Journal of Artificial Intelligence Research [redacted] Submitted [redacted]

Towards Efficient SAT-Based Axiom Pinpointing in Lightweight Description Logics

M. Fareed Arif MUHAMMAD.ARIF.1@UCDCONNECT.IE
University College Dublin, Ireland

Carlos Mencía CMENCIA@GMAIL.COM
University of Oviedo, Spain

Norbert Manthey NORBERT.MANTHEY@TU-DRESDEN.DE
TU Dresden, Germany

Alexey Ignatiev AIGNATIEV@CIENCIAS.ULISBOA.PT
University of Lisbon, Portugal
ISDCT SB RAS, Irkutsk, Russia

Rafael Peñaloza RAFAEL.PELANOZA@UNIBZ.IT
Free University of Bozen-Bolzano, Italy

Joao Marques-Silva JPMS@CIENCIAS.ULISBOA.PT
University of Lisbon, Portugal

Abstract

Description Logics (DLs) are knowledge representation formalisms that have been developed to represent the terminological knowledge of an application domain through the notion of concepts. Among these, the lightweight description logics, such as the $\mathcal{EL}$ family of DLs (including $\mathcal{EL}^+$) stand out due to the availability of polynomial-time reasoning algorithms and the fact that, although inexpressive, they suffice to represent knowledge in several important application domains, such as medical informatics. Building ontologies is an error-prone task, which often results in inconsistencies and unintended inferences. Understanding the causes of (unintended) inferences, and repairing them if necessary is a central task in ontology debugging, which in turn is necessary in the development of reliable ontology-based systems. Axiom pinpointing refers to the task of explaining the causes of an (unintended) subsumption relation to follow from an ontology. More concretely, it consists in computing irreducible subsets of the ontology, called minimal axiom sets (MinAs), which entail the given subsumption relation to be explained. Over the last years, there has been a large body of research in axiom pinpointing, most of the approaches built on specific DL techniques. Notably, recent work has shown that axiom pinpoiting in the $\mathcal{EL}$ family of DLs can be addressed in the propositional domain. This earlier approach relies on encoding axiom pinpointing problems into a Horn formula, and applying propositional satisfiablity (SAT) techniques for computing the MinAs. This paper builds on this earlier work and proposes very efficient SAT-based approaches for axiom pinpointing in the $\mathcal{EL}$ family of DLs. It first reduces the problem of enumerating MinAs to the problem of enumerating the group-MUSes (minimal unsatisfiable subsets) of a Horn formula. Then, it builds on recent work on MUS enumeration and proposes two algorithms based on explicit and implicit hitting set dualization between the group-MUSes and group-MCSes (minimal correction subsets) of the Horn formula. Experimental results on well-known benchmarks derived from medical ontologies show categorical improvements over existing approaches to axiom pinpointing in the $\mathcal{EL}$ family of DLs.

Contents

1. Introduction

Description Logics (DLs) are a well-known family of logic-based knowledge representation formalisms (Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2003). DLs find a wide range of applications in computer science, including the semantic web and representation of ontologies, in particular for the life sciences. A number of reasoning tasks are associated with DLs. Among these, axiom pinpointing corresponds to the problem of computing one (or all) minimal set of axioms (called *MinA*), which explains a subsumption relation in an ontology (Schlobach & Cornet, 2003). Example applications of axiom pinpointing include context-based reasoning, error-tolerant reasoning (Ludwig & Peñaloza, 2014), and ontology debugging and revision (Schlobach, Huang, Cornet, & van Harmelen, 2007; Kalyanpur, Parsia, Sirin, & Grau, 2006), among many others. Axiom pinpointing for different description logics (DLs) has been studied extensively for more than a decade, with related work and techniques arising from the mid 90s (Baader & Hollunder, 1995; Schlobach & Cornet, 2003; Parsia, Sirin, & Kalyanpur, 2005; Meyer, Lee, Booth, & Pan, 2006; Baader, Lutz, & Suntisrivaraporn, 2006; Baader, Peñaloza, & Suntisrivaraporn, 2007; Kalyanpur, Parsia, Horridge, & Sirin, 2007; Sirin, Parsia, Grau, Kalyanpur, & Katz, 2007; Schlobach et al., 2007; Baader & Suntisrivaraporn, 2008; Sebastiani & Vescovi, 2009; Baader & Peñaloza, 2010; Moodley, Meyer, & Varzinczak, 2011; Nguyen, Alechina, & Logan, 2012; Ludwig, 2014; Sebastiani & Vescovi, 2015; Manthey & Peñaloza, 2015; Manthey, Peñaloza, & Rudolph, 2016).

The $\mathcal{EL}$ family of DLs contains lightweight languages that are well-known for having tractable reasoning problems. Despite being inexpressive, the DL $\mathcal{EL}^+$, has been successfully used for representing ontologies in the medical sciences, including the very large and extensively used SNOMED CT ontology (Spackman, Campbell, & Côté, 1997). Work on axiom pinpointing for the $\mathcal{EL}$ family of DLs can be traced back to 2007, when several methods were developed and implemented in the CEL tool (Baader et al., 2006). An alternative approach based on the application of SAT solvers was later proposed (Sebastiani & Vescovi, 2009; Vescovi, 2011; Sebastiani & Vescovi, 2015). This seminal work proposed encoding the standard reasoning methods for $\mathcal{EL}^+$ through a propositional Horn formula and apply methods from SAT to trace the causes of a consequence. This approach was implemented in the EL$^+$SAT system (Sebastiani & Vescovi, 2009; Vescovi, 2011; Sebastiani & Vescovi, 2015), which was shown to consistently outperform CEL in axiom-pinpointing tasks. In (Manthey & Peñaloza, 2015; Manthey et al., 2016), the same ideas were further improved through the inclusion of some optimization methods developed in the DL community, and others created *ad-hoc* to exploit the shape of the constructed Horn formula.

In this paper, we build on this earlier work and make several contributions towards efficient SAT-based axiom pinpointing in the $\mathcal{EL}$ family of Description Logics. First, an important relationship between axiom pinpointing and the enumeration of minimal unsatisfiable subsets (MUSes), or more precisely group-MUSes, of a propositional Horn formula is identified, which serves to capitalize on a large body of recent work on MUS enumeration in the SAT domain. Then, the paper proposes two algorithms for this task that exploit the well-known minimal hitting set duality relationship between MCSes (minimal correction subsets) and MUSes of a propositional formula.

The first method, called EL2MCS (Arif, Mencía, & Marques-Silva, 2015a), exploits hitting set dualization explicitly. More precisely, it first computes the group-MCSes of the Horn formula and then obtains the group-MUSes by computing minimal hitting sets of the set of group-MCSes. The main contribution of this paper is the development of the second method: an efficient group-MUS enumerator for Horn formulae, referred to as HgMUS (Arif, Mencía, & Marques-Silva, 2015b), that does not need to pre-compute the MCSes before-hand. Exploiting the translation from (Sebastiani & Vescovi, 2009), this method finds immediate application in axiom pinpointing in $\mathcal{EL}^+$. In the process of developing HgMUS, the paper also identifies important performance bottlenecks of existing solutions, and in particular of the previous tool EL$^+$SAT (Sebastiani & Vescovi, 2009, 2015). The new group-MUS enumerator for Horn formulae builds on the large body of recent work on problem solving with SAT oracles. This includes, among others, MUS extraction (Belov, Lynce, & Marques-Silva, 2012), MCS extraction and enumeration (Marques-Silva, Heras, Janota, Previti, & Belov, 2013a), and partial MUS enumeration (Previti & Marques-Silva, 2013; Liffiton & Malik, 2013; Liffiton, Previti, Malik, & Marques-Silva, 2016). HgMUS also exploits earlier work on solving Horn propositional formulae (Dowling & Gallier, 1984; Minoux, 1988), and develops novel algorithms for MUS extraction in propositional Horn formulae.

HgMUS was extensively tested over a large set of well-known problem instances that have been used to evaluate axiom-pinpointing methods for $\mathcal{EL}^+$ ontologies. The experimental results obtained demonstrate a conclusive improvement in the performance of the new tool over all other previously existing approaches. In most cases, this improvement could be measured by several orders of magnitude.

An outcome of this work is the new axiom pinpointing tool BEACON (Arif, Mencía, Ignatiev, Manthey, Peñaloza, & Marques-Silva, 2016), that integrates a translator from $\mathcal{EL}^+$ to Horn formulae, a group-MUS enumerator for Horn formulae HgMUS, but also FORQES (Ignatiev, Previti, Liffiton, & Marques-Silva, 2015) for computing smallest MUSes and by extension also smallest MinAs.

The rest of this paper is organized as follows. Section 2 introduces the notation and definitions used throughout the paper. Section 3 overviews past work on applying SAT for axiom pinpointing. Afterwards, Section 4 proposes a number of algorithmic improvements for SAT-based axiom pinpointing. These improvements are motivated by the analysis of EL$^+$SAT. Experimental results on well-known problem instances from axiom pinpointing for the $\mathcal{EL}$ family of DLs are analyzed in Section 5, before providing conclusions and an outlook on future work in Section 6.

2. Preliminaries

This section introduces some basic concepts and notation regarding the lightweight description logic $\mathcal{EL}^+$ and propositional satisfiability (SAT), with special emphasis on the notions used throughout the paper.

2.1 The Lightweight Description Logic $\mathcal{EL}^+$

$\mathcal{EL}^+$ (Baader, Brandt, & Lutz, 2005) is a lightweight DL that has been successfully used to build large ontologies, most notably from the bio-medical domains. As with all DLs, the main elements in $\mathcal{EL}^+$ are concepts that describe classes of individuals from the knowledge domain. Formally, $\mathcal{EL}^+$ *concepts* are built from two disjoint sets $\mathsf{N_C}$ and $\mathsf{N_R}$ of *concept names* and *role names* through the grammar rule

$$C ::= A \mid \top \mid C \sqcap C \mid \exists r.C,$$

where $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$. The concept $\top$ is referred to as the *top* concept.

