rightarrow s_{[b_7]} && \wedge\ s_{[b_7]} \wedge s_{[a_0]} \rightarrow s_{[b_8]}\\
\wedge\ & s_{[b_5]} \wedge s_{[b_8]} \wedge s_{[a_8]} \rightarrow s_{[b_9]} && \wedge\ s_{[b_9]} \wedge s_{[a_9]} \rightarrow s_{[b_{10}]}.
\end{aligned}
$$

Notice that $s_{[a_{10}]}$ and $s_{[a_{11}]}$ 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 $\phi_{\mathcal{O}_{\mathsf{med}}(so)}$.

Consider the formula $\phi^{all}_{\mathcal{O}_{\mathsf{med}}} \wedge \bigwedge_{i=0,\ldots,11} s_{[a_i]} \wedge p_{[\mathsf{Pericarditis}]} \wedge \neg p_{[\mathsf{HeartDisease}]}$. The propagation of $p_{[\mathsf{Pericarditis}]}$ and $\neg p_{[\mathsf{HeartDisease}]}$ causes the propagation of $\neg s_{[b_9]}$ from the last but one clause of $\phi_{\mathcal{O}_{\mathsf{med}}(so)}$. The propagation of $s_{[a_0]}$, $s_{[a_3]}$, $s_{[a_6]}$, $s_{[a_7]}$ and $s_{[a_{11}]}$ causes that of $s_{[b_5]}, s_{[b_7]}$ and hence of $s_{[b_8]}$ (from the fifth, seventh and eighth clauses of $\phi^{all}_{\mathcal{O}_{\mathsf{med}}(po)}$). Thus, since also $s_{[a_8]}$ is propagated, the nine-th clause in $\phi^{all}_{\mathcal{O}_{\mathsf{med}}(po)}$ is falsified. Thus, we conclude that $\mathsf{Pericarditis} \sqsubseteq_{\mathcal{O}_{\mathsf{med}}} \mathsf{HeartDisease}$.

It is easy to see, instead, that the formula $\phi^{all}_{\mathcal{O}_{\mathsf{med}}} \wedge \bigwedge_{a_i \in \mathcal{O}_{\mathsf{med}}} s_{[a_i]} \wedge p_{[\mathsf{Appendicitis}]} \wedge \neg p_{[\mathsf{HeartDisease}]}$ is satisfiable, from which we conclude that $\mathsf{Appendicitis} \not\sqsubseteq_{\mathcal{O}_{\mathsf{med}}} \mathsf{HeartDisease}$. $\diamond$

Example 3.3. We report some sample clauses from the formula $\phi^{all}_{\mathcal{O}_{\mathsf{milk}}(po)}$ for the ontology $\mathcal{O}_{\mathsf{milk}}$ 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

they represent.)

$$
\begin{array}{rcll}
\phi^{all}_{\mathcal{O}_{\mathsf{milk}}(po)} \stackrel{\text{def}}{=} & \ldots & \wedge \ \ldots & m_0 = \exists \mathsf{hasPhysState.liquidState} \sqsubseteq \mathsf{N} \\
s_{[m_5]} \wedge s_{[m_4]} \rightarrow s_{[n_1]} & & \wedge \ \ldots & m_1 = \mathsf{BodyFluid} \sqsubseteq \mathsf{Fluid} \qquad n_1 = \mathsf{Milk} \sqsubseteq \mathsf{Substance} \\
s_{[m_6]} \wedge s_{[m_0]} \rightarrow s_{[n_2]} & & \wedge \ \ldots & m_2 = \mathsf{Liquid} \sqsubseteq \mathsf{Fluid} \qquad n_2 = \mathsf{Milk} \sqsubseteq \mathsf{N} \\
s_{[m_5]} \wedge s_{[n_2]} \wedge s_{[m_3]} \rightarrow s_{[n_3]} & & \wedge \ \ldots & m_3 = \mathsf{BodySubstance} \sqcap \mathsf{N} \sqsubseteq \mathsf{BodyFluid} \qquad n_3 = \mathsf{Milk} \sqsubseteq \mathsf{BodyFluid} \\
s_{[n_1]} \wedge s_{[n_2]} \wedge s_{[m_9]} \rightarrow s_{[n_4]} & & \wedge \ \ldots & m_4 = \mathsf{BodySubstance} \sqsubseteq \mathsf{Substance} \qquad n_4 = \mathsf{Milk} \sqsubseteq \mathsf{Liquid} \\
s_{[n_3]} \wedge s_{[m_1]} \rightarrow s_{[n_5]} & & \wedge \ \ldots & m_5 = \mathsf{Milk} \sqsubseteq \mathsf{BodySubstance} \qquad n_5 = \mathsf{Milk} \sqsubseteq \mathsf{Fluid} \\
s_{[n_4]} \wedge s_{[m_2]} \rightarrow s_{[n_5]} & & \wedge \ \ldots & m_6 = \mathsf{Milk} \sqsubseteq \exists \mathsf{hasPhysState.liquidState} \\
& \ldots & & m_9 = \mathsf{Substance} \sqcap \mathsf{N} \sqsubseteq \mathsf{Liquid}
\end{array}
$$

We notice that, assuming all the $s_{[m_i]}$'s, there are two distinct chains of unit-propagations leading to propagate $s_{[n_5]}$: one from $\{s_{[m_0]}, s_{[m_1]}, s_{[m_3]}, s_{[m_5]}, s_{[m_6]}\}$, propagating $s_{[n_2]}$, $s_{[n_3]}$ and $s_{[n_5]}$, and another from $\{s_{[m_0]}, s_{[m_2]}, s_{[m_4]}, s_{[m_5]}, s_{[m_6]}, s_{[m_9]}\}$, propagating $s_{[n_1]}$, $s_{[n_2]}$, $s_{[n_4]}$ and $s_{[n_5]}$, corresponding respectively to the deduction of n_2, n_3 and n_5 from the axioms $\{m_0, m_1, m_3, m_5, m_6\}$ and to that of n_1, n_2, n_4 and n_5 from $\{m_0, m_2, m_4, m_5, m_6, m_9\}$. Thus we can conclude that $\{m_0, m_1, m_3, m_5, m_6\}$ and $\{m_0, m_2, m_4, m_5, m_6, m_9\}$ are two nMinAs for $n_5 \stackrel{\text{def}}{=} \mathsf{Milk} \sqsubseteq_{\mathcal{O}_{\mathsf{milk}}} \mathsf{Fluid}$. Since they are also minimal, they are also MinAs for n_5. $\diamond$

In practice, in order to build the formula $\phi^{all}_{\mathcal{T}}$, 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 $\phi_{\mathcal{T}(so)}$ and $\phi^{all}_{\mathcal{T}(po)}$ to the empty set of clauses. Then for every non-trivial GCI axiom $a_i \in \mathcal{T}$, add to $\phi_{\mathcal{T}(so)}$ the corresponding assertion clause of type (9);

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

3. for every possible rule instantiation $r(a_i, a_{i'}, a_j, a_k)$ of a completion rule r of Table 2 at p. 9 (either extending $\mathcal{A}$ or not), add to $\phi^{all}_{\mathcal{T}(po)}$ 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 $\mathcal{A}$.) 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 Q and all axioms are included in $\mathcal{A}$. At each iteration one assertion a_h is dequeued, and steps 2. and/or 3. are applied *to all and only* the rule applications whose antecedents are exactly a_h and one or two of the previously-dequeued axioms/assertions $a_1, ..., a_{h-1}$. The novel assertions a_k which are deduced by the rule applications in step 2 are enqueued into Q and added to $\mathcal{A}$. This process ends when Q is empty.

The algorithm exposed above requires a polynomial amount of steps w.r.t. the size of $\mathcal{T}$. 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 $\phi^{all}_{\mathcal{T}}$ that, in Proposition 2 at p. 22, has been proved to be at most polynomial in the size of $\mathcal{T}$.

```
ClauseSet build-φ^all_T (NormalizedOntology T)
// Initialization of Q, A, φ_T(so), φ^all_T(po)
 1.    Q = ∅; A = ∅; φ_T(so) = ∅; φ^all_T(po) = ∅;
 2.    for each primitive concept C in T {
 3.       add C ⊑ C and C ⊑ ⊤ to A; introduce s_[C⊑C] = s_[C⊑⊤] = ⊤;
 4.       enqueue {C ⊑ C, C ⊑ ⊤} into Q; }
 5.    for each GCI or RI axiom ax in T {
 6.       add ax to A; introduce s_[ax];
 7.       if (ax is a non-trivial GCI axiom) then {
 8.          add the clause (s_[ax] → EL+2sat(ax)) to φ_T(so);
 9.          enqueue ax into Q; }}
// Updating A, B and Q (B is the set of already-handled assertions)
10.    B = ∅;
11.    while (Q is not empty) {
12.       dequeue a_h from Q;
13.       for each rule instance r(a_i, a_i', a_j, a_k) such that {a_i, a_i', a_j} \ B = {a_h} {
14.          if a_k ∉ A then {
15.             add a_k to A; introduce s_[a_k];
16.             add the clause (s_[a_k] → EL+2sat(a_k)) to φ_T(so);
17.             enqueue a_k into Q; }
18.          add the clause ((s_[a_i] ∧ s_[a_i'] ∧ s_[a_j]) → s_[a_k]) to φ^all_T(po); }
19.       B = B ∪ {a_h}; }
20.    return φ^all_T ≝ φ_T(so) ∧ φ^all_T(po);
```

Figure 3: Polynomial-time algorithm building the formula $\phi^{all}_{\mathcal{T}}$. Q is a queue of assertions, $\mathcal{A}$ and $\mathcal{B}$ are sets of assertions.

3.2.2 Computing single MinAs.

We consider some sub-ontology $\mathcal{S} \subseteq \mathcal{T}$. Once $\phi^{all}_{\mathcal{T}}$ 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 $\mathcal{P}_{\mathcal{S}} \stackrel{\text{def}}{=} \{s_{[ax_j]} \mid ax_j \in \mathcal{S}\}$, after $\phi^{all}_{\mathcal{T}}$ is parsed and DPLL is initialized, deciding if $C_i \sqsubseteq_{\mathcal{S}} D_i$ holds is equivalent to solving $\phi^{all}_{\mathcal{T}}$ under the assumption list $\mathcal{L}_i \stackrel{\text{def}}{=} \mathcal{P}_{\mathcal{S}} \cup \{p_{[C_i]}, \neg p_{[D_i]}\}$, 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 $\psi^{C_i,D_i}_{\mathcal{S}^*} \stackrel{\text{def}}{=} p_{[D_i]} \vee \neg p_{[C_i]} \vee \bigvee_{a_i \in \mathcal{S}^*} \neg s_{[a_i]}$ s.t. $\mathcal{S}^*$ is an nMinA for $C_i \sqsubseteq_{\mathcal{S}} D_i$. In fact, since $\phi^{all}_{\mathcal{T}}$ is a definite Horn formula, the presence of both $p_{[C_i]}$ and $\neg p_{[D_i]}$ in $\mathcal{L}_i$ 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 $C_i \sqsubseteq D_i$ and having $\mathcal{S}^*$ as

```
variable-list ComputeOneMinA(formula φ^all_T(po), variable-list P_S, literal ¬s_[Ci⊑Di])
1.    if (DpllUnderAssumptions(φ^all_T(po), P_S ∪ {¬s_[Ci⊑Di]}) == unsat) { // if Ci ⊑_S Di
2.       P_S* = {s_[ai] | ai ∈ S*};
         // s.t.  ψ^{Ci,Di}_S* ≝ s_[Ci⊑Di] ∨ ⋁_{ai∈S*} ¬s_[ai]  is the conflict clause returned
3.       for each axiom selector variable s_[axj] in P_S* {
4.          L = P_S* \ {s_[axj]};
5.          if (DpllUnderAssumptions(φ^all_T(po), L) == unsat) // if Ci ⊑_{S*\{axj}} Di
6.             P_S* = L; }
7.       return P_S* \ {¬s_[Ci⊑Di]}; }
8.    else
9.       return ∅;
```

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 $C_i \sqsubseteq D_i$.)

Analogously, by Theorem 3 at p. 22, deciding if $C_i \sqsubseteq_{\mathcal{S}} D_i$ holds corresponds also to solving $\phi^{all}_{\mathcal{T}(po)}$ under the assumption list $\mathcal{L}_i \stackrel{\text{def}}{=} \mathcal{P}_{\mathcal{S}} \cup \{\neg s_{[C_i \sqsubseteq D_i]}\}$, so that the algorithm for pinpointing is changed only in the fact that $\phi^{all}_{\mathcal{T}(po)}$ and $\{\neg s_{[C_i \sqsubseteq D_i]}\}$ are used instead of $\phi^{all}_{\mathcal{T}}$ and $\{p_{[C_i]}, \neg p_{[D_i]}\}$ respectively, returning the clause $\psi^{C_i,D_i}_{\mathcal{S}^*} \stackrel{\text{def}}{=} s_{[C_i \sqsubseteq D_i]} \vee \bigvee_{a_i \in \mathcal{S}^*} \neg s_{[a_i]}$ s.t. $\mathcal{S}^*$ is an nMinA for $C_i \sqsubseteq_{\mathcal{S}} D_i$. Thus, wlog. in the remaining part of this section we will reason using $\phi^{all}_{\mathcal{T}(po)}$ and $\{\neg s_{[C_i \sqsubseteq D_i]}\}$. (The same results, however, can be obtained using $\phi^{all}_{\mathcal{T}}$ and $\{p_{[C_i]}, \neg p_{[D_i]}\}$ 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 $\phi^{all}_{\mathcal{T}(po)}$, the set of the axiom variables $\mathcal{P}_{\mathcal{S}}$ and the query selector variable $s_{[C_i \sqsubseteq D_i]}$, `ComputeOneMinA` computes the set of the axiom selector variables representing one MinA $\mathcal{S}^*$ if $C_i \sqsubseteq_{\mathcal{S}} D_i$, or returns the empty set otherwise. (As before, we assume that $\phi^{all}_{\mathcal{T}(po)}$ has been parsed and `DPLL` has been initialized.) First, as described above, $\phi^{all}_{\mathcal{T}(po)}$ is solved under the assumption list $\mathcal{L}_i \stackrel{\text{def}}{=} \mathcal{P}_{\mathcal{S}} \cup \{\neg s_{[C_i \sqsubseteq D_i]}\}$. If satisfiable, the whole procedure returns the empty set. Otherwise, `DpllUnderAssumptions` return `unsat`, producing the conflict clause $\psi^{C_i,D_i}_{\mathcal{S}^*}$ and hence the set $\mathcal{P}_{\mathcal{S}^*} = \{s_{[a_i]} \mid a_i \in \mathcal{S}^*\}$ s.t. $\mathcal{S}^*$ is an nMinA. Then the algorithm tries to remove one-by-one (the selection variables of) the axioms ax_js in $\mathcal{S}^*$, each time checking whether the reduced set of axioms $\mathcal{S}^* \setminus \{ax_j\}$ is still such that $C_i \sqsubseteq_{\mathcal{S}^* \setminus \{ax_j\}} D_i$. To this extent, the minimization algorithm in rows `3-6` is identical to that in (Baader et al., 2007), except that the check $C_i \sqsubseteq_{\mathcal{S}^* \setminus \{ax_j\}} D_i$ is performed by invoking `DpllUnderAssumptions` over $\phi^{all}_{\mathcal{T}(po)}$ and $\mathcal{P}_{\mathcal{S}^*} \setminus \{s_{[ax_j]}\}$, 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 $\mathcal{S}'$ s.t. $\mathcal{S}' \subset \mathcal{S}^* \setminus \{ax_j\}$, then in row `6` $\mathcal{P}_{\mathcal{S}^*}$ is assigned to $\mathcal{P}_{\mathcal{S}'}$ rather than to $\mathcal{L}$, so that all axioms in $(\mathcal{S}^* \setminus \{ax_j\}) \setminus \mathcal{S}'$ 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 $\mathcal{S} \subseteq \mathcal{T}$. We describe a way of generating *all* MinAs for $C_i \sqsubseteq_{\mathcal{S}} D_i$ from $\phi^{all}_{\mathcal{T}(po)}$ and $\{\neg s_{[C_i \sqsubseteq D_i]}\}$. (As before, the same results can be obtained if $\phi^{all}_{\mathcal{T}}$ 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 $\mathcal{P}_{\mathcal{S}} \stackrel{\text{def}}{=} \{s_{[ax_j]} \mid ax_j \in \mathcal{S}\}$ which cause the inconsistency of the formula $\neg s_{[C_i \sqsubseteq D_i]} \wedge \phi^{all}_{\mathcal{T}(po)}$. 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 φ on the set of axiom selector variables $\mathcal{P}_{\mathcal{S}}$. φ is initially set to $\top$ (i.e., the empty set of clauses).[13] One top-level instance of `DPLL` (namely "`DPLL1`") is used to enumerate a complete set of truth assignments $\{\mu_1, ..., \mu_k\}$ on the axiom selector variables in $\mathcal{P}_{\mathcal{S}}$ which satisfy φ. (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 μ is generated, we extract the subset μ^+ of the positive literals in μ, and we pass μ^+ and $\{\neg s_{[C_i \sqsubseteq D_i]}\}$ to an ad-hoc "theory solver" checking if μ^+ falsifies the formula $\neg s_{[C_i \sqsubseteq D_i]} \wedge \phi^{all}_{\mathcal{T}(po)}$. As theory solver we use the concatenation of the functions `PositiveLiteralsOf`, which selects the positive literals in μ, and `ComputeOneMinA` of Figure 4 at p. 26. (To this extent, here $\neg s_{[C_i \sqsubseteq D_i]} \wedge \phi^{all}_{\mathcal{T}(po)}$ plays the role of the propositional theory T.) Notice that we consider only the positive literals in μ because, by construction, all atoms in $\mathcal{P}_{\mathcal{S}}$ occur only negatively in $\phi^{all}_{\mathcal{T}(po)}$, so that the negative literals cannot have any role in falsifying $\neg s_{[C_i \sqsubseteq D_i]} \wedge \phi^{all}_{\mathcal{T}(po)}$.