The knowledge of the domain is stored in a *TBox* (or *ontology*), which is a finite set of *general concept inclusions* (GCIs) $C \sqsubseteq D$, where C and D are $\mathcal{EL}^+$ concepts, and *role inclusions* (RIs) $r_1 \circ \cdots \circ r_n \sqsubseteq s$, where $n \geq 1$ and $r_i, s \in \mathsf{N_R}$. We will often use the term *axiom* to refer to both GCIs and RIs.[1]

The semantics of this logic is based on *interpretations*, which are pairs of the form $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ where $\Delta^\mathcal{I}$ is a non-empty set called the *domain* and $\cdot^\mathcal{I}$ is the *interpretation function* that maps every concept name $A \in \mathsf{N_C}$ to a set $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ and every role name $r \in \mathsf{N_R}$ to a binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$. The interpretation function is extended to arbitrary $\mathcal{EL}^+$ concepts inductively, defining $\top^\mathcal{I} := \Delta^\mathcal{I}$, $(C \sqcap D)^\mathcal{I} := C^\mathcal{I} \cap D^\mathcal{I}$, and $(\exists r.C)^\mathcal{I} := \{\delta \in \Delta^\mathcal{I} \mid \exists \gamma \in \Delta^\mathcal{I}.(\delta, \gamma) \in r^\mathcal{I}, \gamma \in C^\mathcal{I}\}$. The interpretation $\mathcal{I}$ *satisfies* the GCI $C \sqsubseteq D$ iff $C^\mathcal{I} \subseteq D^\mathcal{I}$; it *satisfies* the RI $r_1 \circ \cdots \circ r_n \sqsubseteq s$ iff $r_1^\mathcal{I} \circ \cdots \circ r_n^\mathcal{I} \subseteq s^\mathcal{I}$, where $\circ$ denotes the usual composition of binary relations. We say that $\mathcal{I}$ is a *model* of the TBox $\mathcal{T}$ iff $\mathcal{I}$ satisfies all the GCIs and RIs in $\mathcal{T}$.

Since every $\mathcal{EL}^+$ TBox is consistent, the main reasoning problem of interest in this logic is to decide subsumption between concepts. We say that a concept C is *subsumed* by D w.r.t. the TBox $\mathcal{T}$ (denoted by $C \sqsubseteq_\mathcal{T} D$) if for every model $\mathcal{I}$ of $\mathcal{T}$ it holds that $C^\mathcal{I} \subseteq D^\mathcal{I}$. It is well known that subsumption can be restricted w.l.o.g. to subsumption between *concept names*. Hence, for the nonce we will focus on this case only. A generalization of this problem is *classification*, which refers to the task of deciding all the subsumption relations between concept names appearing in the TBox $\mathcal{T}$. In the case of $\mathcal{EL}^+$, both concept subsumption and classification can be solved in polynomial time (Baader et al., 2005).

Example 1. *Consider the TBox* $\mathcal{T}_{\text{exa}} = \{A \sqsubseteq \exists r.A, A \sqsubseteq Y, \exists r.Y \sqsubseteq B, Y \sqsubseteq B, B \sqsubseteq C\}$. *Through the transitivity of the subset relation* $\subseteq$*, it is easy to see that* $A \sqsubseteq_{\mathcal{T}_{\text{exa}}} B$ *holds. Moreover, classification of* $\mathcal{T}_{\text{exa}}$ *yields the subsumption relations*

$$\{A \sqsubseteq Y, \quad Y \sqsubseteq B, \quad B \sqsubseteq C, \quad A \sqsubseteq B, \quad A \sqsubseteq C, \quad Y \sqsubseteq C\}.$$

1. For the scope of this paper, we do not consider individual names or assertional axioms (ABoxes), which are sometimes included in the definition of an ontology (Baader et al., 2003).

Rather than merely deciding whether a subsumption relation follows from a TBox, we are interested in understanding the causes of this consequence, and repairing it if necessary.

Definition 1 (MinA, diagnosis, repair)**.** *A minimal axiom set (*MinA*) for $C \sqsubseteq D$ w.r.t. the TBox $\mathcal{T}$ is a minimal subset (w.r.t. set inclusion) $\mathcal{M} \subseteq \mathcal{T}$ such that $C \sqsubseteq_{\mathcal{M}} D$. A* diagnosis *for $C \sqsubseteq D$ w.r.t. $\mathcal{T}$ is a minimal subset (w.r.t. set inclusion) $\mathcal{D} \subseteq \mathcal{T}$ such that $C \not\sqsubseteq_{\mathcal{T} \setminus \mathcal{D}} D$. A* repair *for $C \sqsubseteq D$ w.r.t. $\mathcal{T}$ is a maximal subset (w.r.t. set inclusion) $\mathcal{R} \subseteq \mathcal{T}$ such that $C \not\sqsubseteq_{\mathcal{R}} D$.*

Informally, given a subsumption relation that follows from a TBox $\mathcal{T}$, a MinA is an irreducible subset of $\mathcal{T}$ still entailing the subsumption relation, i.e. MinAs constitute *explanations* or *justifications* for the derivation of the subsumption. A diagnosis is an irreducible set of axioms of $\mathcal{T}$ such that, if removed from $\mathcal{T}$, the resulting TBox no longer entails the subsumption relation, and a repair is a maximal subset of $\mathcal{T}$ from which the subsumption relation does not follow. In other words, diagnoses and repairs represent possible ways of removing a consequence from a TBox $\mathcal{T}$ such that only the strictly necessary axioms are dropped from $\mathcal{T}$.

Note that given a diagnosis $\mathcal{D} \subseteq \mathcal{T}$ for a given subsumption relation w.r.t. $\mathcal{T}$, its complement (i.e. $\mathcal{T} \setminus \mathcal{D}$) constitutes a repair, and *vice versa*. Besides, it is well known that the set of all MinAs can be obtained from the set of all diagnoses, and *vice versa*, through minimal hitting set computation (Reiter, 1987; Schlobach & Cornet, 2003). In the worst case, there can be an exponential number of MinAs and diagnoses for a given subsumption relation w.r.t. a TBox (Baader et al., 2007; Peñaloza, 2009).

Example 2. *Consider again the TBox $\mathcal{T}_{\text{exa}}$ from Example 1. There are two MinAs for $A \sqsubseteq B$, namely $\mathcal{M}_1 = \{A \sqsubseteq Y, Y \sqsubseteq B\}$, and $\mathcal{M}_2 = \{A \sqsubseteq \exists r.A, A \sqsubseteq Y, \exists r.Y \sqsubseteq B\}$. The diagnoses for this subsumption are $\mathcal{D}_1 = \{A \sqsubseteq Y\}$, $\mathcal{D}_2 = \{A \sqsubseteq \exists r.A, Y \sqsubseteq B\}$, and $\mathcal{D}_3 = \{\exists r.Y \sqsubseteq B, Y \sqsubseteq B\}$, which correspond to the hitting sets of $\{\mathcal{M}_1, \mathcal{M}_2\}$. Finally, the repairs for $A \sqsubseteq B$ are the complements of the diagnoses w.r.t. $\mathcal{T}_{\text{exa}}$; more precisely $\mathcal{R}_1 = \{A \sqsubseteq \exists r.A, \exists r.Y \sqsubseteq B, Y \sqsubseteq B, B \sqsubseteq C\}$, $\mathcal{R}_2 = \{A \sqsubseteq Y, \exists r.Y \sqsubseteq B, B \sqsubseteq C\}$, and $\mathcal{R}_3 = \{A \sqsubseteq \exists r.A, A \sqsubseteq Y, B \sqsubseteq C\}$.*

The central problem studied in this paper is referred to as *axiom pinpointing*, which is the task of enumerating MinAs for a given subsumption relation w.r.t. a TBox.

2.2 Propositional Satisfiability

We assume that the reader is familiar with the standard definitions and notation of propositional logic (Biere, Heule, van Maaren, & Walsh, 2009). In this paper we focus on Boolean formulae in conjunctive normal form (CNF). A CNF formula $\mathcal{F}$ is defined over a set of Boolean variables $V(\mathcal{F}) = \{x_1, ..., x_n\}$ as a conjunction of clauses $(c_1 \wedge ... \wedge c_m)$. A clause c is a disjunction of literals $(\ell_1 \vee ... \vee \ell_k)$ and a literal ℓ is either a variable x or its negation $\neg x$. We refer to the set of literals appearing in $\mathcal{F}$ as $L(\mathcal{F})$. Formulae can also be represented as sets of clauses, and clauses as sets of literals. With this representation, $|\mathcal{F}|$ refers to the number of clauses of $\mathcal{F}$ and $||\mathcal{F}||$ is the number of literals in $\mathcal{F}$.

A truth assignment, or interpretation, is a mapping $\mu : V(\mathcal{F}) \rightarrow \{0, 1\}$. If all the variables in $V(\mathcal{F})$ are assigned a truth value, μ is referred to as a *complete* assignment.

Interpretations can also be seen as conjunctions or sets of literals. Truth valuations are lifted to clauses and formulae as follows: μ satisfies a clause c if it contains at least one of its literals. Given a formula $\mathcal{F}$, μ satisfies $\mathcal{F}$ (written $\mu \vDash \mathcal{F}$) if it satisfies all its clauses, being μ referred to as a *model* of $\mathcal{F}$.

Given two formulae $\mathcal{F}$ and $\mathcal{G}$, $\mathcal{F}$ entails $\mathcal{G}$ (written $\mathcal{F} \vDash \mathcal{G}$) iff all the models of $\mathcal{F}$ are also models of $\mathcal{G}$. $\mathcal{F}$ and $\mathcal{G}$ are equivalent (written $\mathcal{F} \equiv \mathcal{G}$) iff $\mathcal{F} \vDash \mathcal{G}$ and $\mathcal{G} \vDash \mathcal{F}$. A formula $\mathcal{F}$ is satisfiable ($\mathcal{F} \nvDash \bot$) if there exists a model for it. Otherwise it is unsatisfiable ($\mathcal{F} \vDash \bot$). SAT is the decision problem of determining the satisfiability of a propositional formula. This problem is in general NP-*complete* (Cook, 1971).

Some applications require computing certain types of models. In this paper, we will make use of maximal models, i.e. models such that a set-wise maximal subset of the variables are assigned value 1.

Definition 2 (MxM)**.** *Let $\mathcal{F}$ be a satisfiable propositional formula, $\mu \vDash \mathcal{F}$ a model of $\mathcal{F}$ and $P \subseteq V(\mathcal{F})$ the set of variables appearing in μ with positive polarity. μ is a maximal model (MxM) of $\mathcal{F}$ iff $\mathcal{F} \cup P \nvDash \bot$ and for all $v \in V(\mathcal{F}) \setminus P$, $\mathcal{F} \cup P \cup \{v\} \vDash \bot$.*

In the following we will often express a maximal model simply by providing the set P of the positive literals it contains.

Example 3. *Consider the satisfiable formula $\mathcal{F} = \{(x_1 \vee x_2 \vee x_3), (\neg x_1 \vee \neg x_2), (\neg x_1 \vee \neg x_3)\}$. There are two maximal models of $\mathcal{F}$: $P_1 = \{x_1\}$ and $P_2 = \{x_2, x_3\}$.*

Horn formulae constitute an important subclass of propositional logic. These are composed of Horn clauses, which have at most one positive literal. A Horn clause is said to be a *definite* clause if it has one positive, and it is a *goal* clause otherwise. Satisfiability of Horn formulae is known to be decidable in linear time (Dowling & Gallier, 1984; Itai & Makowsky, 1987; Minoux, 1988).

Given an unsatisfiable formula $\mathcal{F}$, the following subsets represent different notions regarding (set-wise) minimal unsatisfiability and maximal satisfiability (Liffiton & Sakallah, 2008; Marques-Silva et al., 2013a):

Definition 3 (MUS, MCS, MSS)**.** *$\mathcal{M} \subseteq \mathcal{F}$ is a* Minimally Unsatisfiable Subset *(MUS) of $\mathcal{F}$ iff $\mathcal{M}$ is unsatisfiable and for all $\mathcal{M}' \subsetneq \mathcal{M}$, $\mathcal{M}'$ is satisfiable. $\mathcal{C} \subseteq \mathcal{F}$ is a* Minimal Correction Subset *(MCS) iff $\mathcal{F} \setminus \mathcal{C}$ is satisfiable and for all $\mathcal{C}' \subsetneq \mathcal{C}$, $\mathcal{F} \setminus \mathcal{C}'$ is unsatisfiable. $\mathcal{S} \subseteq \mathcal{F}$ is a* Maximal Satisfiable Subset *(MSS) iff $\mathcal{S}$ is satisfiable and for all $\mathcal{S}'$ s.t. $\mathcal{S} \subsetneq \mathcal{S}' \subseteq \mathcal{F}$, $\mathcal{S}'$ is unsatisfiable.*

In other words, an MUS is an irreducible unsatisfiable subset of $\mathcal{F}$ and, as such, it provides an *explanation* for the unsatisfiability of $\mathcal{F}$. An MCS is an irreducible set of clauses which, if removed from $\mathcal{F}$, would render the formula satisfiable. An MSS is satisfiable subset of the clauses of $\mathcal{F}$ such that if it was added any other clause from $\mathcal{F}$ it would become unsatisfiable. Thus, MCSes and MSSes represent ways of *restoring* the satisfiability of $\mathcal{F}$ so that only the strictly necessary clauses are dropped from it.

Given an MCS $\mathcal{C} \subseteq \mathcal{F}$, its complement (i.e. $\mathcal{F} \setminus \mathcal{C}$) represents an MSS (and *vice versa*). MUSes and MCSes are closely related by the well-known hitting set duality (Reiter, 1987; Bailey & Stuckey, 2005; Birnbaum & Lozinskii, 2003; Slaney, 2014): Every MCS (MUS)

is an irreducible hitting set of all MUSes (MCSes) of $\mathcal{F}$. In the worst case, there can be an exponential number of MUSes and MCSes (Liffiton & Sakallah, 2008; O'Sullivan, Papadopoulos, Faltings, & Pu, 2007).

Example 4. *Consider the formula* $\mathcal{F} = \{(x_1), (\neg x_1 \vee \neg x_2), (x_2), (x_3), (\neg x_2 \vee \neg x_3)\}$*, which is unsatisfiable. Then,*

- *its MUSes are* $\mathcal{M}_1 = \{(x_1), (\neg x_1 \vee \neg x_2), (x_2)\}$*, and* $\mathcal{M}_2 = \{(x_2), (x_3), (\neg x_2 \vee \neg x_3)\}$*;*
- $\mathcal{F}$ *has the five MCSes* $\mathcal{C}_1 = \{(x_2)\}$*,* $\mathcal{C}_2 = \{(x_1), (x_3)\}$*,* $\mathcal{C}_3 = \{(x_1), (\neg x_2 \vee \neg x_3)\}$*,* $\mathcal{C}_4 = \{(\neg x_1 \vee \neg x_2), (x_3)\}$*, and* $\mathcal{C}_5 = \{(\neg x_1 \vee \neg x_2), (\neg x_2 \vee \neg x_3)\}$*; and*
- $\mathcal{F}$ *has five MSSes* $\mathcal{S}_1 = \{(x_1), (\neg x_1 \vee \neg x_2), (x_3), (\neg x_2 \vee \neg x_3)\}$*,* $\mathcal{S}_2 = \{(\neg x_1 \vee \neg x_2), (x_2), (\neg x_2 \vee \neg x_3)\}$*,* $\mathcal{S}_3 = \{(\neg x_1 \vee \neg x_2), (x_2), (x_3)\}$*,* $\mathcal{S}_4 = \{(x_1), (x_2), (\neg x_2 \vee \neg x_3)\}$*, and* $\mathcal{S}_5 = \{(x_1), (x_2), (x_3)\}$*.*

MCSes are closely related to the MaxSAT problem, which consists in finding an assignment satisfying as many clauses as possible. The smallest MCS (largest MSS) represents an optimal solution to MaxSAT. This setting is often generalized by allowing formulae to be partitioned into sets of *hard* and *soft* clauses, i.e., $\mathcal{F} = \{\mathcal{F}_H, \mathcal{F}_S\}$. Hard clauses must be satisfied, while soft clauses can be relaxed if necessary. In this case, MCSes are required to be subsets of $\mathcal{F}_S$. Motivated by several applications, MUSes and related concepts have been extended to CNF formulae where clauses are partitioned into several disjoint sets called groups (Liffiton & Sakallah, 2008).

Definition 4 (Group-Oriented MUS)**.** *Given an explicitly partitioned unsatisfiable CNF formula* $\mathcal{F} = \mathcal{G}_0 \cup ... \cup \mathcal{G}_k$*, a* group-oriented MUS *(or group-MUS) of* $\mathcal{F}$ *is a set of groups* $\mathcal{G}_\mathcal{M} \subseteq \{\mathcal{G}_1, ..., \mathcal{G}_k\}$*, such that* $\{\mathcal{G}_0\} \cup \mathcal{G}_\mathcal{M}$ *is unsatisfiable, and for all* $\mathcal{G}'_\mathcal{M} \subsetneq \mathcal{G}_\mathcal{M}$*,* $\{\mathcal{G}_0\} \cup \mathcal{G}'_\mathcal{M}$ *is satisfiable.*

Note the special role $\mathcal{G}_0$ (*group-0*); this group consists of *background* clauses that are included in every group-MUS. Because of $\mathcal{G}_0$ a group-MUS, as opposed to a MUS, can be empty. Nevertheless, in this paper we assume that $\mathcal{G}_0$ is satisfiable. Equivalently, the related concepts of group-MCS and group-MSS can be defined in the same way.

Definition 5 (Group-Oriented MCS, MSS)**.** *Given an explicitly partitioned unsatisfiable CNF formula* $\mathcal{F} = \mathcal{G}_0 \cup ... \cup \mathcal{G}_k$*, a* group-oriented MCS *(or group-MCS) of* $\mathcal{F}$ *is a set of groups* $\mathcal{G}_\mathcal{C} \subseteq \{\mathcal{G}_1, ..., \mathcal{G}_k\}$*, such that* $\mathcal{F} \setminus \mathcal{G}_\mathcal{C}$ *is satisfiable, and for all* $\mathcal{G}'_\mathcal{C} \subsetneq \mathcal{G}_\mathcal{C}$*,* $\{\mathcal{F} \setminus \mathcal{G}'_\mathcal{C}$ *is unsatisfiable. A* group-oriented MSS *(or group-MSS) of* $\mathcal{F}$ *is a set of groups* $\mathcal{G}_\mathcal{S} \subseteq \{\mathcal{G}_1, ..., \mathcal{G}_k\}$*, such that* $\{\mathcal{G}_0\} \cup \mathcal{G}_\mathcal{S}$ *is satisfiable, and for every* $\mathcal{G}'_\mathcal{S}$ *such that* $\mathcal{G}_\mathcal{S} \subsetneq \mathcal{G}'_\mathcal{S} \subseteq \{\mathcal{G}_1, ..., \mathcal{G}_k\}$*,* $\{\mathcal{G}_0\} \cup \mathcal{G}'_\mathcal{S}$ *is unsatisfiable.*

Example 5. *Consider the unsatisfiable group formula* $\mathcal{F} = \mathcal{G}_0 \cup \mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_3$*, with the groups* $\mathcal{G}_0 = \{(\neg x_1 \vee \neg x_2), (\neg x_2 \vee \neg x_3)\}$*,* $\mathcal{G}_1 = \{(x_1)\}$*,* $\mathcal{G}_2 = \{(x_2)\}$*, and* $\mathcal{G}_3 = \{(x_3)\}$*.*

- *The group-MUSes of* $\mathcal{F}$ *are* $\mathcal{G}_{\mathcal{M}_1} = \{\mathcal{G}_1, \mathcal{G}_2\}$*, and* $\mathcal{G}_{\mathcal{M}_2} = \{\mathcal{G}_2, \mathcal{G}_3\}$*.*
- *The group-MCSes of* $\mathcal{F}$ *are* $\mathcal{G}_{\mathcal{C}_1} = \{\mathcal{G}_2\}$*, and* $\mathcal{G}_{\mathcal{C}_2} = \{\mathcal{G}_1, \mathcal{G}_3\}$*.*
- *The group-MSSes of* $\mathcal{F}$ *are* $\mathcal{G}_{\mathcal{S}_1} = \{\mathcal{G}_1, \mathcal{G}_2\}$*, and* $\mathcal{G}_{\mathcal{S}_2} = \{\mathcal{G}_2\}$*.*

In the case of MaxSAT, the use of groups is investigated in detail in (Heras, Morgado, & Marques-Silva, 2015).

Table 1: $\mathcal{EL}^+$ Completion Rules

Rules	Preconditions		Inferred axiom
R_1	$A \sqsubseteq A_i, 1 \leq i \leq n$	$A_1 \sqcap .... \sqcap A_n \sqsubseteq B$	$A \sqsubseteq B$
R_2	$A \sqsubseteq A_1$	$A_1 \sqsubseteq \exists r.B$	$A \sqsubseteq \exists r.B$
R_3	$A \sqsubseteq \exists r.B$, $B \sqsubseteq B_1$	$\exists r.B_1 \sqsubseteq B_2$	$A \sqsubseteq B_2$
R_4	$A_{i-1} \sqsubseteq \exists r_i.A_i, 1 \leq i \leq n$	$r_1 \circ ... \circ r_n \sqsubseteq r$	$A_0 \sqsubseteq \exists r.A_n$

3. SAT-Based Axiom Pinpointing

To our best knowledge, EL$^+$SAT (Sebastiani & Vescovi, 2009; Vescovi, 2011; Sebastiani & Vescovi, 2015) was the first SAT-based approach to axiom pinpointing in $\mathcal{EL}^+$ ontologies proposed in the literature. This section overviews the main ideas EL$^+$SAT is based on. Some of these ideas will be used in the remaining sections of the paper.