If $\mu^+ \wedge \neg s_{[C_i \sqsubseteq D_i]} \wedge \phi^{all}_{\mathcal{T}(po)} \models \bot$, then μ^+ represents an nMinA for $C_i \sqsubseteq_{\mathcal{S}} D_i$, and `ComputeOneMinA` returns a minimal subset η of μ^+ which caused such inconsistency. By construction, $\mathcal{S}^* \stackrel{\text{def}}{=} \{ax_j \in \mathcal{S} \mid s_{[ax_j]} \in \eta\}$ is a MinA. The clause $\psi \stackrel{\text{def}}{=} \neg\eta = \bigvee_{ax_j \in \mathcal{S}^*} \neg s_{[ax_j]}$ is then permanently added to φ as a blocking clause and it is used as a conflicting clause for driving next backjumping and learning steps. Otherwise, `ComputeOneMinA` returns and $\psi \stackrel{\text{def}}{=} \neg\mu^+$ 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 $\{\eta_1, ..., \eta_k\}$ 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 `ComputeAllMinAs` is a slight variant of the All-SMT procedure of Section 2.2.4, $\neg s_{[C_i \sqsubseteq D_i]} \wedge \phi^{all}_{\mathcal{T}(po)}$ being the propositional theory T and the concatenation `PositiveLiteralsOf`/`ComputeOneMinA` being the theory solver, with only one slight difference: $\neg\eta$ is also learned permanently; although this step is not strictly necessary (see

13. Notice that DPLL does not require that all Boolean variables in *vars* actually occur in φ. If the idea of enumerating truth assignments for the $\top$ formula puzzles the reader, however, he/she may pretend φ is initially a valid propositional formula on all the atoms in $\mathcal{P}_{\mathcal{S}}$, e.g., $\varphi \stackrel{\text{def}}{=} \bigwedge_{A_i \in \mathcal{P}_{\mathcal{S}}} (A_i \vee \neg A_i)$.

```
MinA-Set ComputeAllMinAs (Variable-list P_S, literal {¬s_[C_i⊑D_i]})
// the formula φ^all_T(po) is global, and it is already initialized
1.      φ = ⊤; μ = ∅; vars = P_S; MinAs = ∅;
2.          while (1) {
3.              while (1) {
4.                  status = BCP(φ, μ);
5.                  if (status == unknown) {
6.                      if (TooManyClauses(φ)) {
7.                          DeleteSomeInactiveClauses(φ); }
8.                      break; }
9.                  if (status == sat) {
10.                     μ+ = PositiveLiteralsOf(μ);
11.                     η = ComputeOneMinA(φ^all_T(po), μ+, ¬s_[C_i⊑D_i]);
12.                     if (η != ∅) {    // μ+ contains a MinA
13.                         MinAs = MinAs ∪ η;
14.                         ψ = ¬η;
15.                         φ = φ ∧ ψ; } // ψ learned permanently
16.                     else {           // μ+ contains no MinA
17.                         ψ = ¬μ; }
18.                     ψ' = AnalyzeConflict(φ, μ, ψ);
19.                     φ = φ ∧ ψ'; // ψ' learned temporarily
20.                     Backtrack(ψ',μ,φ);
21.                     if (DecisionLevel(μ) == 0) {
22.                         return MinAs;
23.                 }   }
24.                 else { // status == conflict
25.                     ψ = FalsifiedClause(φ, μ);
26.                     ψ' = AnalyzeConflict(φ, μ, ψ);
27.                     φ = φ ∧ ψ'; ψ' learned temporarily
28.                     Backtrack(ψ',μ,φ);
29.                     if (DecisionLevel(μ) == 0) {
30.                         return MinAs;
31.                 }   }
32.             }
33.             DecideNextBranch(φ, μ);
34.     }   }
```

Figure 5: All-SMT-based algorithm generating "all MinAs" w.r.t. the given query $C_i \sqsubseteq_{\mathcal{T}} D_i$.

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 ψ' —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 $\mu \stackrel{\text{def}}{=} \mu^+ \cup \mu^-$ does not cause the generation of a MinA, then obviously no assignment in the form $\mu' \stackrel{\text{def}}{=} (\mu^+ \setminus \{s_{i1}, ..., s_{ik}\}) \cup (\mu^- \cup \{\neg s_{i1}, ..., \neg s_{ik}\}$ s.t. $\{s_{i1}, ..., s_{ik}\} \subseteq \mu^+$ can generate a MinA as well. Thus, since all the enumerated assigments μ are total, no assigment which is a superset of $\mu^- \stackrel{\text{def}}{=} \mu \setminus \mu^+$ 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.        else {   // μ+ contains no MinA
17.             ψ = ¬(μ \ μ+); }
```

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

A few more facts are important for the overall efficiency. First, `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 $\mu_1 \stackrel{\text{def}}{=} \{s_{[ax_i]} \mid ax_i \in \mathcal{P}_\mathcal{S}\}$, and it always leads to the discovery of the first MinA. Second, the expensive steps of parsing $\phi^{all}_{\mathcal{T}(po)}$ and initializing the two DPLL instances inside `ComputeAllMinAs` and `ComputeOneMinA` (namely "`DPLL2`") respectively are performed only once, before the whole process starts, and both `ComputeAllMinAs` and `ComputeOneMinA` are always invoked incrementally. Thus it is possible to perform different calls to `ComputeAllMinAs` with different axiom-variable lists and queries without reinitializing the procedure from scratch.

Example 3.4. We want to find all the MinAs for the subsumption $\mathsf{Milk} \sqsubseteq_{\mathcal{O}_\mathsf{milk}} \mathsf{Fluid}$ in the ontology $\mathcal{O}_\mathsf{milk}$ (Examples 2.2 at p. 8, 3.3 at p. 23). We run the procedure in Figure 5 at p. 28 on the formula $\phi^{all}_{\mathcal{O}_\mathsf{milk}(po)}$ and w.r.t. the query $s_{[\mathsf{Milk}\sqsubseteq\mathsf{Fluid}]}$.[14] The top-level `DPLL1` enumerates all the truth assignments on the variables $\mathcal{P}_{\mathcal{O}_\mathsf{milk}} \stackrel{\text{def}}{=} \{s_{[m_i]} \mid m_i \in \mathcal{O}_\mathsf{milk}\}$, satisfying the formula φ initially set to $\top$. The first truth assignment produced is $\mu_1^+ = \mu_1 = \mathcal{P}_{\mathcal{O}_\mathsf{milk}} = \{s_{[m_i]} \mid m_i \in \mathcal{O}_\mathsf{milk}\}$: running `ComputeOneMinA` on $\phi^{all}_{\mathcal{O}_\mathsf{milk}(po)}$ and the input assumptions μ_1 and $\{\neg s_{[\mathsf{Milk}\sqsubseteq\mathsf{Fluid}]}\}$ leads to the identification of the first MinA, e.g., $\mathcal{O}^*_{\mathsf{milk}1} \stackrel{\text{def}}{=} \{m_0, m_2, m_4, m_5, m_6, m_9\}$.[15] Thus the blocking clause $\psi_1 \stackrel{\text{def}}{=} (\neg s_{[m_0]} \vee \neg s_{[m_2]} \vee \neg s_{[m_4]} \vee \neg s_{[m_5]} \vee \neg s_{[m_6]} \vee \neg s_{[m_9]})$ is added to φ, which becomes $\varphi = \psi_1$. Since ψ_1 contains (the negation of) no unit-propagated literal, the procedure backjumps on $\psi_1' = \psi_1$, generating another truth assignment μ_2, which we suppose wlog. to be $\mathcal{P}_{\mathcal{O}_\mathsf{milk}} \cup \{\neg s_{[m_5]}\} \setminus \{s_{[m_5]}\}$, so that

14. Here we only simulate the execution of the procedure without showing the encoding $\phi^{all}_{\mathcal{O}_\mathsf{milk}(po)}$, 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).

$\mu_2^+ \stackrel{\text{def}}{=} \mathcal{P}_{\mathcal{O}_{\mathsf{milk}}} \setminus \{s_{[m_5]}\}$. Since $\phi^{all}_{\mathcal{O}_{\mathsf{milk}}(po)} \wedge \mu_2^+ \wedge \{\neg s_{[\mathsf{Milk} \sqsubseteq \mathsf{Fluid}]}\}$ is satisfiable, `ComputeOneMinA` returns the empty set, and $\psi_2' = \psi_2 = \neg(\mu_2 \setminus \mu_2^+) = s_{[m_5]}$ is used by `DPLL1` as a fake conflicting clause, driving the backjumping.[16] Then `DPLL1` generates a third assignment μ_3, which we suppose $\mu_3 = \mathcal{P}_{\mathcal{O}_{\mathsf{milk}}} \setminus \{s_{[m_4]}\} \cup \{\neg s_{[m_4]}\}$ so that $\mu_3^+ = \mathcal{P}_{\mathcal{O}_{\mathsf{milk}}} \setminus \{s_{[m_4]}\}$. In this case `ComputeOneMinA` identifies the new MinA $\mathcal{O}^*_{\mathsf{milk}_2} \stackrel{\text{def}}{=} \{m_0, m_1, m_3, m_5, m_6\}$, producing $\psi_3 \stackrel{\text{def}}{=} (\neg s_{[m_0]} \vee \neg s_{[m_1]} \vee \neg s_{[m_3]} \vee \neg s_{[m_5]} \vee \neg s_{[m_6]})$, which is permanently added to φ, and $\psi_3' = \psi_3$ drives the next backjumping step. Since now $\varphi = \psi_1 \wedge \psi_3$ and no other MinA for $\mathsf{Milk} \sqsubseteq_{\mathcal{O}_{\mathsf{milk}}} \mathsf{Fluid}$ other than $\mathcal{O}^*_{\mathsf{milk}_1}$ and $\mathcal{O}^*_{\mathsf{milk}_2}$ exists, the procedure will proceed until termination enumerating only truth assignments which do not falsify $\neg s_{[\mathsf{Milk} \sqsubseteq \mathsf{Fluid}]} \wedge \phi^{all}_{\mathcal{O}_{\mathsf{milk}}(po)}$. $\diamond$

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 $\neg s_{[ax_i]}$ on learned clauses from those performed on the clauses in $\phi^{all}_{\mathcal{T}(po)}$: 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 $\mathcal{T} \stackrel{\text{def}}{=} \{ax_1, ..., ax_N\}$ can result from normalizing the non-normal TBox $\hat{\mathcal{T}} \stackrel{\text{def}}{=} \{\hat{ax}_1, ..., \hat{ax}_{\hat{N}}\}$ by means of the process hinted in Section 2.1; $|\mathcal{T}|$ is $O(|\hat{\mathcal{T}}|)$. Each original axiom $\hat{ax}_i$ is converted into a set of normalized axioms $\{ax_{i1}, ..., ax_{in_i}\}$, and each normalized axiom ax_j can be reused (in the case of a definition axiom) in the conversion of several original axioms $\hat{ax}_{j1}, ..., \hat{ax}_{jm_j}$. In order to handle non-normal TBoxes $\hat{\mathcal{T}}$, one variant of the technique of Baader et al. (2007) can be adopted: for every $\hat{ax}_i$, we add to $\phi^{all}_{\mathcal{T}(po)}$ [resp. $\phi^{all}_{\mathcal{T}}$] the set of clauses $\{s_{[\hat{ax}_i]} \rightarrow s_{[ax_{i1}]}, ..., s_{[\hat{ax}_i]} \rightarrow s_{[ax_{in_i}]}\}$, and then we use $\mathcal{P}_{\hat{\mathcal{T}}} \stackrel{\text{def}}{=} \{s_{[\hat{ax}_1]}, ..., s_{[\hat{ax}_{\hat{N}}]}\}$ 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 $\mathcal{P}_{\hat{\mathcal{T}}}$ rather than in $\mathcal{P}_{\mathcal{T}}$. (In practice, we treat normalization as the application of a novel kind of completion rules.) Since $\mathcal{P}_{\hat{\mathcal{T}}}$ is typically smaller than $\mathcal{P}_{\mathcal{T}}$, 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 $\hat{ax}_{j1}, ..., \hat{ax}_{jm_j}$, the clause set $\{s_{[\hat{ax}_{j1}]} \rightarrow s_{[a_j]}, ..., s_{[\hat{ax}_{jm_j}]} \rightarrow s_{[a_j]}\}$ is equivalent to $(s_{[\hat{ax}_{j1}]} \vee ... \vee s_{[\hat{ax}_{jm_j}]}) \rightarrow s_{[a_j]}$.)

Alternatively, a more compact solution we adopted allows for using directly and only the selector variables referring to the original axioms $\hat{\mathcal{T}} = \{\hat{ax}_1, ..., \hat{ax}_{\hat{N}}\}$. 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 $m_5 \stackrel{\text{def}}{=} \mathsf{Milk} \sqsubseteq \mathsf{BodySubstance}$ is a key axiom included in any MinA for $\mathsf{Milk} \sqsubseteq_{\mathcal{O}_{\mathsf{milk}}} \mathsf{Fluid}$. To be a $\mathsf{BodySubstance}$ is necessary both in order to be a $\mathsf{BodyFluid}$ (m_3) and in order to be (a $\mathsf{Substance}$ first and then) a Liquid (m_4 and m_9).

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 $\exists r.A \sqcap \exists s.B \sqsubseteq C \sqcap D$ is normalized into the set of top-level normal axioms: $\{X \sqsubseteq C, X \sqsubseteq D\}$ and the set of definition axioms: $\{\exists r.A \sqsubseteq Y, \exists s.B \sqsubseteq Z, Y \sqcap Z \sqsubseteq X\}$, which define Y as $\exists r.A$, Z as $\exists s.B$ and X as $Y \sqcap Z$.

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

(i) use the same original axiom selector variable $s_{[a\hat{x}_i]}$ for all the top-level normal axioms coming out from the normalization of $a\hat{x}_i$; [17]

(ii) associate the same unique selector variable $s_{[a_0]}$ 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 $s_{[a_0]}$, instead, is used to represent and enable all the axioms defining the new concept names coming out from the normalization. Thus, the presence of $s_{[a_0]}$ 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 $\mathcal{T}$ 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 $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$ of Baader et al. (2007) (see also Section 2.1) with $\phi^{all}_{\mathcal{T}(po)}$, and by analyzing the way they are built and used, we highlight the following differences:

(i) $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$ is built only on axiom selector variables in $\mathcal{P}_\mathcal{T} \stackrel{\text{def}}{=} \{s_{[ax_j]} \mid ax_j \in \mathcal{T}\}$, whilst $\phi^{all}_{\mathcal{T}(po)}$ is build on *all* selector variables in $\mathcal{P}_\mathcal{A} \stackrel{\text{def}}{=} \{s_{[a_j]} \mid a_j \in \mathcal{A}\}$ (i.e., of both axioms and inferred assertions);

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

(iii) the algorithm for generating $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$ of (Baader et al., 2007) requires intermediate logical checks, whilst the algorithm for building $\phi^{all}_{\mathcal{T}(po)}$ 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 $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$, whilst it is (the projection to $\mathcal{P}_\mathcal{T}$ of) a *counter-model* of $\phi^{all}_{\mathcal{T}(po)}$.

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 $\phi^{all}_{\mathcal{T}(po)}$ once and then, for each sub-ontology $\mathcal{S} \subseteq \mathcal{T}$ of interest and for each $C_i \sqsubseteq_\mathcal{S} D_i$ 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 $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$ before starting the enumeration of the MinAs.

We stress the fact that, once $\phi^{all}_{\mathcal{T}(po)}$ 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. $C_1 \sqsubseteq_\mathcal{S} D_1, \ldots, C_i \sqsubseteq_\mathcal{S} D_i, \ldots$, we do not need recomputing $\phi^{all}_{\mathcal{T}(po)}$ each time, we just need assuming (i.e. querying) each time a different axiom selector variable, e.g. respectively: $\neg s_{[C_1 \sqsubseteq_\mathcal{S} D_1]}, \ldots, \neg s_{[C_i \sqsubseteq_\mathcal{S} D_i]}, \ldots$. Same discourse holds for $\phi^{all}_{\mathcal{T}}$ 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 $\mathcal{P}_\mathcal{T}$, although the same techniques can be always applied to $\phi^{all}_{\mathcal{T}(po)}$ (or $\phi^{all}_{\mathcal{T}}$) considering some sub-ontology $\mathcal{S}/\mathcal{P}_\mathcal{S}$ instead of $\mathcal{T}/\mathcal{P}_\mathcal{T}$.

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 $\mathcal{P}^{C_i\sqsubseteq D_i}_\mathcal{T}$ of the selector variables in $\phi^{all}_{\mathcal{T}(po)}$ 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 $\mathcal{T}$, $\phi^{all}_{\mathcal{T}(po)}$ be the encoding of $\mathcal{T}$ in Definition 4 and $\mathcal{P}_\mathcal{T} \stackrel{\text{def}}{=} \{s_{[ax_i]} | ax_i \in \mathcal{T}\}$ the set of the axiom selector variables for $\mathcal{T}$. Then the set of selector variables $\mathcal{P}^{C_i\sqsubseteq D_i}_\mathcal{A}$ and the set of rule clauses $\phi^{C_i\sqsubseteq D_i}_\mathcal{T}$ are defined inductively as follows:

base: $s_{[C_i\sqsubseteq D_i]} \in \mathcal{P}^{C_i\sqsubseteq D_i}_\mathcal{A}$;

induction:

- if $s_{[a_i]} \in \mathcal{P}^{C_i\sqsubseteq D_i}_\mathcal{A}$, then every clause $(\bigwedge_j s_{[a_j]}) \rightarrow s_{[a_i]}$ in $\phi^{all}_{\mathcal{T}(po)}$ is also in $\phi^{C_i\sqsubseteq D_i}_\mathcal{T}$;
- if $(\bigwedge_{j=1}^n s_{[a_j]}) \rightarrow s_{[a_i]}$ is in $\phi^{C_i\sqsubseteq D_i}_\mathcal{T}$, then $s_{[a_j]} \in \mathcal{P}^{C_i\sqsubseteq D_i}_\mathcal{A}$ for every $j \in 1, ..., n$.

$\mathcal{P}^{C_i\sqsubseteq D_i}_\mathcal{A}$ and $\phi^{C_i\sqsubseteq D_i}_\mathcal{T}$ 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. $\phi^{all}_{\mathcal{T}(po)}$, namely $\mathcal{P}^{C_i\sqsubseteq D_i}_\mathcal{T}$, the set of axiom selector variables $\mathcal{P}^{C_i\sqsubseteq D_i}_\mathcal{T} \stackrel{\text{def}}{=} \mathcal{P}^{C_i\sqsubseteq D_i}_\mathcal{A} \cap \mathcal{P}_\mathcal{T}$. Finally, the *COI Module* for $C_i \sqsubseteq_\mathcal{T} D_i$ w.r.t. $\phi^{all}_{\mathcal{T}(po)}$, namely $\mathcal{M}^{\text{c.o.i.}}_{C_i\sqsubseteq D_i}$, is the set $\mathcal{M}^{\text{c.o.i.}}_{C_i\sqsubseteq D_i} \subseteq \mathcal{T}$ of axioms such that $\mathcal{M}^{\text{c.o.i.}}_{C_i\sqsubseteq D_i} \stackrel{\text{def}}{=} \{ax_i \mid s_{[ax_i]} \in \mathcal{P}^{C_i\sqsubseteq D_i}_\mathcal{T}\}$. ◇

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

$$
\begin{array}{lrcl}
\phi^{all}_{\mathcal{O}_{\mathsf{med}}(po)} \stackrel{\text{def}}{=} & s_{[a_4]} \wedge s_{[a_7]} \rightarrow s_{[b_1]} & \wedge & s_{[a_4]} \wedge s_{[a_7]} \rightarrow s_{[b_2]} \quad \text{(a)} \\
\wedge & s_{[a_5]} \wedge s_{[a_7]} \rightarrow s_{[b_3]} & \wedge & s_{[a_5]} \wedge s_{[a_7]} \rightarrow s_{[b_4]} \quad \text{(b)} \\
\wedge & \underline{\underline{s_{[a_6]}}} \wedge \underline{\underline{s_{[a_7]}}} \rightarrow \underline{\underline{s_{[b_5]}}} & \wedge & s_{[a_6]} \wedge s_{[a_7]} \rightarrow s_{[b_6]} \quad \text{(c)} \\
\wedge & \underline{\underline{s_{[a_6]}}} \wedge \underline{\underline{s_{[a_3]}}} \wedge \underline{\underline{s_{[a_{11}]}}} \rightarrow \underline{\underline{s_{[b_7]}}} & \wedge & \underline{s_{[b_7]}} \wedge \underline{\underline{s_{[a_0]}}} \rightarrow \underline{\underline{s_{[b_8]}}} \quad \text{(d)} \\
\wedge & \underline{s_{[b_5]}} \wedge \underline{s_{[b_8]}} \wedge \underline{\underline{s_{[a_8]}}} \rightarrow \underline{\underline{s_{[b_9]}}} & \wedge & s_{[b_9]} \wedge s_{[a_9]} \rightarrow s_{[b_{10}]} \quad \text{(e)}
\end{array}
$$

The query $\mathsf{Pericarditis} \sqsubseteq_{\mathcal{O}_{\mathsf{med}}} \mathsf{HeartDisease}$ 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 $\mathcal{P}_{\mathcal{O}_{\mathsf{med}}}^{\mathsf{Pericarditis} \sqsubseteq \mathsf{HeartDisease}}$, the set of clauses containing underlined variables represent the COI formula $\phi_{\mathcal{O}_{\mathsf{med}}}^{\mathsf{Pericarditis} \sqsubseteq \mathsf{HeartDisease}}$, and the set of the axioms represented by the selector variables marked (iii) represent the module $\mathcal{M}^{\mathsf{c.o.i.}}_{\mathsf{Pericarditis} \sqsubseteq \mathsf{HeartDisease}} = \{a_0, a_3, a_6, a_7, a_8, a_{11}\}$. The module is inductively defined backward from the positive literal $s_{[b_9]}$. Due to the first clause in row `(e)` also the literals $s_{[b_8]}, s_{[b_5]}$ are involved in the cone of influence and a_8 is added to the initially-empty module; due to $s_{[b_8]}$ (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 $s_{[b_5]}$ (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 $\mathsf{Pericarditis} \sqsubseteq \mathsf{HeartDisease}$, unlike in the reachability-based module for $\mathsf{Pericarditis}$ (see Example 2.5 for the computation of $\mathcal{M}^{\mathsf{reach}}_{\mathsf{Pericarditis}}$). ◇

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 $\mathcal{T}$ in normal form and the formula $\phi^{all}_{\mathcal{T}(po)}$ as defined in Definition 4, for every pair of concept names C, D in $\mathsf{PC}_{\mathcal{T}}$, the following facts hold:*

(i) $C \sqsubseteq_{\mathcal{T}} D$ if and only if $C \sqsubseteq_{\mathcal{M}^{\mathsf{c.o.i.}}_{C \sqsubseteq D}} D$;

(ii) if $\mathcal{S}$ is a MinA for $C \sqsubseteq_{\mathcal{T}} D$, then $\mathcal{S} \subseteq \mathcal{M}^{\mathsf{c.o.i.}}_{C \sqsubseteq D}$,

where $\mathcal{M}^{\mathsf{c.o.i.}}_{C_i \sqsubseteq D_i} \subseteq \mathcal{T}$ is the COI module for $C \sqsubseteq D$ w.r.t. $\phi^{all}_{\mathcal{T}(po)}$, 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 $\mathcal{P}_{\mathcal{A}}^{C_i \sqsubseteq D_i}$ and the COI formula $\phi_{\mathcal{T}}^{C_i \sqsubseteq D_i}$ starting from the positive selector variable $s_{[C_i \sqsubseteq D_i]}$, 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 $\phi^{all}_{\mathcal{T}(po)}$ 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 ϕ_{a_i} of all the clauses in which the literal $s_{[a_i]}$ appears positively in linear time w.r.t. the cardinality of ϕ_{a_i} itself.

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