EL$^+$SAT consists of two phases. First, a given $\mathcal{EL}^+$ Tbox $\mathcal{T}$ is classified, i.e. all the subsumption relations between concept names in $\mathcal{T}$ are inferred, and this reasoning process is encoded into a (satisfiable) Horn formula. Each of the variables of this Horn formula corresponds to either an original axiom in $\mathcal{T}$ or a consequence derived during the classification of $\mathcal{T}$. Then, in a second step, given a subsumption relation to be explained, the Horn formula is extended and analyzed through SAT and SMT-based techniques in order to compute the MinAs of the subsumption relation w.r.t. $\mathcal{T}$.

3.1 Generation of Horn Formulae

Recall *classification* refers to the task of inferring all the subsumption relations that follow from a TBox $\mathcal{T}$. In the case of the $\mathcal{EL}$ family of DLs, classification can be done in polynomial time by first normalizing $\mathcal{T}$, and then exhaustively applying the *completion rules* shown in Table 1 (Baader et al., 2005, 2007). The main idea behind EL$^+$SAT is to encode the classification process of $\mathcal{T}$ into a Horn formula. Then, given a subsumption relation inferred in the classification of $\mathcal{T}$, this Horn formula will serve to represent the corresponding axiom pinpointing problem in the SAT domain.

3.1.1 Classification and Horn Encoding

The encoding is obtained in the following way:

1. As an initial step, each axiom $a_i \in \mathcal{T}$ is assigned a unique selector variable $s_{[a_i]}$. For trivial GCIs of the form $C \sqsubseteq C$ or $C \sqsubseteq \top$, $s_{[a_i]}$ is the constant `true`. The Horn formula $\phi^{\text{all}}_{\mathcal{T}(\text{po})}$ is initialized to the empty formula.
2. Then, $\mathcal{T}$ is normalized so that its axioms are in any of the forms

$$(A_1 \sqcap ... \sqcap A_k) \sqsubseteq B \ (k \geq 1),$$
$$A \sqsubseteq \exists r.B,$$
$$\exists r.A \sqsubseteq B, \text{ or}$$
$$r_1 \circ ... \circ r_n \sqsubseteq s \ (n \geq 1),$$

where $A, A_i, B \in \mathsf{N_C}$ and $r, r_i, s \in \mathsf{N_R}$.

This process can be performed in linear time and it results in a TBox $\mathcal{T}_{\mathcal{N}}$ where each axiom $a_i \in \mathcal{T}$ is substituted by a set of axioms in normal form $\{a_{i1}, ..., a_{im_i}\}$.

At this point, the Horn clauses $s_{[a_i]} \rightarrow s_{[a_{ik}]}$, with $1 \leq k \leq m_i$, are added to $\phi^{\text{all}}_{\mathcal{T}(\text{po})}$.

3. After normalizing the ontology, $\mathcal{T}_{\mathcal{N}}$ is saturated through the exhaustive application of the *completion rules* shown in Table 1, resulting in the extended TBox class($\mathcal{T}$). Each of the rows in Table 1 constitutes a completion rule. Their application is sound and complete for inferring subsumption relations (Baader et al., 2005). Whenever a rule r can be applied (with antecedents $\mathsf{ant}(r)$) leading to inferring an axiom a_i, the Horn clause

$$\left(\bigwedge_{a_j \in \mathsf{ant}(r)} s_{[a_j]} \right) \rightarrow s_{[a_i]}$$

is added to $\phi^{\text{all}}_{\mathcal{T}(\text{po})}$.

As a result, $\phi^{\text{all}}_{\mathcal{T}(\text{po})}$ will eventually encode all possible derivations of completion rules inferring any axiom such that $X \sqsubseteq_{\mathcal{T}} Y$, with $X, Y \in \mathsf{N_C}$.

Example 6. *Consider again the ontology $\mathcal{T}_{\text{exa}}$ from Example 1, which is already in normal form. On the left, the example shows the consequences derived by the classification of $\mathcal{T}_{\text{exa}}$. On the right, the example shows the Horn encoding of the classification of $\mathcal{T}_{\text{exa}}$, indicating the completion rule that was applied in each case.*

$$\begin{aligned}
\mathit{class}(\mathcal{T}_{\text{exa}}) = \{a_1 :\ & A \sqsubseteq \exists r.A \\
a_2 :\ & A \sqsubseteq Y, \\
a_3 :\ & \exists r.Y \sqsubseteq B, \\
a_4 :\ & Y \sqsubseteq B, \\
a_5 :\ & B \sqsubseteq C, \\
a_6 :\ & A \sqsubseteq B, \\
a_7 :\ & Y \sqsubseteq C, \\
a_8 :\ & A \sqsubseteq C\}
\end{aligned}
\qquad
\begin{aligned}
\phi^{all}_{\mathcal{T}_{\text{exa}}(po)} = \{ & s_{[a_1]} \wedge s_{[a_2]} \wedge s_{[a_3]} \rightarrow s_{[a_6]}, && (R_3) \\
& s_{[a_2]} \wedge s_{[a_4]} \rightarrow s_{[a_6]}, && (R_1) \\
& s_{[a_4]} \wedge s_{[a_5]} \rightarrow s_{[a_7]}, && (R_1) \\
& s_{[a_5]} \wedge s_{[a_6]} \rightarrow s_{[a_8]}, && (R_1) \\
& s_{[a_2]} \wedge s_{[a_7]} \rightarrow s_{[a_8]}\} && (R_1)
\end{aligned}$$

The Horn encoding of the classification of $\mathcal{T}$ only needs to be done once. Using this formula, one can then find all the MinAs for any subsumption relation that can be derived from $\mathcal{T}$. Thus, the cost of constructing $\phi^{all}_{\mathcal{T}(po)}$ can be amortized if several consequences of the TBox are explained and corrected.

3.1.2 Axiom Pinpointing Instances

For computing MinAs of a given subsumption relation $C \sqsubseteq_{\mathcal{T}} D$, EL$^+$SAT builds on the following fundamental result.

Theorem 1 (Theorem 3 in (Sebastiani & Vescovi, 2015))**.** *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$, for every $\mathcal{S} \subseteq \mathcal{T}$ and for every pair of concept names $C, D \in \mathsf{N_C}$, $C \sqsubseteq_{\mathcal{S}} D$ if and only if the Horn propositional formula $\phi^{\text{all}}_{\mathcal{T}(\text{po})} \wedge (\neg s_{[C \sqsubseteq D]}) \wedge \bigwedge_{ax_i \in \mathcal{S}} (s_{[ax_i]})$ is unsatisfiable.*

This theorem states that, given the Horn encoding of the classification of $\mathcal{T}$ $\phi^{\text{all}}_{\mathcal{T}(\text{po})}$, the set of Boolean variables $\{s_{[ax_i]} \,|\, ax_i \in \mathcal{T}\}$ representing the original axioms of $\mathcal{T}$, and the Boolean variable $s_{[C \sqsubseteq D]}$ associated with the subsumption relation to be explained, the MinAs of $\mathcal{T}$ w.r.t. $C \sqsubseteq D$ correspond to subsets of $\{s_{[ax_i]} \,|\, ax_i \in \mathcal{T}\}$ whose activation (setting these variables to 1) makes $\phi^{\text{all}}_{\mathcal{T}(\text{po})} \wedge (\neg s_{[C \sqsubseteq D]})$ unsatisfiable. More concretely, from the enumeration mechanism of EL$^+$SAT, MinAs will correspond exactly to minimal subsets of $\{s_{[ax_i]} \,|\, ax_i \in \mathcal{T}\}$. Noticeably, (Sebastiani & Vescovi, 2015) introduces effective techniques for simplifying the formulas that encode concrete MinA computation instances, such as the *Cone-Of-Influence* (C.O.I.) modularization technique or the so-called x2 optimization. These techniques allow for reducing to a great extent the size of the Horn formula and the number of assumption variables in the original ontology that take place in the enumeration step, and they play a very important role in the efficiency of EL$^+$SAT. This techniques can also be implemented in polynomial time. For further information we refer the reader to (Vescovi, 2011; Sebastiani & Vescovi, 2015).

3.2 EL$^+$SAT Computation of MinAs

As mentioned earlier, the second component of EL$^+$SAT computes the MinAs of the subsumption relation by analyzing the causes of unsatisfiability of an extended formula. The focus on this section is to analyze the MinA enumerator used in EL$^+$SAT, highlighting a number of issues in its design. Figure 1 provides a high-level perspective of the MinA enumerator EL$^+$SAT, which is based on an AllSAT/AllSMT approach (Lahiri, Nieuwenhuis, & Oliveras, 2006).

In this approach, branching is executed in such a manner that variables are always assigned value 1. This guarantees that maximal models are computed. The downside is that the branching heuristic of the SAT solver is affected. The AllSAT approach searches for an assignment that satisfies the Horn formula without taking into account the unit negative clause associated with the target subsumption relation. Every time a solution to the Horn formula is found, EL$^+$SAT enters a dedicated mode, where CDCL-based unit propagation is used to identify a conflict and, if a conflict is found, a MinA is computed and reported. The computation of a MinA uses a standard deletion-based algorithm (Belov et al., 2012). Blocking is based on the branching variables, and so it is solely composed of negative literals, i.e. it is also a Horn formula. This is true independently of having computed a MinA or not. The process is then repeated while solutions to the Horn formula exist.