```
variable-list ComputeCOIModule (formula φ^all_T(po), variable-list P_T, literal s_[Ci⊑Di])
 1.    Q = ∅; P_T^{Ci⊑Di} = ∅ ;
 2.    enqueue s_[Ci⊑Di] in Q; mark s_[Ci⊑Di] as reached;
 3.    while Q is not empty {
 4.       dequeue s_[ai] from Q;
 5.       let φ_ai be the set of all the clauses of the form s_[ai] ∨ ⋁_j ¬s_[aj];
 6.       for each clause c ∈ φ_ai {
 7.          for each ¬s_[aj] occurring in c {
 8.             if (s_[aj] is not reached)
 9.                { enqueue s_[aj] in Q; mark s_[aj] as reached; }
10.             if (s_[aj] ∈ P_T)
11.                P_T^{Ci⊑Di} = P_T^{Ci⊑Di} ∪ {s_[aj]};
12.          }
13.       }
14.    }
15.    return P_T^{Ci⊑Di}
```

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 $\mathcal{T}$, this technique improves dramatically the performance of our approach (see Section 5).

Remark 3. Importantly, this technique works directly on the propositional input formula $\phi^{all}_{\mathcal{T}(po)}$, with no need for re-computing anything from $\mathcal{T}$, 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' assertions $\mathcal{P}_{\mathcal{S}}^{C_i \sqsubseteq D_i}$ for any desired sub-ontology $\mathcal{S}$ of $\mathcal{T}$: once $\mathcal{P}_{\mathcal{A}}^{C_i \sqsubseteq D_i}$ has been computed from $\phi^{all}_{\mathcal{T}(po)}$, we have simply $\mathcal{P}_{\mathcal{S}}^{C_i \sqsubseteq D_i} \stackrel{\text{def}}{=} \mathcal{P}_{\mathcal{A}}^{C_i \sqsubseteq D_i} \cap \mathcal{P}_{\mathcal{S}}$. ◇

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 $\mathcal{T}$ 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 $\phi^{all}_{\mathcal{T}(po)}$ (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 $\phi^{all}_{\mathcal{T}(po)}$ includes (and hides) the semantics of $\mathcal{T}$ 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 $\phi^{all}_{\mathcal{T}(po)}$, 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 $\mathcal{T}$ in normal form and the formula $\phi^{all}_{\mathcal{T}(po)}$ as defined in Definition 4, for every pair of concept names C, D in $\mathsf{PC}_\mathcal{T}$ we have that $\mathcal{M}^{\mathsf{c.o.i.}}_{C\sqsubseteq D} \subseteq \mathcal{M}^{\mathsf{reach}}_{C}$, where $\mathcal{M}^{\mathsf{reach}}_{C}$ and $\mathcal{M}^{\mathsf{c.o.i.}}_{C\sqsubseteq D}$ are, respectively, the reachability-based module (Definition 2) and COI module (Definition 5) for $C \sqsubseteq D$.*

In fact, whilst Reachability-based Modularization computes a module "dependent" only from C_i, COI Modularization computes a module "dependent" from both C_i and D_i (i.e. dependent from the subsumption relation $C_i \sqsubseteq D_i$). For example, the reachability-based module computed for the query $C_i \sqsubseteq D_i$ is the same computed for the query $C_i \sqsubseteq E_i$, because it in both cases it includes everything is syntactically reachable from C_i; more precisely, it also includes in the module everything else is syntactically reachable either from D_i or from E_i. (This fact makes us conjecture that some kind of combination of a forward reachability from C_i and a backward one from D_i 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 $\mathcal{M}^{\mathsf{reach}}_{\mathsf{Lithium}}$ and the COI module $\mathcal{M}^{\mathsf{c.o.i.}}_{\mathsf{L.}\sqsubseteq\mathsf{C.S.}}$ for the existing subsumption relation $\mathsf{Lithium} \sqsubseteq_{\text{NOT-GALEN}} \mathsf{ChemicalSubstance}$.

$$
\begin{array}{rcl}
\mathsf{Lithium} & \sqsubseteq & \mathsf{ElementalChemical} \\
\mathsf{Lithium} & \sqsubseteq & \mathsf{Metal} \\
\mathsf{ElementalChemical} & \sqsubseteq & \mathsf{ChemicalSubstance} \\
\mathsf{Metal} & \sqsubseteq & \mathsf{ComplexChemicals} \\
\mathsf{ComplexChemicals} & \sqsubseteq & \mathsf{ChemicalSubstance} \\
\hline
\mathsf{ChemicalSubstance} & \sqsubseteq & \mathsf{Substance} \\
\mathsf{Substance} & \sqsubseteq & \mathsf{GeneralisedSubstance} \\
\mathsf{GeneralisedSubstance} & \sqsubseteq & \mathsf{DomainCategory} \\
\mathsf{DomainCategory} & \sqsubseteq & \mathsf{TopCategory} \\
\mathsf{Substance} & \sqsubseteq & \exists \mathsf{hasCountability}.\mathsf{infinitelyDivisible} \\
\mathsf{hasCountability} & \sqsubseteq_r & \mathsf{StructuralAppearanceModifier} \\
\mathsf{StructuralAppearanceModifier} & \sqsubseteq_r & \mathsf{StructuralModifierAttribute} \\
\mathsf{StructuralModifierAttribute} & \sqsubseteq_r & \mathsf{hasFeature}
\end{array}
\qquad
\begin{array}{rcl}
\mathsf{hasFeature} & \sqsubseteq_r & \mathsf{FeatureStateAttribute} \\
\mathsf{FeatureStateAttribute} & \sqsubseteq_r & \mathsf{ModifierAttribute} \\
\mathsf{ModifierAttribute} & \sqsubseteq_r & \mathsf{DomainAttribute} \\
\mathsf{DomainAttribute} & \sqsubseteq_r & \mathsf{Attribute} \\
\mathsf{infinitelyDivisible} & \sqsubseteq & \mathsf{mass} \\
\mathsf{mass} & \sqsubseteq & \mathsf{CountabilityStatus} \\
\mathsf{CountabilityStatus} & \sqsubseteq & \mathsf{AbstractStatus} \\
\mathsf{AbstractStatus} & \sqsubseteq & \mathsf{Status} \\
\mathsf{Status} & \sqsubseteq & \mathsf{Aspect} \\
\mathsf{Aspect} & \sqsubseteq & \mathsf{ModifierConcept} \\
\mathsf{ModifierConcept} & \sqsubseteq & \mathsf{DomainCategory}
\end{array}
$$

While $\mathcal{M}^{\mathsf{reach}}_{\mathsf{Lithium}}$ consists of all the 24 listed axioms of NOT-GALEN, $\mathcal{M}^{\mathsf{c.o.i.}}_{\mathsf{L.}\sqsubseteq\mathsf{C.S.}}$ consists of only the first 5 axioms (listed above the separation line).

4.2 Theory Propagation

Another important improvement to the `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, but they were not implemented in the first version of $\mathcal{EL}^+$SAT.) The idea is that the "theory solver" `ComputeOneMinA` can be invoked also when `status = unknown`, that is, when μ is a *partial* truth assignments on $\mathcal{P}_\mathcal{S}$. If μ^+ is enough to falsify the "theory" $\neg s_{[C_i \sqsubseteq D_i]} \wedge \phi^{all}_{\mathcal{T}(po)}$, then `ComputeOneMinA` computes and returns one MinA, without exploring all possible extensions of μ. If μ^+ does not falsify $\neg s_{[C_i \sqsubseteq D_i]} \wedge \phi^{all}_{\mathcal{T}(po)}$, then it may be the case that inside `ComputeOneMinA` μ^+ causes the unit-propagation on $\phi^{all}_{\mathcal{T}(po)}$ of one (or more) literal $\neg s_{[ax_j]}$ s.t. $s_{[ax_j]} \in \mathcal{P}_\mathcal{S}$ and $s_{[ax_j]}$ is unassigned. If this is the case, then $\mu^+ \wedge s_{[ax_j]}$ represents a non-minimal truth assignment falsifying $\neg s_{[C_i \sqsubseteq D_i]} \wedge \phi^{all}_{\mathcal{T}(po)}$. Consequently, (i) the main procedure can unit-propagate $\neg s_{[ax_j]}$, and (ii) $\mu^+ \wedge s_{[ax_j]}$ 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 $\mu^+ \wedge s_{[ax_j]}$ contained a previous MinA, then $\neg s_{[ax_j]}$ 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 $\mathsf{Milk} \sqsubseteq_{\mathcal{O}_{\mathsf{milk}}} \mathsf{Fluid}$ in the ontology $\mathcal{O}_{\mathsf{milk}}$ of Example 2.2 at p. 8, using the encoding $\phi^{all}_{\mathcal{O}_{\mathsf{milk}}(po)}$ of Example 3.3 at p. 23. The query above is labeled by $s_{[n_5]}$. Suppose `ComputeAllMinAs` enumerates the partial assignment $\mu = \mu^+ \stackrel{\text{def}}{=} \{s_{[m_0]}, s_{[m_1]}, s_{[m_2]}, s_{[m_3]}, s_{[m_4]}, s_{[m_5]}\}$, and `ComputeOneMinA` is invoked on $\phi^{all}_{\mathcal{O}_{\mathsf{milk}}(po)}$, μ^+ and $\neg s_{[n_5]}$. This causes the unit-propagation on $\phi^{all}_{\mathcal{O}_{\mathsf{milk}}(po)}$ of $s_{[n_1]}$, $\neg s_{[n_3]}$, $\neg s_{[n_2]}$, and hence $\neg s_{[m_6]}$. Thus $\mu \cup \{s_{[m_6]}\}$ falsifies $\neg s_{[n_5]} \wedge \phi^{all}_{\mathcal{O}_{\mathsf{milk}}(po)}$, and is then minimized by `ComputeOneMinA` into the set $\eta \stackrel{\text{def}}{=} \{s_{[m_0]}, s_{[m_1]}, s_{[m_3]}, s_{[m_5]}, s_{[m_6]}\}$ (corresponding to the MinA $\mathcal{O}^*_{\mathsf{milk}2}$ in Example 3.4 at p. 29). Then `ComputeAllMinAs` learns permanently the clause $\neg\eta$ and adds $\neg s_{[m_6]}$ to μ by unit-propagation, and the search proceeds.

4.3 Working on Smaller Ontologies: $\mathcal{EL}^+$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 $\mathcal{EL}^+$SAT on two different formulas. We first exploit COI modularization to extract a query-dependent sub-ontology on which, subsequently, we run our complete $\mathcal{EL}^+$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 $\mathcal{EL}^+$SAT$\times 2$, consists in three phases:

1. after loading the formula $\phi^{all}_{\mathcal{T}(po)}$, extract the query-dependent sub-ontology $\mathcal{T}_i \stackrel{\text{def}}{=} \mathcal{M}^{\mathsf{c.o.i.}}_{C_i \sqsubseteq D_i}$ by invoking `ComputeCOIModule`;

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

3. enumerate all MinAs applying `ComputeAllMinAs` to $\phi^{all}_{\mathcal{T}_i(po)}$ rather than of to $\phi^{all}_{\mathcal{T}(po)}$.

Implementation-wise, it is convenient to run two distinct instances of $\mathcal{EL}^+$SAT:

(i) a first instance implementing Step 1., which loads only once $\phi^{all}_{\mathcal{T}(po)}$, and then, for every query $C_i \sqsubseteq_{\mathcal{T}} D_i$ [18] given as input, performs COI Modularization, producing $\mathcal{T}_i \stackrel{\text{def}}{=} \mathcal{M}^{\mathsf{c.o.i.}}_{C_i \sqsubseteq D_i}$;

(ii) a second "dynamic" instance of $\mathcal{EL}^+$SAT implementing Steps 2. and 3. for each sub-ontology $\mathcal{T}_i$, producing $\phi^{all}_{\mathcal{T}_i(po)}$ and then using it to enumerate all MinAs for $C_i \sqsubseteq D_i$.

We remark a few facts. First, with the first $\mathcal{EL}^+$SAT instance (i), once $\phi^{all}_{\mathcal{T}(po)}$ has been loaded, a COI module $\mathcal{T}_i$ can be computed in negligible time for an unlimited number of queries $C_i \sqsubseteq D_i$, with no need for reloading $\phi^{all}_{\mathcal{T}(po)}$. Second, since each sub-ontology $\mathcal{T}_i$ is typically much smaller than the original ontology $\mathcal{T}$ (e.g., up to a hundred axioms w.r.t. up to hundreds of thousands axioms), the time taken by the second instance of $\mathcal{EL}^+$SAT (ii) to produce the new formulas $\phi^{all}_{\mathcal{T}_i(po)}$ is typically negligible. Third, overall, $\mathcal{EL}^+$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 $\mathcal{EL}^+$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 `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 $\mathcal{EL}^+$SAT— which includes a customized versions the SAT solver MiniSat2.0 070721 (Eén & 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.[19] 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 $\mathcal{EL}^+$SAT and the other $\mathcal{EL}^+$-specific tool `CEL` (Baader et al., 2006a), version `v.1.1.2`, which implements also the Reachability-based Modularization of Suntisrivaraporn (2009).[20] 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 $\mathcal{EL}^+$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 $\mathcal{S} \subseteq \mathcal{T}$.

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`.

20. Available from `http://lat.inf.tu-dresden.de/systems/cel/` and `http://code.google.com/p/cel/`.

Ontology	notGalen	GeneOnt.	NCI	fullGalen	Snomed'09
# of primitive concepts	2748	20465	27652	23135	310075
# of original axioms	4379	20466	46800	36544	310025
# of normalized axioms	8740	29897	46800	81340	857459
# of role names	413	1	50	949	62
# of role axioms	442	1	0	1014	12

Table 3: Size and structure of the five $\mathcal{EL}^+$ ontologies.

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 $\mathcal{EL}^+$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 concerned, 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 $\mathcal{T}$ we run two groups of 50 test pinpointing queries each, extracted among the subsumption relations which are deducible from $\mathcal{T}$, 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 $C_i \sqsubseteq D_i$ whose selector variable $s_{[C_i \sqsubseteq D_i]}$ appears positively more frequently in $\phi^{all}_{\mathcal{T}(po)}$; 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 $\mathcal{EL}^+$SAT (see Section 4.1) and the Reachability-based Modularization for CEL. It is worth noticing that our modularization technique required negligible time ($\leq 10^{-2}$ secs.) in all the test cases we performed.

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

BASIC (B): the first prototype version used in the feasibility evaluation presented by Sebastiani 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;

$\mathcal{EL}^+$SAT$\times 2$ ($\times 2$): a final version implementing the "two-instances" approach of Section 4.3, which runs two instances of $\mathcal{EL}^+$SAT COI&TP.

	Snomed'09		fullGalen		NCI		GeneOnt.		notGalen		Total
	ran.	sel.	ran.	sel.	ran.	sel.	ran.	sel.	ran.	sel.	
$\mathcal{EL}^+$SAT basic	0	0	0	0	0	0	0	0	0	0	0
$\mathcal{EL}^+$SAT coi	0	1	0	0	0	0	0	0	0	0	1
$\mathcal{EL}^+$SAT coi&tp	0	3	0	0	0	1	0	0	0	0	4
$\mathcal{EL}^+$SAT$\times$2	37	24	32	3	49	49	50	50	35	30	359
CEL	13	22	18	47	1	0	0	0	15	20	136
Total	50	50	50	50	50	50	50	50	50	50	500

Table 4: Number of instances for which each tool has been the best performer.

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.

	# terminated					# MinAs found					Module size avg.	
	$\mathcal{EL}^+$SAT				CEL	$\mathcal{EL}^+$SAT				CEL	$\mathcal{EL}^+$SAT	CEL
test	(B)	(C)	(T)	(×2)		(B)	(C)	(T)	(×2)			
SNOMED'09												
`random`	0	23	24	31	41	146	190	203	463	194	31	46
`selected`	0	0	0	0	21	262	313	575	876	325	129	141
FULLGALEN												
`random`	0	10	15	21	38	74	76	82	82	84	58	8926
`selected`	0	0	0	3	48	55	55	55	55	55	177	14801
NCI												
`random`	0	40	40	46	44	159	172	200	214	150	9	39
`selected`	0	29	29	36	33	417	416	445	455	337	20	48
GENEONT.												
`random`	0	49	49	50	46	132	164	162	168	143	5	20
`selected`	0	17	18	44	8	579	853	881	961	480	20	33
NOTGALEN												
`random`	0	17	34	35	50	67	68	68	68	68	15	95
`selected`	0	0	0	30	50	91	91	91	91	91	31	131
TOTAL	0	185	209	296	379	1982	2398	2762	3433	1927		

(a) Number of terminated problems and generated MinAs.

		Time between MinAs (s)				Total pinpointing time (s)					Modulariz. (s)	
		$\mathcal{EL}^+$SAT				$\mathcal{EL}^+$SAT				CEL	$\mathcal{EL}^+$SAT	CEL
test	%	(B)	(C)	(T)	(×2)	(B)	(C)	(T)	(×2)			
SNOMED'09												
`random`	50th	7.4	1.0	0.5	0.0	99.3	86.9	63.1	0.0	21.4	.000	3.36
	90th	83.	33.	13.	0.9	262.	569.	177.	64.	27.4	.004	3.41
`selected`	50th	7.2	1.5	0.6	0.0	163.	98.	69.	0.4	28.4	.004	3.39
	90th	113.	18.	31.	2.9	655.	402.	791.	322.	31.2	.008	3.41
FULLGALEN												
`random`	50th	3.9	0.5	0.4	0.0	39.9	22.2	25.7	0.0	195.	.002	2.25
	90th	17.6	1.7	0.6	0.0	60.7	24.4	26.4	0.0	1922.	.004	2.27
`selected`	50th	3.7	0.7	0.3	0.0	39.1	36.6	25.1	0.0	321.	.006	2.26
	90th	3.8	0.8	0.4	0.0	39.7	37.2	26.2	0.0	803.	.009	2.27
NCI												
`random`	50th	0.2	0.0	0.0	0.0	2.1	2.0	2.1	0.0	1.3	.001	0.39
	90th	9.1	3.4	6.9	0.2	70.	186.	363.	0.2	4.2	.001	0.39
`selected`	50th	0.3	0.0	0.0	0.0	13.5	2.2	2.3	0.0	2.5	.001	0.39
	90th	31.4	1.0	0.2	0.0	634.	313.	32.	51.	6.8	.001	0.40
GENEONT.												
`random`	50th	0.2	0.0	0.0	0.0	2.1	1.8	1.8	0.0	0.8	.000	0.26
	90th	8.5	0.1	0.1	0.0	21.2	1.9	2.2	0.0	1.4	.001	0.26
`selected`	50th	0.3	0.0	0.0	0.0	292.	144.	90.	1.0	1.8	.001	0.26
	90th	58.7	26.	12.	0.4	845.	885.	885.	52.	2.6	.001	0.27
NOTGALEN												
`random`	50th	0.1	0.0	0.0	0.0	1.0	0.6	0.9	0.0	0.3	.000	0.08
	90th	0.3	0.0	0.0	0.0	1.4	0.6	0.9	0.0	1.3	.001	0.09
`selected`	50th	0.1	0.0	0.0	0.0	1.1	0.9	0.9	0.0	0.7	.001	0.08
	90th	0.2	0.0	0.0	0.0	1.2	1.1	0.9	0.0	1.1	.001	0.09

(b) CPU times in pinpointing and modularization.

Table 5: Summary results of $\mathcal{EL}^+$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 earlyer 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, $\mathcal{EL}^+$SAT$\times 2$, we notice a few facts:

- $\mathcal{EL}^+$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, $\mathcal{EL}^+$SAT$\times 2$ allows for computing some MinAs in negligible time.
- The performances reached by $\mathcal{EL}^+$SAT$\times 2$ for an "average" query (represented by the `random` test suites) are quite satisfactory. Comparing the 50th and 90th percentiles times 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 $\mathcal{EL}^+$SAT$\times 2$.

5.3.2 Comparing $\mathcal{EL}^+$SAT$\times 2$ with CEL

Comparing the data out of the 500 samples on $\mathcal{EL}^+$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.

- $\mathcal{EL}^+$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: $\mathcal{EL}^+$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.
- $\mathcal{EL}^+$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, $\mathcal{EL}^+$SAT$\times 2$ reports less MinAs than CEL only on 6 samples (respectively 1, 4, and 1 in Tables 6, 7, and 8); vice versa, $\mathcal{EL}^+$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)) $\mathcal{EL}^+$SAT$\times 2$ finds 3433 MinAs against the 1927 MinAs found by CEL.
- When both $\mathcal{EL}^+$SAT$\times 2$ and CEL terminate (see the topmost partition of Tables 6-13) $\mathcal{EL}^+$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 $\mathcal{EL}^+$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 $\mathcal{EL}^+$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), $\mathcal{EL}^+$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 $\mathcal{EL}^+$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 $\mathcal{EL}^+$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; Suntisrivaraporn 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; Suntisrivaraporn, 2009; Mendez & Suntisrivaraporn, 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/`

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

References

Baader, F., Brandt, S., & Lutz, C. (2005). Pushing the $\mathcal{EL}$ Envelope. In *Proceedings of IJCAI-05*, pp. 364–369. Morgan-Kaufmann Publishers.

Baader, F., Lutz, C., & Suntisrivaraporn, B. (2006a). CEL—a polynomial-time reasoner for life science ontologies. In *Proceedings of IJCAR'06*, Vol. 4130 of *LNAI*, pp. 287–291. Springer-Verlag.

Baader, F., Lutz, C., & Suntisrivaraporn, B. (2006b). Efficient Reasoning in $\mathcal{EL}^+$. In *Proceedings of DL2006*, Vol. 189 of *CEUR-WS*.

Baader, F., Brandt, S., & Lutz, C. (2008). Pushing the $\mathcal{EL}$ Envelope Further. In *Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. F. (Eds.). (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

Baader, F., & Peñaloza, R. (2007). Axiom Pinpointing in General Tableaux. In *Proceedings of TABLEAUX 2007*, LNAI, pp. 11–27. Springer.

Baader, F., Peñaloza, R., & Suntisrivaraporn, B. (2007). Pinpointing in the Description Logic $\mathcal{EL}^+$. In *Proceedings of KI2007*, Vol. 4667 of *LNCS*, pp. 52–67. Springer.

Baader, F., & Suntisrivaraporn, B. (2008). Debugging SNOMED CT Using Axiom Pinpointing in the Description Logic $\mathcal{EL}^+$. In *Proceedings of KR-MED'08*, Vol. 410 of *CEUR-WS*.

Bail, S., Horridge, M., Parsia, B., & Sattler, U. (2011). The Justificatory Structure of the NCBO BioPortal Ontologies. In *Proceedings of ISWC 2011*, Vol. 7031 of *LNCS*, pp. 67–82. Springer.

Barrett, C. W., Sebastiani, R., Seshia, S. A., & Tinelli, C. (2009). *Satisfiability Modulo Theories*, chap. 26, pp. 825–885. Vol. 185 of Biere et al. (Biere et al., 2009).

Bienvenu, M. (2008). Complexity of Abduction in the $\mathcal{EL}$ Family of Lightweight Description Logics. In *Proceedings of KR2008*, pp. 220–230. AAAI Press.

Biere, A., Heule, H. J. M., van Maaren, H., & Walsh, T. (Eds.). (2009). *Handbook of Satisfiability*, Vol. 185 of *FAIA*. IOS Press.

Biere, A. (2008). Picosat essentials. *Journal on Satisfiability, Boolean Modeling and Computation, JSAT*, *4*(2-4), 75–97.

Cimatti, A., Griggio, A., & Sebastiani, R. (2011). Computing small unsatisfiable cores in satisfiability modulo theories. *Journal of Artificial Intelligence Research, JAIR*, *40*, 701–728.

Clarke, E., Grumberg, O., & Peled, D. A. (1999). *Model Checking*. MIT Press.

Davis, M., Longemann, G., & Loveland, D. (1962). A Machine Program for Theorem-proving. *Communications of the ACM*, *5*, 394–397.

Davis, M., & Putnam, H. (1960). A Computing Procedure for Quantification Theory. *Journal of the ACM*, *7*, 201–215.

Del Vescovo, C., Parsia, B., Sattler, U., & Schneider, T. (2010). The modular structure of an ontology: an empirical study. In *Proceedings of Description Logics 2010*, Vol. 573 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Del Vescovo, C., Parsia, B., Sattler, U., & Schneider, T. (2011). The Modular Structure of an Ontology: Atomic Decomposition. In *Proceedings of IJCAI'11*, pp. 2232–2237. IJCAI/AAAI.

Eén, N., & Biere, A. (2005). Effective Preprocessing in SAT Through Variable and Clause Elimination. In *Proceedings of SAT'05*, Vol. 3569 of *LNCS*, pp. 61–75. Springer.

Eén, N., & Sörensson, N. (2004). An Extensible SAT-solver. In *Proceedings of SAT'03*, Vol. 2919 of *LNCS*, pp. 502–518. Springer.

Giunchiglia, F., & Sebastiani, R. (1996). A SAT-based decision procedure for ALC. In *Proc. of the 5th International Conference on Principles of Knowledge Representation and Reasoning - KR'96*, Cambridge, MA, USA.

Giunchiglia, F., & Sebastiani, R. (2000). Building decision procedures for modal logics from propositional decision procedures - the case study of modal K(m). *Information and Computation*, *162*(1/2).

Grau, B. C., Horrocks, I., Kazakov, Y., & Sattler, U. (2007). Just the right amount: extracting modules from ontologies. In *Proceedings of WWiW-07*, pp. 717–726. ACM.

Haarslev, V., Sebastiani, R., & Vescovi, M. (2011). Automated Reasoning in $\mathcal{ALCQ}$ via *SMT*. In *Proceedings of CADE-23*, Vol. 6803 of *LNCS*, pp. 283–298. Springer.

Horridge, M. (2011). *Justification Based Explanations In Ontologies*. Ph.D. thesis, School of Computer Science, University of Manchester.

Horridge, M., Parsia, B., & Sattler, U. (2008). Laconic and Precise Justifications in OWL. In *Proceedings of ISWC'08*, Vol. 5318 of *LNCS*, pp. 323–338.

Kalyanpur, A., Parsia, B., Horridge, M., & Sirin, E. (2007). Finding All Justifications of OWL DL Entailments. In *Proceedings of 6th ISWC/ASWC*, Vol. 4825 of *LNCS*, pp. 267–280. Springer.

Kalyanpur, A., Parsia, B., Sirin, E., & Hendler, J. A. (2005). Debugging unsatisfiable classes in OWL ontologies. *Journal of Web Semantics*, *3*(4), 268–293.

Kazakov, Y. (2009). Consequence-Driven Reasoning for Horn SHIQ Ontologies. In *Proceedings of IJCAI-09*, pp. 2040–2045.

Kazakov, Y., Kroetzsch, M., & Simancik, F. (2012). Practical Reasoning with Nominals in the EL Family of Description Logics. In *Proceedings of KR2012*. AAAI Press.

Kazakov, Y., & Krötzsch, M. (2013). The Incredible ELK: From Polynomial Procedures to Efficient Reasoning with $\mathcal{EL}$ Ontologies. *Journal of Automated Reasoning*, *52*(2), 1–61. To appear, see `http://link.springer.com/article/10.1007%2Fs10817-013-9296-3`.

Konev, B., Lutz, C., Walther, D., & Wolter, F. (2008a). Logical Difference and Module Extraction with CEX and MEX. In *Proceedings of DL2008*, Vol. 353 of *CEUR-WS*.

Konev, B., Lutz, C., Walther, D., & Wolter, F. (2008b). Semantic Modularity and Module Extraction in Description Logics. In *Proceedings of ECAI'08*, Vol. 178 of *FAIA*, pp. 55–59. IOS Press.

Konev, B., Walther, D., & Wolter, F. (2008c). The logical difference problem for description logic terminologies. In *Proceedings of IJCAR'08*, Vol. 5195 of *LNCS*, pp. 259–274. Springer.

Lahiri, S. K., Nieuwenhuis, R., & Oliveras, A. (2006). SMT Techniques for Fast Predicate Abstraction. In *Proceedings of CAV'06*, Vol. 4144 of *LNCS*, pp. 424–437. Springer.

Lawley, M. J., & Bousquet, C. (2010). Fast classification in Protg: Snorocket as an OWL 2 EL reasoner. In Inc, A. C. S. (Ed.), *Proceedings of IAOA-10*, Vol. 122 of *Conferences in Research and Practice in Information Technology*, pp. 45–49.

Lutz, C., Toman, D., & Wolter, F. (2009). Conjunctive Query Answering in the Description Logic $\mathcal{EL}$ Using a Relational Database System. In *Proceedings of IJCAI-09*, pp. 2070–2075. AAAI Press.

Lynce, I., & Silva, J. P. (2004). On Computing Minimum Unsatisfiable Cores. In *Proceedings of SAT'04*, pp. 305–310.

Magka, D., Kazakov, Y., & Horrocks, I. (2010). Tractable Extensions of the Description Logic $\mathcal{EL}$ with Numerical Datatypes. In *Proceedings of IJCAR'10*, LNCS, pp. 61–75. Springer.

Mendez, J., Ecke, A., & Turhan, A. Y. (2011). Implementing completion-based inferences for the EL-family. In *Proceedings of Description Logic 2011*, Vol. 745 of *CEUR Workshop Proceedings*, pp. 334–344. CEUR-WS.org.

Mendez, J., & Suntisrivaraporn, B. (2009). Reintroducing CEL as an OWL 2 EL Reasoner. In *Proceedings of the 2009 International Workshop on Description Logics (DL2009)*, Vol. 477 of *CEUR-WS*.

Moodley, K., Meyer, T., & Varzinczak, I. J. (2011). Root Justifications for Ontology Repair. In *Proceedings of RR 2011*, Vol. 6902 of *LNCS*, pp. 275–280. Springer.

Moskewicz, M. W., Madigan, C. F., Zhao, Y., Zhang, L., & Malik, S. (2001). Chaff: Engineering an Efficient SAT Solver. In *Proceedings of DAC'01*, pp. 530–535. ACM.

Nieuwenhuis, R., Oliveras, A., & Tinelli, C. (2006). Solving SAT and SAT Modulo Theories: from an Abstract Davis-Putnam-Logemann-Loveland Procedure to DPLL(T). *Journal of the ACM*, *53*(6), 937–977.

Noy, N. F., & Musen, M. A. (2003). The PROMPT suite: interactive tools for ontology merging and mapping. *Int. Journal of Human-Computing Studies*, *59*, 983–1024.