The description above summarizes the key steps of the MinA enumerator used in EL$^+$SAT, but also highlights some of the performance issues that can exist with EL$^+$SAT. First, forcing branching to always assign variables to 1 affects the branching heuristic of the SAT solver. This can have an important impact on the performance of the tool. Second, unit propagation in a CDCL-based SAT solver is quadratic in the worst-case (Gent, 2013), while Horn formula satisfiability is decidable in linear time. The theoretical solution is to use different implementation of how the watched pointers are handled, but this is seldom used in practice. Third, as shown in Section 4 for the concrete setting of Horn formulae, there are more efficient solutions than using deletion-based algorithms for computing MinAs. Finally, and most importantly, the fact that non-MinAs are blocked in the same way

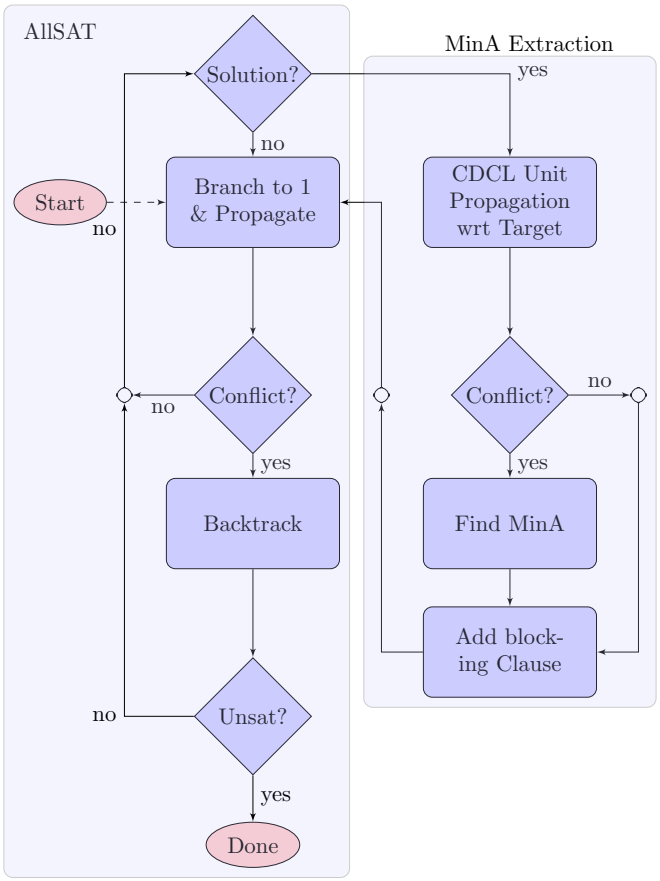


Figure 1: High-level perspective of EL$^+$SAT MinA enumeration

as MinAs, can result in an exponentially larger number of iterations when compared with distinguished blocking for MinAs and non-MinAs (Arif et al., 2015b).

4. Towards Efficient SAT-Based Axiom Pinpointing

This section builds on the ideas introduced by EL$^+$SAT and develops them further. First, it shows that axiom pinpointing in $\mathcal{EL}^+$ ontologies can be reduced to the problem of enumerating (group) MUSes of a Horn formula. Then, it describes two approaches for this problem, which are based on explicit and implicit hitting set dualization between the (group) MUSes and (group) MCSes of unsatisfiable formulae, giving rise to the state-of-the-art algorithms EL2MCS (Arif et al., 2015a) and HgMUS (Arif et al., 2015b) respectively.

4.1 Axiom Pinpointing as Group-MUS Enumeration

Although not explicitly stated, the relationship between axiom pinpointing and MUS extraction has been apparent in earlier work (Baader et al., 2007; Sebastiani & Vescovi, 2009, 2015). Building on Theorem 1, the computation of the MinAs for a subsumption relation

$C \sqsubseteq_{\mathcal{T}} D$ can be reduced to the problem of enumerating group-MUSes of the group Horn formula $\mathcal{H}_{[C \sqsubseteq_{\mathcal{T}} D]} = \{\mathcal{G}_0, \mathcal{G}_1, ..., \mathcal{G}_{|\mathcal{T}|}\}$, where

- $\mathcal{G}_0 = \phi^{\text{all}}_{\mathcal{T}(\text{po})} \cup \{(\neg s_{[C \sqsubseteq D]})\}$; and
- each axiom $ax_i \in \mathcal{T}, 1 \leq i \leq |\mathcal{T}|$ defines a group $\mathcal{G}_i = \{(s_{[a_i]})\}$ with a single unit clause.

Then, we obtain the following result.

Theorem 2. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$, a set $\mathcal{M} \subseteq \mathcal{T}$, and $C, D \in \mathsf{N_C}$, $\mathcal{M}$ is a MinA for $C \sqsubseteq_{\mathcal{T}} D$ if and only if the set $\mathcal{G}_{\mathcal{M}} = \bigcup_{\{i | ax_i \in \mathcal{M}\}} \mathcal{G}_i$ is a group-MUS of $\mathcal{H}_{[C \sqsubseteq_{\mathcal{T}} D]}$.*

Proof. (**If**) By definition, since $\mathcal{G}_{\mathcal{M}}$ is a group-MUS of $\mathcal{H}_{[C \sqsubseteq_{\mathcal{T}} D]}$, it holds that the formula $\phi^{\text{all}}_{\mathcal{T}(\text{po})} \wedge \bigwedge_{ax_i \in \mathcal{M}} (s_{[ax_i]}) \wedge (\neg s_{[C \sqsubseteq D]})$ is unsatisfiable, and for all $\mathcal{M}' \subsetneq \mathcal{M}$ the formula $\phi^{\text{all}}_{\mathcal{T}(\text{po})} \wedge \bigwedge_{ax_i \in \mathcal{M}'} (s_{[ax_i]}) \wedge (\neg s_{[C \sqsubseteq D]})$ is satisfiable. By Theorem 1, it follows that $C \sqsubseteq_{\mathcal{M}} D$ and for all $\mathcal{M}' \subsetneq \mathcal{M}$, $C \not\sqsubseteq_{\mathcal{M}'} D$. Thus, $\mathcal{M}$ is a MinA for $C \sqsubseteq_{\mathcal{T}} D$.
(**Only if**) Since $\mathcal{M}$ is a MinA for $C \sqsubseteq_{\mathcal{T}} D$, we have that $C \sqsubseteq_{\mathcal{M}} D$ and for all $\mathcal{M}' \subsetneq \mathcal{M}$, $C \not\sqsubseteq_{\mathcal{M}'} D$. By Theorem 1, the formula $\phi^{\text{all}}_{\mathcal{T}(\text{po})} \wedge \bigwedge_{ax_i \in \mathcal{M}} (s_{[ax_i]}) \wedge (\neg s_{[C \sqsubseteq D]})$ is unsatisfiable (and hence also is $\{\mathcal{G}_0\} \cup \mathcal{G}_{\mathcal{M}}$ in $\mathcal{H}_{[C \sqsubseteq_{\mathcal{T}} D]}$) and for all $\mathcal{M}' \subsetneq \mathcal{M}$, the formula $\phi^{\text{all}}_{\mathcal{T}(\text{po})} \wedge \bigwedge_{ax_i \in \mathcal{M}'} (s_{[ax_i]}) \wedge (\neg s_{[C \sqsubseteq D]})$ is satisfiable (and so is $\mathcal{G}_0 \cup \mathcal{G}_{\mathcal{M}'}$ in $\mathcal{H}_{[C \sqsubseteq_{\mathcal{T}} D]}$). Thus, $\mathcal{G}_{\mathcal{M}}$ is a group-MUS of $\mathcal{H}_{[C \sqsubseteq_{\mathcal{T}} D]}$. □

The minimal hitting set duality relationship between MinAs and diagnoses for a given subsumption relation w.r.t. a TBox, which as seen holds also between group-MUSes and group-MCSes of unsatisfiable group formulae, allows us to map diagnoses to group-MCSes. At the same time, since each repair is the complement of a diagnosis, and each group-MSS is the complement of a group-MCS, repairs can be related with group-MSSes.

Corollary 1. *Let $\mathcal{T}$ be an $\mathcal{EL}^+$ TBox, and C, D be two concept names. Then:*

- *$\mathcal{D} \subseteq \mathcal{T}$ is a diagnosis for $C \sqsubseteq_{\mathcal{T}} D$ if and only if $\mathcal{G}_{\mathcal{D}} = \bigcup_{\{i | ax_i \in \mathcal{D}\}} \mathcal{G}_i$ is a group-MCS of $\mathcal{H}_{[C \sqsubseteq_{\mathcal{T}} D]}$, and*
- *$\mathcal{R} \subseteq \mathcal{T}$ is a repair for $C \sqsubseteq_{\mathcal{T}} D$ if and only if $\mathcal{G}_{\mathcal{R}} = \bigcup_{\{i | ax_i \in \mathcal{R}\}} \mathcal{G}_i$ is a group-MSS of $\mathcal{H}_{[C \sqsubseteq_{\mathcal{T}} D]}$.*

The following example illustrates these results.

Example 7. *Let us consider the $\mathcal{EL}^+$ ontology $\mathcal{T}_{\text{exa}}$ from Example 1. The classification of $\mathcal{T}_{\text{exa}}$ and the definition of $\phi^{all}_{\mathcal{T}_{\text{exa}}(po)}$ are shown in Example 6. As we can observe, the axiom $a_6 : A \sqsubseteq B$ belongs to the classification of $\mathcal{T}_{\text{exa}}$. For this subsumption relation, we define $\mathcal{H}_{[A \sqsubseteq_{\mathcal{T}_{\text{exa}}} B]} = \{\mathcal{G}_0, \mathcal{G}_1, ..., \mathcal{G}_5\}$, where $\mathcal{G}_0 = \phi^{all}_{\mathcal{T}_{\text{exa}}(po)} \cup \{(\neg s_{[a_6]})\}$, and for $i = 1, ..., 5$, $\mathcal{G}_i = \{(s_{[a_i]})\}$, i.e.*

$$\begin{aligned}
\mathcal{G}_0 = \{&(\neg s_{[a_1]} \vee \neg s_{[a_2]} \vee \neg s_{[a_3]} \vee s_{[a_6]}),\\
&(\neg s_{[a_2]} \vee \neg s_{[a_4]} \vee s_{[a_6]}),\\
&(\neg s_{[a_4]} \vee \neg s_{[a_5]} \vee s_{[a_7]}),\\
&(\neg s_{[a_5]} \vee \neg s_{[a_6]} \vee s_{[a_8]}),\\
&(\neg s_{[a_2]} \vee \neg s_{[a_7]} \vee s_{[a_8]}),\\
&(\neg s_{[a_6]})\}
\end{aligned}
\qquad
\begin{aligned}
\mathcal{G}_1 &= \{(s_{[a_1]})\}\\
\mathcal{G}_2 &= \{(s_{[a_2]})\}\\
\mathcal{G}_3 &= \{(s_{[a_3]})\}\\
\mathcal{G}_4 &= \{(s_{[a_4]})\}\\
\mathcal{G}_5 &= \{(s_{[a_5]})\}
\end{aligned}$$

$\mathcal{H}_{[A \sqsubseteq B]}$ has two group-MUSes: $\mathcal{G}_{\mathcal{M}_1} = \{\mathcal{G}_2, \mathcal{G}_4\}$ and $\mathcal{G}_{\mathcal{M}_2} = \{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$ corresponding to the MinAs $\mathcal{M}_1 = \{A \sqsubseteq Y, Y \sqsubseteq B\}$, and $\mathcal{M}_2 = \{A \sqsubseteq \exists r.A, A \sqsubseteq Y, \exists r.Y \sqsubseteq B\}$ respectively. The group-MCSes of $\mathcal{H}_{[A \sqsubseteq B]}$ are $\mathcal{G}_{\mathcal{C}_1} = \{\mathcal{G}_2\}$, $\mathcal{G}_{\mathcal{C}_2} = \{\mathcal{G}_1, \mathcal{G}_4\}$, and $\mathcal{G}_{\mathcal{C}_3} = \{\mathcal{G}_3, \mathcal{G}_4\}$, corresponding to the diagnoses $\mathcal{D}_1 = \{A \sqsubseteq Y\}$, $\mathcal{D}_2 = \{A \sqsubseteq \exists r.A, Y \sqsubseteq B\}$, and $\mathcal{D}_3 = \{\exists r.Y \sqsubseteq B, Y \sqsubseteq B\}$ respectively.