Peñaloza, R., & Sertkaya, B. (2010a). Complexity of Axiom Pinpointing in the DL-Lite Family of Description Logics. In *Proceedings of ECAI'10*, Vol. 215 of *FAIA*, pp. 29–34. IOS Press.

Peñaloza, R., & Sertkaya, B. (2010b). On the Complexity of Axiom Pinpointing in the $\mathcal{EL}$ Family of Description Logics. In *Proceedings of KR2010*, pp. 280–289. AAAI Press.

Plaisted, D., & Greenbaum, S. (1986). A Structure-preserving Clause Form Translation. *Journal of Symbolic Computation*, *2*, 293–304.

Rector, A., & Horrocks, I. (1997). Experience Building a Large, Re-usable Medical Ontology using a Description Logic with Transitivity and Concept Inclusions. In *Proceedings of the Workshop on Ontological Engineering, AAAI'97*. AAAI Press.

Schlobach, S., & Cornet, R. (2003). Non-standard reasoning services for the debugging of description logic terminologies. In *Proceeding of IJCAI-03*, pp. 355–362. Morgan Kaufmann.

Sebastiani, R. (2007). Lazy Satisfiability Modulo Theories. *Journal on Satisfiability, Boolean Modeling and Computation, JSAT*, *3*, 141–224.

Sebastiani, R., & Vescovi, M. (2009a). Automated Reasoning in Modal and Description Logics via SAT Encoding: the Case Study of $K(m)/\mathcal{ALC}$-Satisfiability. *Journal of Artificial Intelligence Research, JAIR*, *35*(1), 275–341.

Sebastiani, R., & Vescovi, M. (2009b). Axiom Pinpointing in Lightweight Description Logics via Horn-SAT Encoding and Conflict Analysis. In *Proceedings of CADE-22*, Vol. 5663 of *LNCS*, pp. 84–99. Springer. Also available at `http://disi.unitn.it/~rseba/publist.html`.

Seidenberg, J., & Rector, A. (2006). Web Ontology Segmentation: Analysis, Classification and Use. In *Proceedingd of WWW'06*, pp. 13–22. ACM.

Silva, J. P., & Sakallah, K. A. (1996). GRASP – A new Search Algorithm for Satisfiability. In *Proceedings of ICCAD'96*, pp. 220–227. IEEE Computer Society.

Sioutos, N., de Coronado, S., Haber, M. W., Hartel, F. W., Shaiu, W., & Wright, L. W. (2007). NCI Thesaurus: A semantic model integrating cancer-related clinical and molecular information. *Journal of Biomedical Informatics*, *40*(1), 30–43.

Spackman, K. A. (2000). Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED RT. *Journal of American Medical Informatics Association, JAMIA (Fall Symposium Special Issue)*.

Spackman, K. A., Campbell, K. E., & Cote, R. A. (1997). SNOMED RT: A reference terminology for health care. *Journal of American Medical Informatics Association, JAMIA (Proceedings of the Fall Symposium Supplement)*.

Suntisrivaraporn, B. (2009). *Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies*. Ph.D. thesis, University of Dresden.

Suntisrivaraporn, B., Baader, F., Schulz, S., & Spackman, K. A. (2007). Replacing sep-triplets in snomed ct using tractable description logic operators. In *Proceedings of AIME'07*, LNCS. Springer-Verlag.

Suntisrivaraporn, B., Qi, G., Ji, Q., & Haase, P. (2008). A Modularization-Based Approach to Finding All Justifications for OWL DL Entailments. In *Proceedings of ASWC'08*, pp. 1–15. Springer-Verlag.

The G. O. Consortium (2000). Gene ontology: Tool for the unification of biology. *Nature Genetics*, *25*, 25–29.

Zhang, L., Madigan, C. F., Moskewicz, M. W., & Malik, S. (2001). Efficient Conflict Driven Learning in Boolean Satisfiability Solver. In *Proceedings of ICCAD'01*, pp. 279–285.

Zhang, L., & Malik, S. (2002). The Quest for Efficient Boolean Satisfiability Solvers. In *Proceedings of CAV'02*, No. 2404 in LNCS, pp. 17–36. Springer.

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 $\mathsf{PC}_\mathcal{T}$, $C \sqsubseteq_\mathcal{T} D$ if and only if the Horn propositional formula $\phi_\mathcal{T} \wedge p_{[C]} \wedge \neg p_{[D]}$ 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 $C \sqsubseteq_\mathcal{T} D$. Thus, if $C \sqsubseteq_\mathcal{T} D$ then $\phi_\mathcal{T} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable.

(**If.**) Vice versa, if $\phi_\mathcal{T} \wedge p_{[C]} \wedge \neg p_{[D]}$ 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 $\psi_1, ..., \psi_n$ in $\phi_\mathcal{T}$. Since $\phi_\mathcal{T}$ is a definite Horn formula, every resolution step can be written in the form

$$\frac{(\bigwedge_i p_{[X_i]}) \rightarrow p_{[X_k]} \qquad (p_{[X_k]} \wedge \bigwedge_j p_{[X_j]}) \rightarrow p_{[X_n]}}{(\bigwedge_i p_{[X_i]} \wedge \bigwedge_j p_{[X_j]}) \rightarrow p_{[X_n]}} \tag{11}$$

s.t. all X's are concepts. Each corresponding derivation step

$$\frac{(\bigsqcap_i X_i) \sqsubseteq X_k \qquad (X_k \sqcap \bigsqcap_j X_j) \sqsubseteq X_n}{(\bigsqcap_i X_i \sqcap \bigsqcap_j X_j) \sqsubseteq X_n} \tag{12}$$

is valid in $\mathcal{EL}^+$. [24] Moreover, by definition, each ψ_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 $C \sqsubseteq_\mathcal{T} D$. □

Proposition 2. *The size of the formula $\phi^{all}_\mathcal{T}$ 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 $r(a_i, a_{i'}, a_j, a_k)$ on the axioms and assertions in $\mathcal{A}$ is upper-bounded by $|\mathcal{A}|^2 \cdot |\mathcal{T}|$. Recalling that $|\mathcal{A}| \leq |\mathsf{PC}_\mathcal{T}|^2 \cdot |\mathsf{PR}_\mathcal{T}|$ and hence $|\mathcal{A}|$ is polynomial in $|\mathcal{T}|$, we have the thesis. □

Lemma 9. *A literal ℓ can be derived by unit-propagation from $\phi^{all}_\mathcal{T} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]}$ [resp. $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]}$] if and only if ℓ is a positive occurrence of a selector variable $s_{[a_k]}$ for some subsumption relation a_k which is deducible from $\mathcal{S}$.*

Proof.
(**If.**) If the subsumption relation a_k is deducible from $\mathcal{S}$ 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 $\mathcal{S}' \subseteq \mathcal{S}$, and such that $r_N = r(a_i, a_{i'}, a_j, a_k)$ for some rule r. We reason by induction on the number N of rules applied in order to deduce a_k from $\mathcal{S}'$.

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 $\mathcal{S}'$. By construction, the rule clause (10) $c_{r_N} = (s_{[a_i]} \wedge s_{[a_{i'}]} \wedge s_{[a_j]}) \rightarrow s_{[a_k]}$ is included in $\phi^{all}_{\mathcal{T}(po)}$, and thus $s_{[a_k]}$ is derived by unit-propagation from $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]}$ where all the axiom selector variables relative to the axioms of $\mathcal{S}$ 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, $\phi^{all}_{\mathcal{T}(po)}$ includes the corresponding set of rule clauses (10) $c_{r_1}, \ldots, c_{r_N}$, where $c_{r_N} = (s_{[a_i]} \wedge s_{[a_{i'}]} \wedge s_{[a_j]}) \rightarrow s_{[a_k]}$ and $a_i, a_{i'}$ and a_j are either axioms of $\mathcal{S}'$ 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 $\phi^{all}_{\mathcal{T}(po)}$ is a subformula of $\phi^{all}_{\mathcal{T}}$, 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 $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]}$:

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

Step: Let $\mathcal{L}$ be the set of all literals derived by N applications of unit-propagations steps and assume that all literals in $\mathcal{L}$ are in the form $s_{[a_j]}$. 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 $\phi^{all}_{\mathcal{T}(po)}$) do. Thus only positive selection variables $s_{[a_k]}$ can be propagated, and only from rule clauses (10) in the form $(s_{[a_i]} \wedge s_{[a_{i'}]} \wedge s_{[a_j]}) \rightarrow s_{[a_k]}$ such that $\{s_{[a_i]}, s_{[a_{i'}]}, s_{[a_j]}\} \subseteq \mathcal{L}$ 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 $\mathcal{S}$, so that a_k is also derived from $\mathcal{S}$ by means of r.

We remark that the literals which can be unit-propagated from $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]}$ are all and only those which can be unit-propagated from $\phi^{all}_{\mathcal{T}} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]}$ because, as above stated at point (ii), assertion clauses (9) play no role in unit-propagations. □

Theorem 3. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form, for every $\mathcal{S} \subseteq \mathcal{T}$ and for every pair of concept names C, D in $\mathsf{PC}_{\mathcal{T}}$, $C \sqsubseteq_{\mathcal{S}} D$ if and only if the Horn propositional formula $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge \neg s_{[C \sqsubseteq D]}$ is unsatisfiable.*

Proof. Due to the soundness and completeness of the set of classification rules reported in Section 2.1, $C \sqsubseteq_{\mathcal{S}} D$ if and only if there exists a sequence of rule applications $r_1, \ldots, r_N$ generating $C \sqsubseteq D$ from $\mathcal{S}$. If and only if this is the case, by definition, $\phi^{all}_{\mathcal{T}(po)}$ contains all the rule clauses of type (10) corresponding to all the rule applications $r_1, \ldots, r_N$.

(**Only if.**) Thus, on the one hand, if $C \sqsubseteq_{\mathcal{S}} D$, then $\bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]}$ and all the clauses of type (10) corresponding to all the rule applications $r_1, \ldots, r_N$, from Lemma 9, force $s_{[C \sqsubseteq D]}$ to be true, which falsifies the unit clause $\neg s_{[C \sqsubseteq D]}$. Thus $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge \neg s_{[C \sqsubseteq D]}$ is unsatisfiable.

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

Theorem 4. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form, for every $\mathcal{S} \subseteq \mathcal{T}$ and for every pair of concept names C, D in $\mathsf{PC}_{\mathcal{T}}$, $C \sqsubseteq_{\mathcal{S}} D$ if and only if the Horn propositional formula $\phi^{all}_{\mathcal{T}} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable.*

Proof.
(**Only if.**) Suppose $C \sqsubseteq_{\mathcal{S}} D$. By Theorem 3, since $\phi^{all}_{\mathcal{T}} \stackrel{\text{def}}{=} \phi_{\mathcal{T}(so)} \wedge \phi^{all}_{\mathcal{T}(po)}$, we have that $\phi^{all}_{\mathcal{T}} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \models s_{[C \sqsubseteq D]}$. Moreover, if $C \sqsubseteq_{\mathcal{S}} D$ then $C \sqsubseteq_{\mathcal{T}} D$; thus, by Definition 4 and due to the soundness and completeness of the classification rules, $\phi^{all}_{\mathcal{T}}$ contains the assertion clause

$$s_{[C \sqsubseteq D]} \rightarrow (p_{[C]} \rightarrow p_{[D]}). \tag{13}$$

Thus $\phi^{all}_{\mathcal{T}} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable.

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

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

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

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

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

Since $\phi^*_{\mathcal{T}} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable, then $\phi^*_{\mathcal{T}} \models p_{[C]} \rightarrow p_{[D]}$. 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 $\psi_1, ..., \psi_n$ in $\phi^*_{\mathcal{T}}$.

We notice that all ψ_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. [25] Thus, since $\phi^*_{\mathcal{T}}$ is a definite Horn formula, every resolution step in P can be written in the form

$$\frac{(\bigwedge_i p_{[X_i]}) \rightarrow p_{[X_k]} \qquad (p_{[X_k]} \wedge \bigwedge_j p_{[X_j]}) \rightarrow p_{[X_n]}}{(\bigwedge_i p_{[X_i]} \wedge \bigwedge_j p_{[X_j]}) \rightarrow p_{[X_n]}} \tag{14}$$

s.t. all X's are concepts. Each corresponding derivation step

$$\frac{(\bigsqcap_i X_i) \sqsubseteq X_k \qquad (X_k \sqcap \bigsqcap_j X_j) \sqsubseteq X_n}{(\bigsqcap_i X_i \sqcap \bigsqcap_j X_j) \sqsubseteq X_n} \tag{15}$$

is valid in $\mathcal{EL}^+$. [26] Moreover, each ψ_i is $\mathcal{EL}^+2SAT(a_i)$ for some a_i which is either in $\mathcal{S}$ or has been derived from $\mathcal{S}$. Hence, there is a valid derivation of $C \sqsubseteq D$ from $\mathcal{S}$ in $\mathcal{EL}^+$, so that $C \sqsubseteq_{\mathcal{S}} D$.

Notice that from the latter fact we can also conclude that the clause $p_{[C]} \rightarrow p_{[D]}$ belongs to $\phi_{\mathcal{S}}$ and therefore, by Definition 4, that (13) belongs to $\phi^{all}_{\mathcal{S}}$ (more precisely $\phi_{\mathcal{S}(so)}$). Thus (13) belongs also to $\phi^{all}_{\mathcal{T}}$. □

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

Corollary 5. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form, for every pair of concept names C, D in $\mathsf{PC}_{\mathcal{T}}$, $C \sqsubseteq_{\mathcal{T}} D$ if and only if the Horn propositional formula $\phi^{all}_{\mathcal{T}} \wedge \bigwedge_{a_i \in \mathcal{T}} s_{[ax_i]} \wedge p_{[C]} \wedge \neg p_{[D]}$ [resp. $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{T}} s_{[ax_i]} \wedge \neg s_{[C \sqsubseteq D]}$] 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 $\mathcal{T}$ in normal form and the formula $\phi^{all}_{\mathcal{T}(po)}$ as defined in Definition 4, for every pair of concept names C, D in $\mathsf{PC}_{\mathcal{T}}$, the following facts hold:*

(i) $C \sqsubseteq_{\mathcal{T}} D$ if and only if $C \sqsubseteq_{\mathcal{M}^{\mathsf{c.o.i.}}_{C \sqsubseteq D}} D$;

(ii) if $\mathcal{S}$ is a MinA for $C \sqsubseteq_{\mathcal{T}} D$, then $\mathcal{S} \subseteq \mathcal{M}^{\mathsf{c.o.i.}}_{C \sqsubseteq D}$,

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

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 $\mathcal{M}^{\mathsf{c.o.i.}}_{C_i \sqsubseteq D_i} \subseteq \mathcal{T}$ *is the cone-of-influence module for* $C \sqsubseteq D$ *w.r.t.* $\phi^{all}_{\mathcal{T}(po)}$*, as defined in Definition 5.*

Proof.

(i) By Theorem 3 and by Definition 5 of $\mathcal{M}^{\mathsf{c.o.i.}}_{C \sqsubseteq D}$ w.r.t. $\phi^{all}_{\mathcal{T}(po)}$, in place of fact (i), we can equivalently prove that the Horn propositional formula $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{T}} s_{[ax_i]} \wedge \neg s_{[C \sqsubseteq D]}$, namely $\psi^{C \sqsubseteq D}_{\mathcal{T}}$, is unsatisfiable if and only if the Horn propositional formula $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{M}^{\mathsf{c.o.i.}}_{C \sqsubseteq D}} s_{[ax_i]} \wedge \neg s_{[C \sqsubseteq D]}$, namely $\psi^{C \sqsubseteq D}_{\mathcal{M}^{\mathsf{c.o.i.}}_{C \sqsubseteq D}}$, is unsatisfiable. On the one hand, since $\psi^{C \sqsubseteq D}_{\mathcal{T}} = \psi^{C \sqsubseteq D}_{\mathcal{M}^{\mathsf{c.o.i.}}_{C \sqsubseteq D}} \wedge \bigwedge_{ax_i \in (\mathcal{T} \setminus \mathcal{M}^{\mathsf{c.o.i.}}_{C \sqsubseteq D})} s_{[ax_i]}$, if $\psi^{C \sqsubseteq D}_{\mathcal{M}^{\mathsf{c.o.i.}}_{C \sqsubseteq D}}$ is unsatisfiable then $\psi^{C \sqsubseteq D}_{\mathcal{T}}$ is also trivially unsatisfiable. On the other hand, we consider any minimal subset $\mathcal{S}$ of $\mathcal{T}$ such that $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge \neg s_{[C \sqsubseteq D]}$ 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 $\mathcal{T}$ such that $C \sqsubseteq_{\mathcal{S}} D$ and $C \not\sqsubseteq_{\mathcal{S}'} D$ for every $\mathcal{S}' \subset \mathcal{S}$. By Theorem 3 we have that $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge \neg s_{[C \sqsubseteq D]}$ is unsatisfiable. Since $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]}$ is trivially satisfiable, it follows that $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \models s_{[C \sqsubseteq D]}$. But since, by Theorem 3, $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in (\mathcal{S} \setminus \{ax_j\})} s_{[ax_i]} \wedge \neg s_{[C \sqsubseteq D]}$ is instead satisfiable for every $ax_j \in \mathcal{S}$, then every $s_{[ax_j]}$ is included in the subformula of $\phi^{all}_{\mathcal{T}(po)}$ responsible for the deduction of $s_{[C \sqsubseteq D]}$ (i.e. $s_{[ax_j]}$ is included in $\phi_{\mathcal{P}^{C \sqsubseteq D}_{\mathcal{T}}}$), so that $s_{[ax_j]} \in \mathcal{P}^{C \sqsubseteq D}_{\mathcal{T}}$, for every $ax_j \in \mathcal{S}$. Accordingly, by Definition 5, $ax_j \in \mathcal{M}^{\mathsf{c.o.i.}}_{C \sqsubseteq D}$ for every $ax_j \in \mathcal{S}$ and we have the thesis $\mathcal{S} \subseteq \mathcal{M}^{\mathsf{c.o.i.}}_{C \sqsubseteq D}$.

□

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

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 $\mathcal{P}^{C_i \sqsubseteq D_i}_{\mathcal{A}}$. In each iteration, once dequeued the selector variable $s_{[a_i]}$ at Step `4`, the algorithm: (i) gets the clauses ϕ_{a_i} (Step `5`), (ii) executes a fixed number of instructions for every literal of every clause $c \in \phi_{a_i}$ (Steps `6-13`). Since (i) is performed in linear time w.r.t. the number of clauses in ϕ_{a_i} 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 $O(|\phi^{C_i \sqsubseteq D_i}_{\mathcal{T}}|)$ (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 $\mathcal{T}$ in normal form and the formula $\phi^{all}_{\mathcal{T}(po)}$ as defined in Definition 4, for every pair of concept names C, D in $\mathsf{PC}_\mathcal{T}$ it holds $\mathcal{M}^{\mathsf{c.o.i.}}_{C\sqsubseteq D} \subseteq \mathcal{M}^{\mathsf{reach}}_{C}$, where $\mathcal{M}^{\mathsf{reach}}_{C}$ and $\mathcal{M}^{\mathsf{c.o.i.}}_{C\sqsubseteq D}$ 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 $\mathcal{M}^{\mathsf{c.o.i.}}_{C\sqsubseteq D}$ for $C \sqsubseteq_\mathcal{T} D$ is defined w.r.t. the formula $\phi^{all}_{\mathcal{T}(po)}$, 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 $\mathcal{M}^{\mathsf{c.o.i.}}_{X\sqsubseteq \hat{D}} \subseteq \mathcal{M}^{\mathsf{reach}}_{X}$ for every pair of concept names $X \in \mathsf{PC}_\mathcal{T}$ and $\hat{D} \in \mathsf{NC}_\mathcal{T}$.

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 $X \sqsubseteq \hat{D}$ (third column), then such a completion rule is encoded into $\phi^{all}_{\mathcal{T}(po)}$ and, accordingly, the axiom ax becomes part of the cone-of-influence module $\mathcal{M}^{\mathsf{c.o.i.}}_{X\sqsubseteq \hat{D}}$ for $X \sqsubseteq \hat{D}$. 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 $\mathcal{T}$. In particular we prove that $ax \in \mathcal{M}^{\mathsf{c.o.i.}}_{X\sqsubseteq \hat{D}}$ implies $ax \in \mathcal{M}^{\mathsf{reach}}_{X}$.

Notice the following two facts:

(i) : given a subsumption relation a, the cone-of-influence module $\mathcal{M}^{\mathsf{c.o.i.}}_{a}$ for a is fully included in any cone-of-influence module $\mathcal{M}^{\mathsf{c.o.i.}}_{a'}$, with a' any subsumption relation such that $s_{[a]}$ is in the cone of influence $\mathcal{P}^{a'}_{\mathcal{T}}$ 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 $X \sqsubseteq C_1, \ldots, X \sqsubseteq C_k$ (with $k \geq 1$), and on the axiom $ax \stackrel{\text{def}}{=} C_1 \sqcap \cdots \sqcap C_k \sqsubseteq D$. Thus, $ax \in \mathcal{M}^{\mathsf{c.o.i.}}_{X\sqsubseteq \hat{D}}$. 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 $X \sqsubseteq C$, the symbol C and the axiom $ax \stackrel{\text{def}}{=} C \sqsubseteq \exists r.D$.

3. Suppose that the third completion rule is applied on the previously deduced subsumptions $X \sqsubseteq \exists r.E$ and $E \sqsubseteq C$, and on the axiom $ax = \exists r.C \sqsubseteq D$. Thus, $ax \in \mathcal{M}^{\mathsf{c.o.i.}}_{X\sqsubseteq \hat{D}}$. 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 $X \sqsubseteq \exists r.C$, the symbol r and the axiom $ax \stackrel{\text{def}}{=} r \sqsubseteq s$.

5. Suppose that the fifth completion rule is applied on the previously deduced subsumptions $X \sqsubseteq \exists r_1.E_1, \ldots, E_{n-1} \sqsubseteq \exists r_n.D$ (with $n \geq 2$), and on the axiom $ax \stackrel{\text{def}}{=} r_1 \circ \cdots \circ r_n$. Thus, $ax \in \mathcal{M}^{\mathsf{c.o.i.}}_{X \sqsubseteq \hat{D}}$. 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 $\phi^{all}_{\mathcal{T}(po)}$ that if $ax \in \mathcal{M}^{\mathsf{c.o.i.}}_{X \sqsubseteq \hat{D}}$ 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 $\mathcal{M}^{\mathsf{c.o.i.}}_{X \sqsubseteq \hat{D}} \subseteq \mathcal{M}^{\mathsf{reach}}_{X}$. □

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 results of modularization, the CPU times required by the tools and the number of MinAs resulting from the search. In particular, in the left-most part of the tables 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$\times 2$ 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$\times 2$ (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 $\phi^{all}_{\mathcal{T}}/\phi^{all}_{\mathcal{T}(po)}$ 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 ($\times 2$) 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 by $\mathcal{EL}^+$SAT$\times 2$ and/or by CEL. Thus, e.g., the problems which have been completely solved both by $\mathcal{EL}^+$SAT$\times 2$ and by CEL (with both $\mathcal{EL}^+$SAT$\times 2$ 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.

	Module size		MinAs search time (sec.)						#MinAs				
	$\mathcal{EL}^+$SAT	CEL	$\mathcal{EL}^+$SAT					CEL	$\mathcal{EL}^+$SAT				CEL
			(B)	(C)	(T)	*enc.*	(×2)		(B)	(C)	(T)	(×2)	
1	9	31	19.5	3.7*	1.0*	*0.0*	**0.0***	21.4*	2	3	3	3	3
2	2	17	6.5	0.6*	0.5*	*0.0*	**0.0***	20.5*	1	1	1	1	1
5	2	12	6.5	0.6*	0.5*	*0.0*	**0.0***	20.5*	1	1	1	1	1
6	7	9	6.7	0.8*	0.7*	*0.0*	**0.0***	20.6*	1	1	1	1	1
7	7	9	19.5	0.9*	0.9*	*0.0*	**0.0***	20.8*	2	2	2	2	2
10	4	15	13.0	0.7*	0.7*	*0.0*	**0.0***	20.8*	2	2	2	2	2
11	10	27	65.4	7.0*	14.0*	*0.0*	**0.0***	22.0*	4	5	5	5	5
12	5	21	6.6	0.7*	0.6*	*0.0*	**0.0***	20.7*	1	1	1	1	1
13	10	47	38.7	21.1*	2.7*	*0.0*	**0.0***	22.6*	3	6	6	6	6
17	11	22	13.3	1.0*	1.1*	*0.0*	**0.0***	21.3*	2	2	2	2	2
18	6	10	6.6	0.7*	0.6*	*0.0*	**0.0***	20.6*	1	1	1	1	1
19	2	3	6.5	0.6*	0.5*	*0.0*	**0.0***	20.5*	1	1	1	1	1
21	8	42	26.2	1.1*	1.2*	*0.0*	**0.0***	21.4*	3	3	3	3	3
23	5	11	6.6	0.7*	0.6*	*0.0*	**0.0***	20.6*	1	1	1	1	1
24	4	6	6.6	0.7*	0.6*	*0.0*	**0.0***	20.6*	1	1	1	1	1
28	3	42	6.5	0.6*	0.6*	*0.0*	**0.0***	20.7*	1	1	1	1	1
29	2	54	6.5	0.6*	0.5*	*0.0*	**0.0***	20.6*	1	1	1	1	1
31	3	6	6.5	0.6*	0.5*	*0.0*	**0.0***	20.5*	1	1	1	1	1
32	7	9	25.8	0.9*	0.9*	*0.0*	**0.0***	20.8*	2	2	2	2	2
33	8	14	38.7	1.2*	1.2*	*0.0*	**0.0***	21.3*	3	3	3	3	3
36	5	8	6.6	0.7*	0.6*	*0.0*	**0.0***	20.6*	1	1	1	1	1
39	9	10	6.7	0.8*	0.7*	*0.0*	**0.0***	20.6*	1	1	1	1	1
45	14	44	104.4	136.0*	256.6*	*0.0*	**0.0***	22.7*	4	4	4	4	4
14	6	91	6.6	0.7	0.6*	*0.0*	**0.0***	20.9*	1	1	1	1	1
26	20	39	64.7	3.2	1.1	*0.0*	**0.0***	21.5*	2	2	2	2	2
38	16	25	39.6	521.0	32.1	*0.0*	**0.1***	23.0*	4	7	7	7	7
40	16	25	6.5	0.6	0.6	*0.0*	**0.0***	20.6*	1	1	1	1	1
47	19	27	52.1	2.1	1.7	*0.0*	**0.0***	22.1*	3	3	3	3	3
8	24	53	59.4	552.4	114.2	*0.0*	**8.6***	27.2	5	15	20	20	10
20	23	55	146.8	483.5	31.3	*0.0*	**43.0***	26.1	6	10	14	26	10
46	19	21	284.5	409.4	114.9	*0.0*	**1.0***	23.9	7	10	10	12	10
3	47	66	92.8	125.6	34.2	*0.0*	249.1	**33.2***	4	4	7	8	8
4	38	46	6.8	1.0	0.7	*0.0*	0.0	**21.0***	1	1	1	1	1
9	75	88	19.7	1.2	1.0	*0.0*	0.0	**21.9***	2	2	2	2	2
15	65	75	72.3	243.5	64.2	*0.0*	0.4	**27.0***	3	6	6	6	6
16	70	74	85.4	6.1	1.3	*0.0*	0.0	**22.6***	3	3	3	3	3
22	48	62	27.4	126.9	473.6	*0.0*	1.2	**30.1***	3	4	5	5	8
25	44	50	13.7	2.4	1.1	*0.0*	0.0	**21.7***	2	2	2	2	2
30	57	63	182.1	501.3	2.1	*0.0*	0.0	**22.8***	3	3	3	3	3
34	147	161	6.7	0.8	0.7	*0.0*	0.0	**21.3***	1	1	1	1	1
35	81	87	19.3	1.0	0.7	*0.0*	0.0	**21.0***	2	2	2	2	2
37	52	58	6.7	0.8	0.7	*0.0*	0.0	**21.0***	1	1	1	1	1
41	117	134	39.4	665.3	252.7	*0.0*	2.0	**25.0***	3	5	5	5	5
43	62	78	136.8	10.6	2.1	*0.0*	0.0	**23.7***	2	2	3	3	3
27	37	89	642.6	75.5	10.0	*0.0*	63.6	29.9	9	10	9	**13**	10
42	140	171	727.8	718.7	13.5	*0.0*	406.5	34.5	13	17	14	**151**	10
44	56	68	116.4	111.2	7.4	*0.0*	688.0	30.0	3	9	7	**15**	10
48	56	68	445.4	7.6	460.1	*0.0*	36.3	27.0	6	5	12	**66**	10
49	38	39	327.2	70.7	166.4	*0.0*	174.0	27.4	7	11	12	**20**	10
50	59	67	137.9	21.7	6.3	*0.0*	896.8	26.7	9	8	5	**39**	10

Table 6: Results of the 50 `random` test queries for SNOMED-CT, with **best performance**.

	Module size		MinAs search time (sec.)						#MinAs				
	$\mathcal{EL}^+$SAT	CEL	$\mathcal{EL}^+$SAT					CEL	$\mathcal{EL}^+$SAT				CEL
			(B)	(C)	(T)	*enc.*	(×2)		(B)	(C)	(T)	(×2)	
8	143	156	19.2	1.3	0.7	*0.0*	0.1	**20.9***	2	2	2	2	2
11	119	130	25.8	1.6	0.9	*0.0*	0.1	**21.4***	3	3	3	3	3
20	156	164	6.7	0.7	0.7	*0.0*	0.0	**21.0***	1	1	1	1	1
21	136	178	6.6	0.8	0.6	*0.0*	0.0	**21.3***	1	1	1	1	1
22	199	214	6.7	0.7	0.6	*0.0*	0.1	**21.3***	1	1	1	1	1
23	190	198	6.6	0.7	0.6	*0.0*	0.1	**21.4***	1	1	1	1	1
28	143	151	6.6	0.7	0.7	*0.0*	0.0	**21.2***	1	1	1	1	1
29	171	177	19.8	1.6	1.0	*0.1*	0.1	**21.9***	2	2	2	2	2
30	123	134	6.5	0.6	0.5	*0.0*	0.1	**20.6***	1	1	1	1	1
31	127	139	6.5	0.6	0.5	*0.0*	0.1	**20.7***	1	1	1	1	1
33	103	112	13.3	1.5	0.9	*0.0*	0.0	**21.7***	2	2	2	2	2
34	145	158	6.6	0.7	0.6	*0.0*	0.1	**21.1***	1	1	1	1	1
35	181	196	6.6	0.7	0.6	*0.0*	0.1	**21.2***	1	1	1	1	1
39	80	92	6.5	0.6	0.5	*0.0*	0.0	**20.7***	1	1	1	1	1
41	67	76	6.5	0.6	0.5	*0.0*	0.0	**20.6***	1	1	1	1	1
43	136	142	38.9	42.6	9.4	*0.0*	0.1	**24.1***	3	6	6	6	6
44	138	151	6.5	0.6	0.5	*0.0*	0.1	**20.7***	1	1	1	1	1
45	134	147	6.5	0.6	0.5	*0.0*	0.1	**20.7***	1	1	1	1	1
46	120	126	58.3	129.8	2.3	*0.0*	0.0	**22.6***	3	4	4	4	4
47	86	94	6.6	0.7	0.6	*0.0*	0.0	**20.8***	1	1	1	1	1
48	146	152	19.3	1.1	0.7	*0.0*	0.1	**21.0***	2	2	2	2	2
1	136	142	26.8	298.3	625.5	*0.0*	34.9	29.2	3	7	19	**27**	10
2	106	112	324.1	375.1	926.7	*0.0*	169.9	29.4	4	10	20	**29**	10
3	91	104	176.0	121.8	469.0	*0.0*	322.3	27.3	9	9	23	**83**	10