It should be observed that the difference between the enumeration of plain MUSes of Horn formulae and the enumeration of group-MUSes is significant in terms of complexity. First, enumeration of group-MUSes of Horn formulae cannot be achieved in total polynomial time, unless P = NP. This is an immediate consequence from the fact that axiom pinpointing for the $\mathcal{EL}$ family of DLs cannot be achieved in total polynomial time, unless P = NP (Baader et al., 2007), and that axiom pinpointing for the $\mathcal{EL}$ family of DLs can be reduced in polynomial time to group-MUS enumeration of Horn formulae. Second, the MUSes of Horn formulae can be enumerated with only a polynomial delay between answers, which in particular implies a total polynomial time (Peñaloza & Sertkaya, 2010).

The following two subsections develop two approaches for enumerating the group-MUSes of Horn formulae. The first one, EL2MCS (Arif et al., 2015a) is based on explicit hitting set dualization between the MUSes and the MCSes of the Horn formula. The second approach, HgMUS (Arif et al., 2015b) follows an implicit hitting set dualization instead.

4.2 Group-MUS Enumeration by Explicit Hitting Set Dualization

Enumeration of MUSes has been the subject of research that can be traced to the seminal work of Reiter (Reiter, 1987). A well-known family of algorithms uses (explicit) minimal hitting set dualization (Birnbaum & Lozinskii, 2003; Bailey & Stuckey, 2005; Liffiton & Sakallah, 2008). The organization of these algorithms can be summarized as follows: First compute all the MCSes of a CNF formula. Second, MUSes are obtained by computing the minimal hitting sets of the set of MCSes.

EL2MCS (Arif et al., 2015a) is an approach based on this explicit hitting set dualization. This method relies on maximum satisfiability (MaxSAT): In the first step, it enumerates the group-MCSes of the Horn formula following methods developed for MaxSAT. Then, in the second step, it computes the minimal hitting sets (i.e. the group-MUSes) using an existing algorithm (Liffiton & Sakallah, 2008).

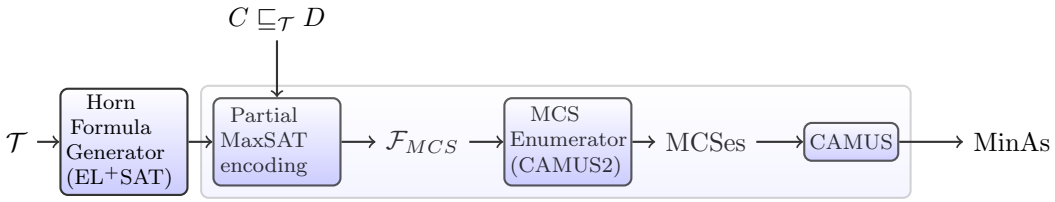


Figure 2: The EL2MCS tool

4.2.1 Axiom Pinpointing Using MaxSAT

The approach followed by EL2MCS is to model the problem as partial maximum satisfiability (MaxSAT), and enumerate over the MUSes of the MaxSAT problem formulation. As mentioned before, the first phase of EL2MCS consists in enumerating the MCSes of a MaxSAT problemm formulation, as follows. A formula $\mathcal{F}_{MCS}$ is defined as follows: all the clauses in $\mathcal{G}_0$ of the Horn formula $\mathcal{H}_{[C \sqsubseteq_{\mathcal{T}} D]}$ defined above are declared as *hard clauses*, i.e. both $\phi^{\text{all}}_{\mathcal{T}(\text{po})}$ and $(\neg s_{[C \sqsubseteq D]})$ are encoded as hard clauses. In addition, the variable $s_{[ax_i]}$ associated with each axiom $ax_i \in \mathcal{T}$ (corresponding to each group $\mathcal{G}_i$, with $i > 0$, in $\mathcal{H}_{[C \sqsubseteq_{\mathcal{T}} D]}$) denotes a *unit soft clause* $(s_{[ax_i]})$. The intuitive justification for this construction is that the goal is to include as many axioms as possible, leaving out a minimal set which, if included, would cause the complete formula to be unsatisfiable. Thus, each of these sets represents an MCS of the MaxSAT problem formulation (and so a group-MCS of $\mathcal{H}_{[C \sqsubseteq_{\mathcal{T}} D]}$), but also a minimal set of axioms that needs to be dropped for the subsumption relation not to hold (i.e. a *diagnosis*). MCS enumeration can be implemented with a MaxSAT solver (Liffiton & Sakallah, 2008; Morgado, Liffiton, & Marques-Silva, 2012) or with a dedicated algorithm (Marques-Silva et al., 2013a). Extensive experimental results have shown that the first option yields better performance, and hence it is the adopted approach.

In a second phase, in order to get the MUSes of the MaxSAT problem formulation (and so the group-MUSes of $\mathcal{H}_{[C \sqsubseteq_{\mathcal{T}} D]}$), EL2MCS enumerates the minimal hitting sets of the set of all the MCSes computed in the first phase. This is achieved by exploiting the hitting set dualization algorithm used in CAMUS (Liffiton & Sakallah, 2008). (The use of SAT-based approaches turns out not to be efficient when compared to CAMUS and related algorithms (Liffiton & Sakallah, 2008).)

Example 8. *Let us consider the group Horn formula $\mathcal{H}_{[A \sqsubseteq_{\mathcal{T}_{\text{exa}}} B]}$ in Example 7.*

In the first phase, the MaxSAT formula $\mathcal{F}_{MCS}$ is built, with the hard clauses $\phi^{all}_{\mathcal{T}_{\text{exa}}(po)} \cup \{(\neg s_{a_6})\}$, and the soft clauses $\{(s_{[a_1]}), (s_{[a_2]}), (s_{[a_3]}), (s_{[a_4]}), (s_{[a_5]})\}$. The MCSes of $\mathcal{F}_{MCS}$ are $\mathcal{C}_1 = \{(s_{[a_2]})\}$, $\mathcal{C}_2 = \{(s_{[a_1]}), (s_{[a_4]})\}$, and $\mathcal{C}_3 = \{(s_{[a_3]}), (s_{[a_4]})\}$. As shown in Example 7, these sets correspond to the group-MCSes of $\mathcal{H}_{[A \sqsubseteq_{\mathcal{T}_{\text{exa}}} B]}$ (diagnoses for $A \sqsubseteq_{\mathcal{T}_{\text{exa}}} B$).

In the second phase, CAMUS is used to compute the minimal hitting sets of the MCSes, obtaining $\mathcal{M}_1 = \{(s_{[a_2]}), (s_{[a_4]})\}$ and $\mathcal{M}_2 = \{(s_{[a_1]}), (s_{[a_2]}), (s_{[a_3]})\}$. As shown in Example 7, these sets correspond to the group-MUSes of $\mathcal{H}_{[A \sqsubseteq_{\mathcal{T}_{\text{exa}}} B]}$ (MinAs for $A \sqsubseteq_{\mathcal{T}_{\text{exa}}} B$).

4.2.2 EL2MCS Tool

The organization of the EL2MCS tool is depicted in Figure 2. The first step is similar to EL$^+$SAT (Sebastiani & Vescovi, 2009, 2015) in that the propositional Horn formula $\phi^{\text{all}}_{\mathcal{T}(\text{po})}$ is generated. The next step, however, produces a partial MaxSAT encoding. From this

Algorithm 1: eMUS (Previti & Marques-Silva, 2013) / MARCO (Liffiton et al., 2016)

Input: $\mathcal{F}$ a CNF formula

```
1  I ← {p_i | c_i ∈ F}                                  // Variable p_i picks clause c_i
2  Q ← ∅
3  while true do
4      (st, P) ← MaximalModel(Q)
5      if not st then return
6      F' ← {c_i | p_i ∈ P}                              // Pick selected clauses
7      if not SAT(F') then
8          M ← ComputeMUS(F')
9          ReportMUS(M)
10         b ← {¬p_i | c_i ∈ M}                          // Negative clause blocking the MUS
11     else
12         b ← {p_i | p_i ∈ I \ P}                       // Positive clause blocking the MCS
13     Q ← Q ∪ {b}
```

encoding it becomes possible to enumerate the MCSes of the partial MaxSAT formula using an off-the-shelf tool like CAMUS2 (Marques-Silva et al., 2013a), which is a modern implementation of the of MCS enumerator available in CAMUS (Liffiton & Sakallah, 2008) capable of solving MaxSAT problems. The final step is to exploit minimal hitting set duality for computing all the MUSes given the set of MCSes (Liffiton & Sakallah, 2008). This is achieved with the help of the original CAMUS tool. The hypergraph traversal computation tools, shd (Murakami & Uno, 2011) and MTminer (Hëbert, Bretto, & Crëmilleux, 2007), could be used instead for this latter phase. It should be observed that, although the MCS enumeration uses CAMUS2, other alternative MCS enumeration approaches were considered during development (Marques-Silva et al., 2013a). The choice of CAMUS2 arose from the better performance that it demostrated.

4.3 Group-MUS Enumeration by Implicit Hitting Set Dualization

The main drawback of using explicit minimal hitting set dualization for finding group-MUSes is that it must first compute all group-MCSes, before the search for the first group-MUS starts. Thus, if the number of MCSes is large, this approach will take a long time find any MUS, and even longer to find them all, even if the total number of MUSes is small. As a result, recent work considered what can be described as implicit minimal hitting set dualization (Liffiton & Malik, 2013; Previti & Marques-Silva, 2013; Liffiton et al., 2016). In these approaches (implemented in systems like eMUS (Previti & Marques-Silva, 2013) and MARCO (Liffiton et al., 2016)), either an MUS or an MCS is computed at each step of the algorithm, with the guarantee that at least one MUS will be obtained at the outset. In some settings, implicit minimal hitting set dualization is the only solution for finding some MUSes of a CNF formula. As pointed out in this recent work, implicit minimal hitting set dualization aims to complement, but not replace, the explicit dualization alternative, and in some settings where enumeration of MCSes is feasible, the latter may be the preferred option (Previti & Marques-Silva, 2013; Liffiton et al., 2016).