4	119	130	564.4	51.0	836.7	*0.0*	122.0	29.2	10	11	25	**26**	10
5	124	135	206.9	55.8	141.8	*0.0*	173.9	30.4	7	14	19	**56**	10
6	180	196	839.1	104.4	58.1	*0.0*	73.0	30.9	21	21	32	**54**	10
7	134	140	335.4	306.0	279.7	*0.0*	5.5	30.5	7	9	22	**19**	10
9	110	124	287.7	143.8	457.4	*0.0*	192.2	28.3	6	10	13	**19**	10
10	116	132	406.9	12.0	928.9	*0.0*	6.0	28.7	12	9	24	**30**	10
12	143	149	158.0	316.5	809.4	*0.0*	586.0	31.0	4	8	14	**11**	10
13	180	195	907.0	71.4	34.6	*0.0*	414.7	30.6	12	15	26	**60**	10
14	121	137	54.8	8.0	571.6	*0.0*	276.1	31.2	4	4	14	**20**	10
15	137	143	119.2	8.1	80.6	*0.0*	29.4	31.7	5	5	19	**27**	10
16	141	147	309.5	603.7	21.6	*0.0*	548.2	31.1	6	5	10	**22**	10
17	125	138	574.7	32.6	449.9	*0.0*	0.7	29.3	15	12	34	**43**	10
18	112	122	194.0	45.8	348.4	*0.0*	123.8	28.4	11	11	10	**32**	10
19	112	122	867.0	704.3	124.9	*0.0*	0.7	28.8	9	13	20	**38**	10
24	146	158	129.7	26.8	859.6	*0.0*	0.4	31.3	5	9	**21**	9	10
25	119	128	470.6	6.0	5.1	*0.0*	0.0	30.2	8	5	11	**11**	10
26	120	131	67.2	33.5	4.8	*0.0*	492.6	30.2	5	10	10	**29**	10
27	111	122	83.0	343.1	729.2	*0.0*	61.2	29.6	9	19	71	**65**	10
32	128	150	224.4	133.7	186.2	*0.0*	302.4	31.8	6	6	7	**24**	10
36	161	173	158.0	264.6	2.4	*0.0*	232.4	33.2	5	**13**	5	8	10
37	113	121	189.1	14.0	224.7	*0.0*	104.6	29.8	7	6	11	**30**	10
38	105	112	182.5	48.2	505.5	*0.0*	766.1	28.8	7	10	**18**	14	10
40	97	105	909.6	862.9	554.2	*0.0*	159.6	28.5	12	13	19	**26**	10
42	70	74	333.1	1.8	1.1	*0.0*	21.6	27.1	12	2	3	**17**	10
49	157	231	704.5	5.2	2.8	*0.0*	0.0	33.2	6	4	6	6	**10**
50	94	100	443.3	265.2	669.5	*0.0*	0.0	28.4	4	8	**14**	6	10

Table 7: Results of the 50 `selected` test queries for Snomed-CT, with **best performance**.

	Module size		MinAs search time (sec.)						#MinAs				
	$\mathcal{EL}^+$SAT	CEL	$\mathcal{EL}^+$SAT					CEL	$\mathcal{EL}^+$SAT				CEL
			(B)	(C)	(T)	*enc.*	(×2)		(B)	(C)	(T)	(×2)	
8	5	143	3.7	0.4*	0.4*	*0.0*	**0.0***	1.9*	1	1	1	1	1
15	4	135	3.7	0.3*	0.3*	*0.0*	**0.0***	1.8*	1	1	1	1	1
27	3	13	3.7	0.3*	0.3*	*0.0*	**0.0***	1.7*	1	1	1	1	1
29	3	12	3.7	0.3*	0.2*	*0.0*	**0.0***	1.6*	1	1	1	1	1
31	6	11	3.8	0.4*	0.4*	*0.0*	**0.0***	1.7*	1	1	1	1	1
32	12	26	71.4	2.0*	1.1*	*0.0*	**0.0***	2.3*	4	4	4	4	4
35	6	138	3.7	0.4*	0.4*	*0.0*	**0.0***	2.0*	1	1	1	1	1
39	2	4	3.6	0.3*	0.3*	*0.0*	**0.0***	1.6*	1	1	1	1	1
42	3	14790	3.6	0.3*	0.3*	*0.0*	**0.0***	152.3*	1	1	1	1	1
45	7	11	3.8	0.4*	0.3*	*0.0*	**0.0***	1.7*	1	1	1	1	1
5	5	30	3.8	0.4	0.3*	*0.0*	**0.0***	1.7*	1	1	1	1	1
6	11	60	3.8	0.4	0.4*	*0.0*	**0.0***	1.8*	1	1	1	1	1
17	5	56	3.6	0.3	0.3*	*0.0*	**0.0***	1.7*	1	1	1	1	1
36	5	30	3.6	0.4	0.3*	*0.0*	**0.0***	1.6*	1	1	1	1	1
49	4	14789	3.7	0.4	0.4*	*0.0*	**0.0***	195.4*	1	1	1	1	1
16	18	163	3.9	0.6	0.5	*0.0*	**0.0***	2.3*	1	1	1	1	1
20	112	14891	3.8	0.5	0.4	*0.0*	**0.0***	611.1*	1	1	1	1	1
25	23	14789	3.7	0.5	0.4	*0.0*	**0.0***	584.5*	1	1	1	1	1
26	87	14789	3.7	0.4	0.3	*0.0*	**0.0***	254.5*	1	1	1	1	1
33	45	15086	3.7	0.4	0.5	*0.0*	**0.0***	497.9*	1	1	1	1	1
48	26	111	49.0	4.1	1.5	*0.0*	**0.0***	3.5*	2	2	2	2	2
7	81	141	11.2	1.5	0.8	*0.0*	0.0	**4.1***	2	2	2	2	2
9	84	14789	3.8	0.4	0.4	*0.0*	0.0	**407.0***	1	1	1	1	1
10	60	14789	3.8	0.5	0.5	*0.0*	0.0	**818.8***	1	1	1	1	1
11	47	119	14.2	1.6	0.6	*0.0*	0.0	**2.9***	2	2	2	2	2
13	57	14835	3.6	0.3	0.3	*0.0*	0.0	**76.1***	1	1	1	1	1
18	59	14789	3.9	0.6	0.5	*0.0*	0.0	**681.1***	1	1	1	1	1
19	70	258	3.8	0.5	0.4	*0.0*	0.0	**2.8***	1	1	1	1	1
21	97	14804	4.0	0.6	0.5	*0.0*	0.0	**893.4***	1	1	1	1	1
22	38	14789	3.7	0.4	0.3	*0.0*	0.0	**477.5***	1	1	1	1	1
23	81	14789	3.8	0.5	0.4	*0.0*	0.0	**773.0***	1	1	1	1	1
24	139	14789	3.9	0.5	0.4	*0.0*	0.0	**758.3***	1	1	1	1	1
30	121	14790	3.6	0.4	0.3	*0.0*	0.0	**9.8***	1	1	1	1	1
34	72	14869	3.7	0.3	0.3	*0.0*	0.0	**79.5***	1	1	1	1	1
37	120	14789	24.7	2.6	0.9	*0.0*	0.0	**835.9***	2	2	2	2	2
38	42	85	3.9	0.5	0.3	*0.0*	0.0	**2.0***	1	1	1	1	1
46	101	14808	13.9	1.3	0.5	*0.0*	0.0	**988.3***	2	2	2	2	2
50	73	14811	10.5	1.2	0.6	*0.0*	0.0	**919.0***	2	2	2	2	2
1	57	14790	14.7	2.1	1.0	*0.0*	**0.0**	3470.0	2	2	2	2	2
2	202	14789	14.6	1.6	0.7	*0.0*	**0.1**	1271.0	2	2	2	2	2
3	122	14826	4.0	0.7	0.5	*0.0*	**0.0**	1496.0	1	1	1	1	1
4	38	14802	11.1	1.5	0.6	*0.0*	**0.0**	1865.0	2	2	2	2	2
12	196	14952	4.2	0.7	0.6	*0.0*	**0.1**	1594.0	1	1	1	1	1
14	99	14789	21.6	2.3	1.0	*0.0*	**0.0**	2073.0	2	2	2	2	2
28	40	253	113.4	54.8	718.0	*0.0*	**1.4**	20.0	7	9	10	10	10
40	126	14790	43.1	4.3	308.0	*0.0*	3.3	8162.0	3	3	8	8	**10**
41	84	14839	14.3	1.8	0.8	*0.0*	**0.0**	1560.0	2	2	2	2	2
43	69	14819	7.7	1.2	0.8	*0.0*	**0.0**	1922.0	2	2	2	2	2
44	102	14789	28.3	2.8	1.2	*0.0*	**0.0**	2900.0	2	2	2	2	2
47	71	14789	4.1	0.7	0.6	*0.0*	**0.0**	2083.0	1	1	1	1	1

Table 8: Results of the 50 `random` test queries for FULL-GALEN, with **best performance**.

	Module size		MinAs search time (sec.)						#MinAs				
	$\mathcal{EL}^+$SAT	CEL	$\mathcal{EL}^+$SAT					CEL	$\mathcal{EL}^+$SAT				CEL
			(B)	(C)	(T)	*enc.*	(×2)		(B)	(C)	(T)	(×2)	
8	147	14789	3.7	0.6	0.3	*0.0*	**0.0***	207.1*	1	1	1	1	1
9	147	14789	3.7	0.7	0.2	*0.0*	**0.0***	210.5*	1	1	1	1	1
14	135	14789	3.6	0.7	0.4	*0.0*	**0.0***	175.3*	1	1	1	1	1
1	242	14835	3.1	0.8	0.4	*0.1*	0.1	**639.3***	1	1	1	1	1
2	234	14838	3.8	0.8	0.4	*0.1*	0.1	**621.9***	1	1	1	1	1
3	230	14836	3.7	0.8	0.4	*0.0*	0.1	**545.4***	1	1	1	1	1
4	230	14835	3.8	0.8	0.4	*0.0*	0.1	**588.0***	1	1	1	1	1
5	157	14789	11.0	2.0	0.7	*0.0*	0.1	**690.6***	2	2	2	2	2
6	161	14789	3.6	0.7	0.3	*0.0*	0.1	**322.4***	1	1	1	1	1
7	146	14789	3.6	0.7	0.3	*0.0*	0.0	**321.2***	1	1	1	1	1
10	184	14789	3.7	0.7	0.4	*0.0*	0.1	**396.8***	1	1	1	1	1
11	167	14789	3.6	0.7	0.4	*0.0*	0.1	**325.4***	1	1	1	1	1
12	167	14789	3.7	0.7	0.4	*0.0*	0.0	**427.9***	1	1	1	1	1
13	199	14793	3.7	0.7	0.4	*0.1*	0.1	**297.5***	1	1	1	1	1
15	146	14789	3.6	0.6	0.4	*0.0*	0.0	**288.9***	1	1	1	1	1
16	198	14789	3.6	0.7	0.2	*0.0*	0.1	**229.8***	1	1	1	1	1
17	151	14789	3.6	0.6	0.3	*0.0*	0.0	**208.7***	1	1	1	1	1
18	200	14789	3.7	0.6	0.3	*0.1*	0.1	**204.0***	1	1	1	1	1
19	200	14789	3.6	0.7	0.3	*0.1*	0.1	**228.5***	1	1	1	1	1
20	161	14789	3.7	0.8	0.4	*0.0*	0.0	**378.9***	1	1	1	1	1
21	116	14789	3.6	0.7	0.4	*0.0*	0.0	**240.8***	1	1	1	1	1
22	206	14789	10.9	2.0	0.7	*0.1*	0.1	**801.7***	2	2	2	2	2
23	161	14789	3.6	0.7	0.4	*0.0*	0.1	**497.9***	1	1	1	1	1
24	163	14789	3.7	0.7	0.4	*0.0*	0.0	**500.5***	1	1	1	1	1
25	153	14789	3.6	0.6	0.3	*0.0*	0.0	**69.0***	1	1	1	1	1
26	197	14789	3.7	0.6	0.4	*0.0*	0.1	**300.2***	1	1	1	1	1
27	197	14789	3.7	0.6	0.3	*0.0*	0.1	**303.4***	1	1	1	1	1
28	197	14789	3.7	0.7	0.3	*0.0*	0.1	**331.8***	1	1	1	1	1
29	193	14789	3.7	0.6	0.3	*0.0*	0.1	**223.2***	1	1	1	1	1
30	185	14789	3.7	0.7	0.3	*0.0*	0.1	**223.5***	1	1	1	1	1
31	185	14789	3.6	0.5	0.2	*0.0*	0.1	**202.9***	1	1	1	1	1
32	193	14789	3.6	0.7	0.2	*0.0*	0.1	**204.2***	1	1	1	1	1
33	205	14943	3.8	0.8	0.3	*0.0*	0.1	**937.8***	1	1	1	1	1
34	180	14789	3.8	0.5	0.3	*0.0*	0.0	**929.9***	1	1	1	1	1
35	178	14789	10.8	1.7	0.5	*0.0*	0.1	**705.7***	2	2	2	2	2
36	164	14790	3.7	0.4	0.3	*0.0*	0.1	**503.2***	1	1	1	1	1
37	133	14789	3.8	0.6	0.3	*0.0*	0.0	**733.4***	1	1	1	1	1
38	173	14789	10.8	1.3	0.5	*0.0*	0.1	**803.2***	2	2	2	2	2
39	163	14789	3.7	0.4	0.3	*0.0*	0.1	**396.7***	1	1	1	1	1
40	119	14789	3.7	0.4	0.2	*0.0*	0.0	**311.0***	1	1	1	1	1
41	145	14789	3.7	0.4	0.2	*0.0*	0.0	**215.4***	1	1	1	1	1
42	158	14789	3.6	0.4	0.2	*0.0*	0.0	**212.2***	1	1	1	1	1
43	168	14789	3.6	0.4	0.2	*0.0*	0.1	**168.1***	1	1	1	1	1
44	142	14789	3.6	0.4	0.2	*0.0*	0.0	**393.9***	1	1	1	1	1
45	182	14789	3.6	0.4	0.2	*0.0*	0.1	**252.5***	1	1	1	1	1
48	204	14874	3.7	0.4	0.2	*0.0*	0.1	**158.9***	1	1	1	1	1
49	204	14874	3.7	0.4	0.2	*0.0*	0.1	**158.9***	1	1	1	1	1
50	215	14789	11.0	1.3	0.5	*0.1*	0.1	**902.5***	2	2	2	2	2
46	189	14826	3.8	0.5	0.3	*0.0*	**0.1**	1538.	1	1	1	1	1
47	181	14840	3.8	0.5	0.3	*0.0*	**0.1**	1427.	1	1	1	1	1

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

	Module size		MinAs search time (sec.)						#MinAs				
	$\mathcal{EL}^+$SAT	CEL	$\mathcal{EL}^+$SAT					CEL	$\mathcal{EL}^+$SAT				CEL
			(B)	(C)	(T)	*enc.*	(×2)		(B)	(C)	(T)	(×2)	
1	2	13	0.2	0.0*	0.0*	*0.0*	**0.0***	1.2*	1	1	1	1	1
2	3	42	0.2	0.0*	0.1*	*0.0*	**0.0***	1.3*	1	1	1	1	1
3	3	17	0.2	0.0*	0.1*	*0.0*	**0.0***	1.2*	1	1	1	1	1
4	4	4	0.2	0.0*	0.1*	*0.0*	**0.0***	1.2*	1	1	1	1	1
5	14	25	0.6	0.3*	0.2*	*0.0*	**0.0***	2.0*	3	4	4	4	4
6	4	50	0.2	0.0*	0.1*	*0.0*	**0.0***	1.3*	1	1	1	1	1
7	5	19	0.2	0.0*	0.0*	*0.0*	**0.0***	1.3*	1	1	1	1	1
8	5	91	0.2	0.0*	0.1*	*0.0*	**0.0***	1.4*	1	1	1	1	1
10	2	29	0.2	0.0*	0.1*	*0.0*	**0.0***	1.2*	1	1	1	1	1
11	9	42	0.9	0.1*	0.1*	*0.0*	**0.0***	1.6*	3	3	3	3	3
13	5	39	0.2	0.0*	0.1*	*0.0*	**0.0***	1.3*	1	1	1	1	1
14	4	158	0.2	0.0*	0.1*	*0.0*	**0.0***	1.4*	1	1	1	1	1
15	2	270	0.2	0.0*	0.1*	*0.0*	**0.0***	1.3*	1	1	1	1	1
16	2	66	0.2	0.0*	0.1*	*0.0*	**0.0***	1.3*	1	1	1	1	1
17	4	7	0.2	0.0*	0.1*	*0.0*	**0.0***	1.2*	1	1	1	1	1
18	8	39	0.6	0.1*	0.1*	*0.0*	**0.0***	1.5*	2	2	2	2	2
19	3	26	0.2	0.0*	0.1*	*0.0*	**0.0***	1.3*	1	1	1	1	1
21	10	49	1.3	0.1*	0.2*	*0.0*	**0.0***	2.0*	4	4	4	4	4
22	2	13	0.2	0.0*	0.1*	*0.0*	**0.0***	1.2*	1	1	1	1	1
23	2	35	0.2	0.0*	0.1*	*0.0*	**0.0***	1.2*	1	1	1	1	1
24	2	25	0.2	0.0*	0.0*	*0.0*	**0.0***	1.2*	1	1	1	1	1
26	3	19	0.2	0.0*	0.1*	*0.0*	**0.0***	1.2*	1	1	1	1	1
28	5	51	0.2	0.0*	0.1*	*0.0*	**0.0***	1.3*	1	1	1	1	1
29	3	5	0.2	0.0*	0.1*	*0.0*	**0.0***	1.2*	1	1	1	1	1
31	2	50	0.2	0.0*	0.1*	*0.0*	**0.0***	1.2*	1	1	1	1	1
32	4	15	0.2	0.0*	0.1*	*0.0*	**0.0***	1.2*	1	1	1	1	1
33	12	24	0.4	0.1*	0.1*	*0.0*	**0.0***	1.6*	2	2	2	2	2
34	11	67	74.2	0.2*	1.0*	*0.0*	**0.0***	1.9*	3	3	3	3	3
35	2	3	0.2	0.0*	0.1*	*0.0*	**0.0***	1.2*	1	1	1	1	1
36	6	6	0.2	0.0*	0.1*	*0.0*	**0.0***	1.2*	1	1	1	1	1
37	3	4	0.2	0.0*	0.1*	*0.0*	**0.0***	1.2*	1	1	1	1	1
38	6	25	0.6	0.0*	0.1*	*0.0*	**0.0***	1.4*	2	2	2	2	2
39	8	17	0.2	0.0*	0.1*	*0.0*	**0.0***	1.4*	1	1	1	1	1
42	6	9	0.4	0.0*	0.1*	*0.0*	**0.0***	1.3*	2	2	2	2	2
43	2	2	0.2	0.0*	0.1*	*0.0*	**0.0***	1.2*	1	1	1	1	1
44	4	11	0.2	0.0*	0.1*	*0.0*	**0.0***	1.3*	1	1	1	1	1
45	9	54	1.1	0.1*	0.1*	*0.0*	**0.0***	1.7*	3	3	3	3	3
46	7	54	0.4	0.0*	0.1*	*0.0*	**0.0***	1.5*	2	2	2	2	2
47	5	7	0.2	0.0*	0.1*	*0.0*	**0.0***	1.2*	1	1	1	1	1
49	6	31	0.4	0.0*	0.1*	*0.0*	**0.0***	1.4*	2	2	2	2	2
9	19	56	12.6	4.8	0.4	*0.0*	**0.0***	4.9*	6	7	7	7	7
20	24	34	7.8	932.9	0.6	*0.0*	**0.0***	9.6*	5	8	8	8	8
27	20	20	66.6	107.9	457.1	*0.0*	**0.4***	4.0*	7	8	8	8	8
30	21	22	78.7	459.2	897.7	*0.0*	**0.1***	3.9	7	12	12	12	10
41	26	67	624.8	400.8	360.5	*0.0*	**0.1***	4.9	10	11	11	11	10
50	22	54	54.0	865.3	114.9	*0.0*	**0.2***	9.6	9	10	10	10	10
48	31	57	14.0	1.7	375.2	*0.0*	67.7	**251.0***	9	8	9	9	9
12	32	51	229.9	558.2	21.7	*0.0*	22.8	3.7	14	21	18	**22**	10
25	38	50	67.7	0.3	373.9	*0.0*	792.	4.2	18	7	27	**33**	10
40	33	57	335.5	183.7	940.8	*0.0*	44.6	3.9	17	22	32	**36**	10

Table 10: Results of the 50 `random` test queries for NCI, with **best performance**.

	Module size		MinAs search time (sec.)						#MinAs				
	$\mathcal{EL}^+$SAT	CEL	$\mathcal{EL}^+$SAT					CEL	$\mathcal{EL}^+$SAT				CEL
			(B)	(C)	(T)	*enc.*	(×2)		(B)	(C)	(T)	(×2)	
4	15	57	3.0	0.5*	0.9*	*0.0*	**0.0***	3.1*	6	6	6	6	6
5	14	51	6.1	0.2*	0.2*	*0.0*	**0.0***	2.5*	6	6	6	6	6
6	13	15	2.4	0.3*	0.2*	*0.0*	**0.0***	1.9*	4	4	4	4	4
8	16	26	15.1	0.2*	0.3*	*0.0*	**0.0***	2.4*	5	5	5	5	5
10	12	19	15.7	0.9*	0.1*	*0.0*	**0.0***	2.0*	6	6	6	6	6
11	12	36	4.4	0.1*	0.2*	*0.0*	**0.0***	1.9*	5	5	5	5	5
12	12	36	2.1	0.2*	0.2*	*0.0*	**0.0***	1.9*	5	5	5	5	5
13	12	39	9.0	0.1*	0.1*	*0.0*	**0.0***	1.9*	4	4	4	4	4
14	14	43	17.5	0.1*	0.2*	*0.0*	**0.0***	2.4*	5	5	5	5	5
15	15	44	15.1	0.2*	0.2*	*0.0*	**0.0***	3.1*	7	7	7	7	7
16	14	22	5.0	0.1*	0.2*	*0.0*	**0.0***	1.9*	4	4	4	4	4
18	12	15	1.4	0.1*	0.1*	*0.0*	**0.0***	1.7*	4	4	4	4	4
19	12	35	1.8	0.1*	0.1*	*0.0*	**0.0***	1.8*	4	4	4	4	4
20	10	32	1.4	0.1*	0.1*	*0.0*	**0.0***	1.7*	4	4	4	4	4
21	11	57	1.4	0.1*	0.1*	*0.0*	**0.0***	1.8*	4	4	4	4	4
22	12	38	1.7	0.1*	0.2*	*0.0*	**0.0***	1.9*	4	4	4	4	4
23	11	54	1.7	0.3*	0.2*	*0.0*	**0.0***	1.8*	5	5	5	5	5
24	11	45	1.5	0.1*	0.2*	*0.0*	**0.0***	1.9*	5	5	5	5	5
25	14	45	2.3	0.2*	0.2*	*0.0*	**0.0***	2.3*	6	6	6	6	6
27	13	58	1.7	0.1*	0.1*	*0.0*	**0.0***	2.2*	5	5	5	5	5
28	15	37	2.0	0.2*	0.2*	*0.0*	**0.0***	2.4*	5	5	5	5	5
29	13	58	1.7	0.1*	0.2*	*0.0*	**0.0***	2.2*	5	5	5	5	5
30	9	60	16.2	0.1*	0.1*	*0.0*	**0.0***	1.7*	4	4	4	4	4
31	8	43	13.2	0.1*	0.1*	*0.0*	**0.0***	1.5*	4	4	4	4	4
32	16	56	36.1	0.2*	1.4*	*0.0*	**0.0***	3.2*	6	6	6	6	6
33	11	40	1.2	0.1*	0.1*	*0.0*	**0.0***	1.8*	4	4	4	4	4
34	13	21	1.1	0.1*	0.1*	*0.0*	**0.0***	1.9*	4	4	4	4	4
35	10	69	1.1	0.1*	0.1*	*0.0*	**0.0***	1.7*	4	4	4	4	4
41	17	66	15.2	9.8*	0.2*	*0.0*	**0.0***	3.5*	6	6	6	6	6
7	22	35	19.2	6.3	3.3	*0.0*	**0.0***	10.1*	7	7	7	7	7
9	23	44	90.2	116.1	9.6	*0.0*	**0.0***	15.3*	7	7	7	7	7
17	22	24	11.6	0.3	0.2	*0.0*	**0.0***	4.4*	6	6	6	6	6
49	26	38	3.2	0.3	3.9	*0.0*	**0.0***	15.3*	5	7	7	7	7
26	20	68	73.1	3.2	2.9	*0.0*	**4.6***	3.6	10	11	11	11	10
42	26	65	183.7	308.3	0.4	*0.0*	**0.1***	6.8	11	11	11	11	10
43	19	68	833.8	1.2	0.7	*0.0*	**0.0***	4.4	10	10	10	10	10