Algorithm 1 shows the eMUS enumeration algorithm (Previti & Marques-Silva, 2013), also used in the most recent version of MARCO (Liffiton et al., 2016). It relies on a two-solver approach aimed at enumerating the MUSes/MCSes of an unsatisfiable formula $\mathcal{F}$. On the one hand, a formula $\mathcal{Q}$ is used to enumerate subsets of $\mathcal{F}$. This formula is defined over a set of variables $I = \{p_i \mid c_i \in \mathcal{F}\}$, each one of them associated with one clause $c_i \in \mathcal{F}$. Iteratively until $\mathcal{Q}$ becomes unsatisfiable, eMUS computes a maximal model P of $\mathcal{Q}$ and tests the satisfiability of the corresponding subformula $\mathcal{F}' \subseteq \mathcal{F}$. If it is satisfiable, $\mathcal{F}'$ represents an MSS of $\mathcal{F}$, and the clause $I \setminus P$ is added to $\mathcal{Q}$, preventing the algorithm from generating any subset of the MSS (superset of the MCS) again. Otherwise, if $\mathcal{F}'$ is unsatisfiable, it is reduced to an MUS $\mathcal{M}$, which is blocked adding to $\mathcal{Q}$ a clause made of the variables in I associated with $\mathcal{M}$ with negative polarity. This way, no superset of $\mathcal{M}$ will be generated. Algorithm 1 is guaranteed to find all MUSes and MCSes of $\mathcal{F}$, in a number of iterations that corresponds to the sum of the number of MUSes and MCSes.

HgMUS is a novel and efficient group-MUS enumerator for Horn formulae based on implicit minimal hitting set dualization, which incorporates specific features for the problem formulation. In this context, $\mathcal{H}$ denotes the group of clauses $\mathcal{G}_0$, i.e. the background clauses. Moreover, $\mathcal{I}$ denotes the set of (individual) groups of clauses, with $\mathcal{I} = \{\mathcal{G}_1, \ldots, \mathcal{G}_k\}$. So, the unsatisfiable group-Horn formula corresponds to $\mathcal{F} = \mathcal{H} \cup \mathcal{I}$. Also, the formula $\mathcal{Q}$ shown in Algorithm 1 is defined on a set of variables associated to the groups in $\mathcal{I}$. For the problem instances obtained from axiom pinpointing for the $\mathcal{EL}$ family of DLs, ($\mathcal{H}_{[C \sqsubseteq_{\mathcal{T}} D]}$) each group of clauses contains a single unit clause. However, the algorithm works for arbitrary groups of clauses.

4.3.1 Organization of HgMUS

The high-level organization of HgMUS mimics that of eMUS/MARCO (see Algorithm 1), with a few essential differences. First, the satisfiability testing step (because it operates on Horn formulae) uses the dedicated linear time algorithm LTUR (Minoux, 1988). Moreover, the simplicity of LTUR enables very efficient implementations, that use adjacency lists for representing clauses instead of the now more commonly used watched literals. Second, the problem formulation motivates using a dedicated MUS extraction algorithm, which is shown to be more effective in this concrete case than other well-known approaches (Belov et al., 2012). Third, we also highlight important aspects of the eMUS/MARCO implicit minimal hitting set dualization approach, which we claim have been overlooked in earlier work (Vescovi, 2011; Sebastiani & Vescovi, 2015).

The following subsections describe the different components of HgMUS in detail.

4.3.2 Computing Maximal Models

The use of maximal models for computing either MCSes of a formula or a set of clauses that contain an MUS was proposed in earlier work (Previti & Marques-Silva, 2013), which exploited SAT with preferences for computing maximal models (Giunchiglia & Maratea, 2006; Rosa & Giunchiglia, 2013). The use of SAT with preferences for computing maximal models is also exploited in related work (Sebastiani & Vescovi, 2009, 2015).

Computing maximal models of a formula $\mathcal{Q}$ can be reduced to the problem of extracting an MSS of a formula $\mathcal{Q}'$ (Marques-Silva et al., 2013a), where the clauses of $\mathcal{Q}$ are hard

Algorithm 2: Computation of Maximal Models

Input: $\mathcal{Q}$ a CNF formula
Output: (st, P): with st a Boolean and P an MxM (if it exists)

1 $(P, U, B) \leftarrow (\{\{x\} \mid \neg x \notin L(\mathcal{Q})\}, \{\{x\} \mid \neg x \in L(\mathcal{Q})\}, \emptyset)$
2 $(st, P, U) \leftarrow$ `InitialAssignment`$(\mathcal{Q} \cup P)$
3 **if not** st **then return** $(\mathsf{false}, \emptyset)$
4 **while** $U \neq \emptyset$ **do**
5 $\quad l \leftarrow$ `SelectLiteral`(U)
6 $\quad (st, \mu) =$ `SAT`$(\mathcal{Q} \cup P \cup B \cup \{l\})$
7 $\quad$ **if** st **then** $(P, U) \leftarrow$ `UpdateSatClauses`(μ, P, U)
8 $\quad$ **else** $(U, B) \leftarrow (U \setminus \{l\}, B \cup \{\neg l\})$
9 **return** (true, P) `// P is an MxM of` $\mathcal{Q}$

and, for each variable $x_i \in V(\mathcal{Q})$, it includes a unit soft clause $c_i = (x_i)$. Also, recent work (Marques-Silva et al., 2013a; Grégoire, Lagniez, & Mazure, 2014; Bacchus, Davies, Tsimpoukelli, & Katsirelos, 2014; Mencía, Previti, & Marques-Silva, 2015) has shown that state-of-the-art MCS/MSS computation approaches outperform SAT with preferences. HgMUS uses a dedicated algorithm based on the LinearSearch MCS extraction algorithm (Marques-Silva et al., 2013a), due to its good performance in MCS enumeration. Since all soft clauses are unit, it can also be related with the novel Literal-Based eXtractor algorithm (Mencía et al., 2015). Shown in Algorithm 2, it relies on making successive calls to a SAT solver. It maintains three sets of literals: P, an under-approximation of an MxM (i.e. positive literals s.t. $\mathcal{Q} \cup P \not\models \bot$), B, with negative literals $\neg l$ such that $\mathcal{Q} \cup P \cup \{l\} \models \bot$ (i.e. *backbone literals*), and U, with the remaining set of positive literals to be tested. Initially, P and U are initialized from a model $\mu \models \mathcal{Q}$, P including the literals appearing with positive polarity in μ and U including the literals with negative polarity in μ. Then, iteratively, it tries to extend P with a new literal $l \in U$, by testing the satisfiability of $\mathcal{Q} \cup P \cup B \cup \{l\}$. If it is satisfiable, all the literals in U satisfied by the model (including l) are moved to P. Otherwise, l is removed from U and $\neg l$ is added to B. This algorithm has a query complexity of $\mathcal{O}(|V(\mathcal{Q})|)$.

Algorithm 2 integrates a new technique, which consists in pre-initializating P with the pure positive literals appearing in $\mathcal{Q}$ and U with the remaining ones (line 1), and then requiring the literals of P to be satisfied by the initial assignment (line 2). It can be easily proved that these pure literals are included in all MxMs of $\mathcal{Q}$, so a number of calls to the SAT solver could be avoided. Moreover, the SAT solver will never branch on these variables, easing the decision problems. Note that, in this context, $\mathcal{Q}$ is made of two types of clauses: positive clauses blocking MCSes of the Horn formula, and negative clauses blocking MUSes. So, with this technique, the computation of MxMs is restricted to the variables representing groups appearing in some MUS of the Horn formula.[2]

4.3.3 Adding Blocking Clauses

One important aspect of HgMUS are the blocking clauses created and added to the formula $\mathcal{Q}$ (see Algorithm 1). These follow what was first proposed in eMUS (Previti & Marques-

2. SATPin (Manthey & Peñaloza, 2015; Manthey et al., 2016) also exploits this insight of *relevant* variables, but not in the context of MxMs.

Silva, 2013) and MARCO (Liffiton & Malik, 2013; Liffiton et al., 2016). For each MUS, the blocking clause consists of a set of negative literals, requiring at least one of the clauses in the MUS *not* to be included in future selected sets of clauses. For each MCS, the blocking clause consists of a set of positive literals, requiring at least one of the clauses in the MCS to be included in future selected sets of clauses. The way MCSes are handled is essential to prevent that MCS and sets containing the same MCS to be selected again. Although conceptually simple, it can be shown that existing approaches may not guarantee that supersets of MCSes (or subsets of the MSSes) are not selected. As argued before, this is the case with EL$^+$SAT (Vescovi, 2011; Sebastiani & Vescovi, 2015).

The following example illustrates the importance of blocking clauses.

Example 9. *Let us consider the group Horn formula $\mathcal{H}_{[A\sqsubseteq_{\mathcal{T}_{\text{exa}}}B]}$ in Example 7. $\mathcal{Q}$ is defined over variables $\{p_1,...,p_5\}$, each associated with a group in $\{\mathcal{G}_1,...,\mathcal{G}_5\}$. The following table illustrates the operation of* HgMUS*, showing, for each iteration, the maximal model, the group-MUS or group-MCS extracted and the blocking clause added to $\mathcal{Q}$.*

$\mathcal{Q}$	*MxM model*	*MUS*	*MCS*	*Blocking clause*
$\emptyset$	$\{p_1,p_2,p_3,p_4,p_5\}$	$\{\mathcal{G}_2,\mathcal{G}_4\}$	-	$b_1=(\neg p_2\vee\neg p_4)$
$\{b_1\}$	$\{p_1,p_3,p_4,p_5\}$	-	$\{\mathcal{G}_2\}$	$b_2=(p_2)$
$\{b_1,b_2\}$	$\{p_1,p_2,p_3,p_5\}$	$\{\mathcal{G}_1,\mathcal{G}_2,\mathcal{G}_3\}$	-	$b_3=(\neg p_1\vee\neg p_2\vee\neg p_3)$
$\{b_1,b_2,b_3\}$	$\{p_2,p_3,p_5\}$	-	$\{\mathcal{G}_1,\mathcal{G}_4\}$	$b_4=(p_1\vee p_4)$
$\{b_1,b_2,b_3,b_4\}$	$\{p_1,p_2,p_5\}$	-	$\{\mathcal{G}_3,\mathcal{G}_4\}$	$b_5=(p_3\vee p_4)$
$\{b_1,b_2,b_3,b_4,b_5\}$	*UNSAT*	-	-	-

If maximal models were blocked (instead of MCSes), the algorithm would require much more iterations. As illustrated in the following table, it would compute a large number of (non minimal) correction subsets (CS) before terminating.