1	38	75	519.0	249.6	30.1	*0.0*	51.4	2.9	29	32	29	**33**	10
2	28	63	851.3	95.7	115.4	*0.0*	5.9	3.1	23	21	23	**23**	10
3	31	68	211.4	311.6	0.9	*0.0*	147.	2.6	16	18	18	**18**	10
36	38	91	7.6	20.3	12.6	*0.0*	148.	5.8	11	11	11	**12**	10
37	27	66	30.0	5.1	1.0	*0.0*	0.0	6.8	9	11	11	**11**	10
38	29	68	803.7	3.0	0.9	*0.0*	17.0	6.1	17	17	18	**18**	10
39	43	55	588.4	618.3	884.9	*0.0*	73.8	3.2	15	20	27	**27**	10
40	32	91	141.7	436.6	1.3	*0.0*	0.0	5.5	11	9	11	**11**	10
44	34	46	183.0	47.9	5.3	*0.0*	0.3	3.2	12	10	14	**14**	10
45	31	43	4.7	21.7	12.8	*0.0*	8.5	14.6	8	10	11	**11**	10
46	40	52	696.5	497.2	53.8	*0.0*	724.	2.9	26	19	24	**27**	10
47	36	48	282.7	607.6	136.9	*0.0*	0.0	9.1	9	9	**13**	12	10
48	40	52	942.7	501.9	53.4	*0.0*	725.	5.0	23	19	24	**27**	10
50	28	40	631.9	77.1	18.7	*0.0*	0.2	4.8	12	11	12	**12**	10

Table 11: Results of the 50 `selected` test queries for NCI, with **best performance**.

	Module size		MinAs search time (sec.)						#MinAs				
	$\mathcal{EL}^+$SAT	CEL	$\mathcal{EL}^+$SAT					CEL	$\mathcal{EL}^+$SAT				CEL
			(B)	(C)	(T)	*enc.*	(×2)		(B)	(C)	(T)	(×2)	
1	3	8	0.2	0.0*	0.0*	*0.0*	**0.0***	0.7*	1	1	1	1	1
2	3	18	0.5	0.0*	0.0*	*0.0*	**0.0***	0.8*	2	2	2	2	2
3	2	6	0.2	0.0*	0.0*	*0.0*	**0.0***	0.7*	1	1	1	1	1
4	7	35	0.7	0.1*	0.1*	*0.0*	**0.0***	1.1*	3	4	4	4	4
5	9	37	0.9	0.0*	0.1*	*0.0*	**0.0***	1.3*	5	5	5	5	5
6	5	23	3.6	0.0*	0.1*	*0.0*	**0.0***	0.9*	3	3	3	3	3
8	5	11	0.7	0.0*	0.0*	*0.0*	**0.0***	0.8*	2	2	2	2	2
9	3	19	0.2	0.0*	0.0*	*0.0*	**0.0***	0.8*	1	1	1	1	1
10	2	7	0.2	0.0*	0.0*	*0.0*	**0.0***	0.7*	1	1	1	1	1
11	11	36	71.5	0.3*	0.4*	*0.0*	**0.0***	1.6*	5	6	6	6	6
12	4	7	0.2	0.0*	0.0*	*0.0*	**0.0***	0.8*	1	1	1	1	1
13	6	19	9.6	0.1*	0.1*	*0.0*	**0.0***	0.9*	3	3	3	3	3
14	12	24	1.2	0.8*	4.2*	*0.0*	**0.0***	1.5*	6	7	7	7	7
15	4	9	0.3	0.0*	0.0*	*0.0*	**0.0***	0.8*	2	2	2	2	2
16	2	8	0.2	0.0*	0.0*	*0.0*	**0.0***	0.7*	1	1	1	1	1
17	3	16	0.2	0.0*	0.0*	*0.0*	**0.0***	0.7*	1	1	1	1	1
18	6	28	0.9	0.0*	0.0*	*0.0*	**0.0***	0.9*	2	2	2	2	2
19	9	55	207.	0.2*	0.1*	*0.0*	**0.0***	1.3*	5	5	5	5	5
20	9	42	1.1	0.1*	0.2*	*0.0*	**0.0***	1.4*	5	5	5	5	5
21	7	15	0.2	0.0*	0.0*	*0.0*	**0.0***	0.8*	1	1	1	1	1
22	3	31	0.2	0.0*	0.0*	*0.0*	**0.0***	0.8*	1	1	1	1	1
23	7	55	0.5	0.1*	0.1*	*0.0*	**0.0***	1.2*	3	4	4	4	4
24	2	7	0.2	0.0*	0.0*	*0.0*	**0.0***	0.7*	1	1	1	1	1
25	2	4	0.2	0.0*	0.0*	*0.0*	**0.0***	0.7*	1	1	1	1	1
26	4	12	0.2	0.0*	0.0*	*0.0*	**0.0***	0.8*	1	1	1	1	1
28	3	10	0.2	0.0*	0.0*	*0.0*	**0.0***	0.7*	1	1	1	1	1
29	9	26	1.1	0.2*	0.1*	*0.0*	**0.0***	1.1*	3	4	4	4	4
31	7	8	0.2	0.0*	0.0*	*0.0*	**0.0***	0.8*	1	1	1	1	1
32	10	16	1.1	0.1*	0.1*	*0.0*	**0.0***	0.9*	2	2	2	2	2
33	5	55	0.7	0.0*	0.1*	*0.0*	**0.0***	0.9*	3	3	3	3	3
34	2	20	0.2	0.0*	0.0*	*0.0*	**0.0***	0.8*	1	1	1	1	1
35	5	8	0.9	0.0*	0.1*	*0.0*	**0.0***	0.8*	2	2	2	2	2
36	3	20	0.2	0.0*	0.0*	*0.0*	**0.0***	0.7*	1	1	1	1	1
37	4	6	0.2	0.0*	0.0*	*0.0*	**0.0***	0.7*	1	1	1	1	1
38	2	4	0.2	0.0*	0.0*	*0.0*	**0.0***	0.7*	1	1	1	1	1
39	2	26	0.2	0.0*	0.0*	*0.0*	**0.0***	0.7*	1	1	1	1	1
40	6	10	0.9	0.0*	0.1*	*0.0*	**0.0***	0.8*	3	3	3	3	3
41	8	35	19.5	0.1*	0.3*	*0.0*	**0.0***	1.1*	4	4	4	4	4
42	3	27	0.2	0.0*	0.0*	*0.0*	**0.0***	0.8*	1	1	1	1	1
43	2	5	0.2	0.0*	0.0*	*0.0*	**0.0***	0.7*	1	1	1	1	1
44	4	30	0.2	0.0*	0.0*	*0.0*	**0.0***	0.8*	1	1	1	1	1
45	3	28	0.2	0.0*	0.0*	*0.0*	**0.0***	0.8*	1	1	1	1	1
46	10	17	1.3	0.2*	0.5*	*0.0*	**0.0***	1.2*	3	6	6	6	6
47	5	6	0.2	0.0*	0.0*	*0.0*	**0.0***	0.7*	1	1	1	1	1
49	8	21	190.	0.1*	0.1*	*0.0*	**0.0***	0.9*	3	3	3	3	3
50	4	13	0.5	0.0*	0.0*	*0.0*	**0.0***	0.8*	2	2	2	2	2
7	15	18	14.5	0.4*	0.6*	*0.0*	**0.0***	1.8	8	12	12	12	10
30	13	14	1.3	3.8*	0.3*	*0.0*	**0.0***	1.4	5	11	11	11	10
48	13	22	691.	1.5*	0.5*	*0.0*	**0.0***	1.6	11	11	11	11	10
27	22	27	346.	652.	265.	*0.0*	**49.1***	2.1	13	27	25	31	10

Table 12: Results of the 50 `random` test queries for GeneOntology, with **best performance**.

	Module size		MinAs search time (sec.)						#MinAs				
	$\mathcal{EL}^+$SAT	CEL	$\mathcal{EL}^+$SAT					CEL	$\mathcal{EL}^+$SAT				CEL
			(B)	(C)	(T)	*enc.*	(×2)		(B)	(C)	(T)	(×2)	
8	16	25	141.0	24.3*	3.1*	*0.0*	**0.3***	2.1*	4	9	9	9	9
27	15	23	123.2	26.8*	9.6*	*0.0*	**0.2***	1.5*	5	7	7	7	7
39	15	23	289.6	36.2*	15.8*	*0.0*	**0.1***	1.5*	4	7	7	7	7
42	13	41	677.5	2.5*	0.2*	*0.0*	**0.0***	1.8*	9	9	9	9	9
47	15	23	23.1	11.9*	3.8*	*0.0*	**0.1***	1.5*	3	7	7	7	7
49	12	28	2.3	0.2*	2.1*	*0.0*	**0.0***	1.2*	6	6	6	6	6
50	11	27	50.7	0.8*	0.3*	*0.0*	**0.0***	1.1*	6	6	6	6	6
3	20	33	161.9	118.3	94.0	*0.0*	**0.0***	3.5*	9	9	9	9	9
7	18	27	3.6	94.9*	174.4*	*0.0*	**0.8***	1.7	12	16	16	16	10
25	15	27	997.3	22.1*	6.7*	*0.0*	**0.2***	1.5	12	16	16	16	10
28	13	20	40.7	4.1*	0.6*	*0.0*	**0.0***	1.4	9	10	10	10	10
30	18	35	3.0	229.2*	156.4*	*0.0*	**0.7***	1.8	11	17	17	17	10
32	13	20	197.6	2.4*	1.9*	*0.0*	**0.0***	2.3	9	10	10	10	10
37	18	35	736.0	167.4*	59.6*	*0.0*	**0.7***	1.9	12	17	17	17	10
40	14	22	651.2	1.5*	0.3*	*0.0*	**0.1***	1.4	11	11	11	11	10
41	12	25	592.3	3.4*	0.6*	*0.0*	**0.0***	1.4	11	11	11	11	10
44	19	33	847.1	398.8*	34.1*	*0.0*	**0.6***	1.6	12	26	26	26	10
45	17	25	158.9	96.1*	54.4*	*0.0*	**0.7***	1.7	14	15	15	15	10
18	19	41	19.6	165.6	231.3*	*0.0*	**3.0***	2.1	5	16	16	16	10
1	25	45	352.4	19.3	3.4	*0.0*	**2.1***	2.8	8	11	11	11	10
4	20	37	843.4	123.1	267.1	*0.0*	**7.7***	1.6	16	20	20	20	10
5	26	37	41.0	100.6	19.9	*0.0*	**440.5***	1.9	12	15	13	20	10
6	23	40	534.0	822.7	562.3	*0.0*	**26.4***	1.7	18	21	23	24	10
9	20	37	891.2	205.1	698.5	*0.0*	**1.6***	1.7	15	19	20	20	10
10	22	39	26.2	938.2	404.2	*0.0*	**11.5***	1.7	16	22	18	23	10
11	20	37	874.7	693.9	416.1	*0.0*	**1.0***	1.6	16	20	20	20	10
12	20	37	448.3	385.7	773.7	*0.0*	**6.1***	2.7	16	20	20	20	10
13	22	31	398.7	865.3	839.0	*0.0*	**20.9***	1.8	8	28	24	30	10
16	24	41	354.8	107.7	478.6	*0.0*	**10.4***	1.9	10	10	18	21	10
17	21	44	934.1	997.0	917.3	*0.0*	**15.0***	2.0	7	15	20	20	10
21	25	37	678.2	163.2	497.7	*0.0*	**4.9***	2.1	15	16	28	28	10
22	24	35	106.4	130.2	37.4	*0.0*	**0.1***	1.7	11	12	18	18	10
24	23	32	91.4	947.4	46.5	*0.0*	**1.0***	1.9	15	17	15	18	10
26	21	30	307.2	883.0	868.4	*0.0*	**4.2***	1.8	15	15	21	21	10
29	23	37	11.5	951.9	883.3	*0.0*	**43.0***	2.8	13	25	32	32	10
31	25	45	293.8	780.3	522.6	*0.0*	**34.7***	1.9	18	35	36	38	10
33	20	32	616.2	26.7	1.2	*0.0*	**0.1***	3.3	8	14	14	14	10
34	25	45	222.6	265.2	0.2	*0.0*	**48.0***	1.7	25	25	9	38	10
35	21	33	18.2	118.7	342.9	*0.0*	**0.1***	1.9	9	16	16	16	10
36	23	37	692.6	830.5	88.2	*0.0*	**59.6***	1.6	18	29	32	32	10
38	20	32	99.2	26.1	3.7	*0.0*	**0.3***	2.5	9	14	14	14	10
43	20	32	524.6	19.9	29.7	*0.0*	**0.0***	2.6	11	14	14	14	10
46	23	37	313.4	203.6	375.9	*0.0*	**16.0***	1.7	14	32	30	32	10
48	20	29	793.8	142.5	413.6	*0.0*	**6.5***	1.8	11	20	20	20	10
2	28	45	678.2	537.2	901.8	*0.0*	992.1	2.0	15	23	24	**31**	10
14	30	40	39.7	497.9	977.3	*0.0*	220.9	1.8	20	26	23	**32**	10
15	29	38	67.2	555.7	965.0	*0.0*	51.5	1.9	19	30	37	**37**	10
19	29	38	371.4	719.3	914.3	*0.0*	41.7	1.9	11	30	34	**34**	10
20	26	37	45.3	986.1	563.3	*0.0*	768.3	1.9	9	18	18	**19**	10
23	26	37	2.6	729.5	7.3	*0.0*	4.5	1.9	7	16	14	**19**	10

Table 13: Results of the 50 `selected` test queries for GeneOntology, with **best performance**.

	Module size		MinAs search time (sec.)						#MinAs				
	$\mathcal{EL}^+$SAT	CEL	$\mathcal{EL}^+$SAT					CEL	$\mathcal{EL}^+$SAT				CEL
			(B)	(C)	(T)	*enc.*	(×2)		(B)	(C)	(T)	(×2)	
2	3	6	0.1	0.0*	0.0*	*0.0*	**0.0***	0.2*	1	1	1	1	1
6	2	58	0.1	0.0*	0.0*	*0.0*	**0.0***	0.2*	1	1	1	1	1
7	6	17	0.1	0.0*	0.0*	*0.0*	**0.0***	0.2*	1	1	1	1	1
11	9	9	0.1	0.0*	0.0*	*0.0*	**0.0***	0.2*	1	1	1	1	1
14	6	7	0.1	0.0*	0.0*	*0.0*	**0.0***	0.2*	1	1	1	1	1
16	5	9	0.1	0.0*	0.0*	*0.0*	**0.0***	0.2*	1	1	1	1	1
18	8	9	0.1	0.0*	0.0*	*0.0*	**0.0***	0.2*	1	1	1	1	1
19	8	22	0.1	0.0*	0.0*	*0.0*	**0.0***	0.3*	1	1	1	1	1
21	3	10	0.1	0.0*	0.0*	*0.0*	**0.0***	0.2*	1	1	1	1	1
24	4	9	0.1	0.0*	0.0*	*0.0*	**0.0***	0.2*	1	1	1	1	1
32	3	7	0.1	0.0*	0.0*	*0.0*	**0.0***	0.2*	1	1	1	1	1
35	4	23	0.2	0.0*	0.0*	*0.0*	**0.0***	0.3*	2	2	2	2	2
36	5	22	0.1	0.0*	0.0*	*0.0*	**0.0***	0.2*	1	1	1	1	1
40	2	6	0.1	0.0*	0.0*	*0.0*	**0.0***	0.2*	1	1	1	1	1
43	2	6	0.1	0.0*	0.0*	*0.0*	**0.0***	0.2*	1	1	1	1	1
46	3	8	0.1	0.0*	0.0*	*0.0*	**0.0***	0.2*	1	1	1	1	1
47	3	8	0.1	0.0*	0.0*	*0.0*	**0.0***	0.2*	1	1	1	1	1
1	12	153	0.6	0.0	0.0*	*0.0*	**0.0***	1.0*	2	2	2	2	2
3	3	48	0.1	0.0	0.0*	*0.0*	**0.0***	0.2*	1	1	1	1	1
5	8	22	0.1	0.0	0.0*	*0.0*	**0.0***	0.2*	1	1	1	1	1
9	12	36	0.1	0.0	0.0*	*0.0*	**0.0***	0.3*	1	1	1	1	1
10	12	34	0.1	0.0	0.0*	*0.0*	**0.0***	0.3*	1	1	1	1	1
12	7	181	0.1	0.0	0.0*	*0.0*	**0.0***	0.3*	1	1	1	1	1
20	9	239	0.1	0.0	0.0*	*0.0*	**0.0***	0.9*	1	1	1	1	1
26	14	60	0.1	0.0	0.0*	*0.0*	**0.0***	0.2*	1	1	1	1	1
29	5	297	0.1	0.0	0.0*	*0.0*	**0.0***	1.0*	1	1	1	1	1
30	10	153	0.4	0.0	0.0*	*0.0*	**0.0***	0.7*	2	2	2	2	2
31	7	165	0.1	0.0	0.0*	*0.0*	**0.0***	0.3*	1	1	1	1	1
33	9	171	0.1	0.0	0.0*	*0.0*	**0.0***	0.4*	1	1	1	1	1
34	3	69	0.1	0.0	0.0*	*0.0*	**0.0***	0.2*	1	1	1	1	1
37	5	135	0.1	0.0	0.0*	*0.0*	**0.0***	0.4*	1	1	1	1	1
38	18	134	0.1	0.0	0.0*	*0.0*	**0.0***	0.4*	1	1	1	1	1
42	14	53	0.1	0.0	0.0*	*0.0*	**0.0***	0.3*	1	1	1	1	1
49	15	69	0.1	0.0	0.0*	*0.0*	**0.0***	0.4*	1	1	1	1	1
8	25	151	0.1	0.0	0.0	*0.0*	**0.0***	0.5*	1	1	1	1	1
4	31	113	0.3	0.0	0.0	*0.0*	0.0	**0.9***	2	2	2	2	2
13	30	109	0.1	0.0	0.0	*0.0*	0.0	**0.6***	1	1	1	1	1
15	30	246	0.1	0.0	0.0	*0.0*	0.0	**1.5***	1	1	1	1	1
17	31	113	0.1	0.0	0.0	*0.0*	0.0	**0.2***	1	1	1	1	1
22	33	121	0.6	0.2	0.1	*0.0*	0.0	**1.7***	3	4	4	4	4
23	30	130	484.	0.1	0.1	*0.0*	0.0	**2.1***	5	5	5	5	5
25	41	307	0.5	0.0	0.0	*0.0*	0.0	**7.2***	2	2	2	2	2
27	31	113	0.3	0.0	0.0	*0.0*	0.0	**1.3***	2	2	2	2	2
28	32	119	0.3	0.0	0.0	*0.0*	0.0	**0.9***	2	2	2	2	2
39	32	114	0.4	0.0	0.0	*0.0*	0.0	**1.2***	2	2	2	2	2
41	31	86	0.6	0.0	0.0	*0.0*	0.0	**1.0***	2	2	2	2	2
44	31	113	0.4	0.0	0.0	*0.0*	0.0	**1.2***	2	2	2	2	2
45	78	452	0.1	0.0	0.0	*0.0*	0.0	**3.9***	1	1	1	1	1
48	31	113	0.1	0.0	0.0	*0.0*	0.0	**0.3***	1	1	1	1	1
50	34	130	0.3	0.0	0.0	*0.0*	0.0	**1.0***	2	2	2	2	2

Table 14: Results of the 50 `random` test queries for NOT-Galen, with **best performance**.

	Module size		MinAs search time (sec.)						#MinAs				
	$\mathcal{EL}^+$SAT	CEL	$\mathcal{EL}^+$SAT					CEL	$\mathcal{EL}^+$SAT				CEL
			(B)	(C)	(T)	*enc.*	(×2)		(B)	(C)	(T)	(×2)	
1	32	97	0.3	0.0	0.0	*0.0*	**0.0***	0.6*	2	2	2	2	2
2	31	96	0.3	0.0	0.0	*0.0*	**0.0***	0.8*	2	2	2	2	2
6	31	116	0.3	0.0	0.0	*0.0*	**0.0***	0.7*	2	2	2	2	2
14	31	106	0.3	0.0	0.0	*0.0*	**0.0***	0.6*	2	2	2	2	2
15	31	105	0.3	0.0	0.0	*0.0*	**0.0***	0.6*	2	2	2	2	2
16	31	116	0.3	0.0	0.0	*0.0*	**0.0***	1.0*	2	2	2	2	2
18	31	106	0.2	0.0	0.0	*0.0*	**0.0***	1.0*	2	2	2	2	2
19	31	105	0.2	0.0	0.0	*0.0*	**0.0***	0.6*	2	2	2	2	2
21	31	110	0.2	0.0	0.0	*0.0*	**0.0***	0.6*	2	2	2	2	2
22	31	113	0.2	0.0	0.0	*0.0*	**0.0***	1.1*	2	2	2	2	2
23	31	96	0.2	0.0	0.0	*0.0*	**0.0***	0.6*	2	2	2	2	2
24	31	112	0.2	0.0	0.0	*0.0*	**0.0***	0.6*	2	2	2	2	2
25	31	97	0.2	0.0	0.0	*0.0*	**0.0***	0.6*	2	2	2	2	2
26	31	112	0.2	0.0	0.0	*0.0*	**0.0***	0.6*	2	2	2	2	2
27	31	115	0.2	0.0	0.0	*0.0*	**0.0***	0.7*	2	2	2	2	2
28	31	106	0.3	0.0	0.0	*0.0*	**0.0***	0.8*	2	2	2	2	2
29	31	109	0.2	0.0	0.0	*0.0*	**0.0***	0.6*	2	2	2	2	2
30	34	86	0.1	0.0	0.0	*0.0*	**0.0***	0.3*	1	1	1	1	1
31	31	113	0.3	0.0	0.0	*0.0*	**0.0***	0.7*	2	2	2	2	2
32	31	116	0.2	0.0	0.0	*0.0*	**0.0***	0.6*	2	2	2	2	2
33	33	297	0.1	0.0	0.0	*0.0*	**0.0***	1.4*	1	1	1	1	1
34	33	180	0.1	0.0	0.0	*0.0*	**0.0***	0.6*	1	1	1	1	1
35	33	183	0.1	0.0	0.0	*0.0*	**0.0***	0.6*	1	1	1	1	1
36	33	232	0.1	0.0	0.0	*0.0*	**0.0***	1.1*	1	1	1	1	1
37	33	215	0.1	0.0	0.0	*0.0*	**0.0***	0.8*	1	1	1	1	1
38	33	277	0.1	0.0	0.0	*0.0*	**0.0***	1.2*	1	1	1	1	1
39	31	111	0.3	0.0	0.0	*0.0*	**0.0***	0.7*	2	2	2	2	2
40	31	111	0.2	0.0	0.0	*0.0*	**0.0***	0.6*	2	2	2	2	2
41	29	110	0.1	0.0	0.0	*0.0*	**0.0***	0.4*	1	1	1	1	1
49	28	239	0.1	0.0	0.0	*0.0*	**0.0***	0.9*	1	1	1	1	1
3	33	130	0.3	0.0	0.0	*0.0*	0.0	**0.7***	2	2	2	2	2
4	33	130	0.3	0.0	0.0	*0.0*	0.0	**0.7***	2	2	2	2	2
5	33	127	0.3	0.0	0.0	*0.0*	0.0	**0.7***	2	2	2	2	2
7	33	130	0.2	0.0	0.0	*0.0*	0.0	**0.7***	2	2	2	2	2
8	33	130	0.2	0.0	0.0	*0.0*	0.0	**1.1***	2	2	2	2	2
9	33	130	0.2	0.0	0.0	*0.0*	0.0	**1.0***	2	2	2	2	2
10	33	130	0.2	0.0	0.0	*0.0*	0.0	**0.7***	2	2	2	2	2
11	33	130	0.2	0.0	0.0	*0.0*	0.0	**1.2***	2	2	2	2	2
12	33	130	0.3	0.0	0.0	*0.0*	0.0	**0.7***	2	2	2	2	2
13	33	130	0.3	0.0	0.0	*0.0*	0.0	**0.7***	2	2	2	2	2
17	33	132	0.3	0.0	0.0	*0.0*	0.0	**0.6***	2	2	2	2	2
20	33	127	0.3	0.0	0.0	*0.0*	0.0	**0.7***	2	2	2	2	2
42	31	113	0.2	0.0	0.0	*0.0*	0.0	**0.6***	2	2	2	2	2
43	31	116	0.2	0.0	0.0	*0.0*	0.0	**1.0***	2	2	2	2	2
44	31	110	0.2	0.0	0.0	*0.0*	0.0	**0.7***	2	2	2	2	2
45	31	119	0.2	0.0	0.0	*0.0*	0.0	**0.6***	2	2	2	2	2
46	31	113	0.2	0.0	0.0	*0.0*	0.0	**0.6***	2	2	2	2	2
47	31	113	0.2	0.0	0.0	*0.0*	0.0	**0.6***	2	2	2	2	2
48	31	113	0.3	0.0	0.0	*0.0*	0.0	**0.6***	2	2	2	2	2
50	31	113	0.3	0.0	0.0	*0.0*	0.0	**0.7***	2	2	2	2	2

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