$\mathcal{Q}$	*MxM model*	*MUS*	*CS*	*Blocking clause*
$\emptyset$	$\{p_1,p_2,p_3,p_4,p_5\}$	$\{\mathcal{G}_2,\mathcal{G}_4\}$	-	$b_1:(\neg p_2\vee\neg p_4)$
$\{b_1\}$	$\{p1,p_3,p_4,p_5\}$	-	$\{\mathcal{G}_2\}$	$b_2:(\neg p_1\vee\neg p_3\vee\neg p_4\vee\neg p_5)$
$\{b_1,b_2\}$	$\{p_3,p_4,p_5\}$	-	$\{\mathcal{G}_1,\mathcal{G}_2\}$	$b_3:(\neg p_3\vee\neg p_4\vee\neg p_5)$
$\{b_1,...,b_3\}$	$\{p_1,p_4,p_5\}$	-	$\{\mathcal{G}_2,\mathcal{G}_3\}$	$b_4:(\neg p_1\vee\neg p_4\vee\neg p_5)$
$\{b_1,...,b_4\}$	$\{p_1,p_3,p_4\}$	-	$\{\mathcal{G}_2,\mathcal{G}_5\}$	$b_5:(\neg p_1\vee\neg p_3\vee\neg p_4)$
$\{b_1,...,b_5\}$	$\{p_1,p_4\}$	-	$\{\mathcal{G}_2,\mathcal{G}_3,\mathcal{G}_5\}$	$b_6:(\neg p_1\vee\neg p_4)$
$\{b_1,...,b_6\}$	$\{p_1,p_2,p_3,p_5\}$	$\{\mathcal{G}_1,\mathcal{G}_2,\mathcal{G}_3\}$	-	$b_7:(\neg p_1\vee\neg p_2\vee\neg p_3)$
$\{b_1,...,b_7\}$	$\{p_1,p_2,p_5\}$	-	$\{\mathcal{G}_3,\mathcal{G}_4\}$	$b_8:(\neg p_1\vee\neg p_2\vee\neg p_5)$
$\{b_1,...,b_8\}$	$\{p_1,p_2\}$	-	$\{\mathcal{G}_3,\mathcal{G}_4,\mathcal{G}_5\}$	$b_9:(\neg p_1\vee\neg p_2)$
...	...	...	...	...

4.3.4 Deciding Satisfiability of Horn Formulae

It is well-known that Horn formulae can be decided in linear time (Dowling & Gallier, 1984; Itai & Makowsky, 1987; Minoux, 1988). HgMUS implements the LTUR algorithm (Minoux, 1988). There are important reasons for this choice. First, LTUR is expected to be more efficient than plain unit propagation, since only variables assigned value 1 need to be

Algorithm 3: LTUR

```
   Input: H: input Horn formula
   Output: falsified clause, if any; α: antecedents
1  (Q, η, γ, α) ← Initialize(H)
2  while Q ≠ ∅ do                                  // Q: queue of variables assigned 1
3      x_j ← ExtractFirstVariable(Q)
4      foreach c_i ∈ N(x_j) do
5          η(c_i) ← η(c_i) − 1
6          if η(c_i) = 0 then
7              if γ(c_i) then return ({c_i}, α)
8              x_r ← PickVariable(P(c_i))
9              if α(x_r) = ∅ then
10                 AppendToQueue(Q, x_r)
11                 α(x_r) ← {c_i}
12 return (∅, α)
```

propagated. Second, most implementations of unit propagation in CDCL SAT solvers (i.e. that use watched literals) are not guaranteed to run in linear time (Gent, 2013); this is for example the case with *all* implementations of Minisat (Eén & Sörensson, 2003) and its variants, for which unit propagation runs in worst-case quadratic time. As a result, using an off-the-shelf SAT solver and exploiting only unit propagation (as is done for example in earlier work (Sebastiani & Vescovi, 2009, 2015; Manthey & Peñaloza, 2015)) is unlikely to be the most efficient solution.

The organization of LTUR is summarized in Algorithm 3. LTUR can be viewed as one-sided unit propagation, in the sense that only variables assigned value 1 are propagated; all other variables are assigned value 0 by default. In this context, $\eta : \mathcal{H} \to \mathbb{N}_0$ associates a counter with each clause, representing the number of negative literals *not* assigned value 0. The function $\gamma : \mathcal{H} \to \{0, 1\}$ denotes whether a clause $c \in \mathcal{F}$ is a goal clause, in which case $\gamma(c) = 1$. Finally, $\alpha : V(\mathcal{H}) \to 2^{\mathcal{H}}$ is a function that assigns to each variable x a set of clauses $\alpha(x)$ that are deemed responsible for assigning value 1 to x. If the value of x is not determined to be 1, then $\alpha(x) = \emptyset$. Besides, we denote by $N(x)$ the set of clauses where the variable x appears in negative polarity, and $P(c)$ denotes the set of of variables appearing in positive polarity in clause c. The initialization step sets the initial values of the η counters, the α values, and the value of the γ flag. The queue Q is initialized with the variables in the unit positive clauses. The main loop analyzes the variables assigned value 1 in order. Notice that a variable x is assigned value 1 iff $\alpha(x) \neq \emptyset$. For each variable $x \in Q$, the counter η of the clauses where x occurs as a negative literal is decreased. If $\eta(c) = 0$ for some clause c, then either the formula is inconsistent, if c is a goal clause, or the positive literal of c is assigned value 1 and added to Q. The process is repeated while variables are assigned value 1.

Besides the advantages listed above, the use of a linear time algorithm for deciding the satisfiability of Horn formulae turns out to be instrumental for MUS extraction, as shown in the next section. In order to use LTUR for MUS extraction, an incremental version has been implemented, which allows for the incremental addition of clauses to the formula and incremental identification of variables assigned value 1. Clearly, the amortized run time of

Algorithm 4: Insertion-based (de Siqueira N. & Puget, 1988) MUS extraction using LTUR (Minoux, 1988)

Input: $\mathcal{H}$, denotes the $\mathcal{G}_0$ clauses; $\mathcal{I}$, denotes the set of (individual) group clauses
Output: $\mathcal{M}$, denotes the computed MUS

```
1  (M, c_r) ← (H, 0)
2  LTUR_prop(M, M)                          // Start by propagating G_0 clauses
3  while true do
4      if c_r > 0 then
5          M ← M ∪ {c_r}                    // Add transition clause c_r to M
6          if not LTUR_prop(M, {c_r}) then
7              LTUR_undo(M, M)
8              return M \ H                 // Remove G_0 clauses from computed MUS
9      S ← ∅
10     while true do
11         c_r ← SelectRemoveClause(I)      // Target transition clause
12         S ← S ∪ {c_r}
13         if not LTUR_prop(M ∪ S, {c_r}) then
14             I ← S \ {c_r}                // Update working set of groups
15             LTUR_undo(M, S)
16             break                        // c_r represents a transition clause
```

LTUR, after adding $m = |\mathcal{F}|$ clauses, is $\mathcal{O}(||\mathcal{F}||)$, with $||\mathcal{F}||$ the number of literals appearing in $\mathcal{F}$.

4.3.5 MUS Extraction in Horn Formulae

For arbitrary CNF formulae, a number of approaches exist for MUS extraction, with the most commonly used one being the deletion-based approach (Bakker, Dikker, Tempelman, & Wognum, 1993; Belov et al., 2012), but other alternatives include the QuickXplain algorithm (Junker, 2004) and the more recent Progression algorithm (Marques-Silva, Janota, & Belov, 2013b). It is also well-known and generally accepted that, due to its query complexity, the insertion-based algorithm (de Siqueira N. & Puget, 1988) for MUS extraction is in practice not competitive with existing alternatives (Belov et al., 2012).

Somewhat surprisingly, this is not the case with Horn formulae when (an incremental implementation of) the LTUR algorithm is used. A modified insertion-based MUS extraction algorithm that exploits LTUR is shown in Algorithm 4. `LTUR_prop` propagates the consequences of adding some new set of clauses, given some existing incremental context. `LTUR_undo` unpropagates the consequences of adding some set of clauses (in order), given some existing incremental context. Besides, in this algorithm, clauses are represented as integers (0 means no clause). The organization of the algorithm mimics the standard insertion-based MUS extraction algorithm (de Siqueira N. & Puget, 1988), but the use of the incremental LTUR yields run time complexity that improves over other approaches. Consider the operation of the standard insertion-based algorithm (de Siqueira N. & Puget, 1988), in which clauses are iteratively added to the working formula. When the formula becomes unsatisfiable, a *transition clause* (Belov et al., 2012) has been identified, which is then added to the MUS being constructed. The well-known query complexity of the

insertion-based algorithm is $\mathcal{O}(m \times k)$ where m is the number of clauses and k is the size of a largest MUS. Now consider that the incremental LTUR algorithm is used. To find the first transition clause, the amortized run time is $\mathcal{O}(||\mathcal{F}||)$. Clearly, this holds true for *any* transition clause, and so the run time of MUS extraction with the LTUR algorithm becomes $\mathcal{O}(|\mathcal{M}| \times ||\mathcal{F}||)$, where $\mathcal{M} \subseteq \mathcal{I}$ is a largest MUS. Algorithm 4 highlights the main differences with respect to a standard insertion-based MUS extraction algorithm. In contrast, observe that for a deletion-based algorithm the run time complexity will be $\mathcal{O}(|\mathcal{I}| \times ||\mathcal{F}||)$. In situations where the sizes of MUSes are much smaller than the number of groups in $\mathcal{I}$, this difference can be significant. As a result, when extracting MUSes from Horn formulae, and when using a polynomial time incremental decision procedure, an insertion-based algorithm should be used instead of other more commonly used alternatives.

5. Experimental Results

This section[3] evaluates the methods proposed in the paper, EL2MCS and HgMUS, and compares them with other state-of-the-art tools for axiom pinpointing in $\mathcal{EL}^+$. For this purpose, a set of benchmarks from well-known bio-medical ontologies was considered. These have been used in earlier work, e.g. (Sebastiani & Vescovi, 2009, 2015; Arif et al., 2015a, 2015b; Manthey & Peñaloza, 2015; Manthey et al., 2016). Since all experiments consist of converting axiom pinpointing problems into group-MUS enumeration problems, the tool that uses HgMUS[4] as its back-end is named EL2MUS. Thus, in this section, the results for EL2MUS illustrate the performance of the group-MUS enumerator described in Section 4.3. HgMUS was also used as a back-end in the BEACON tool (Arif et al., 2016), which incorporates its own implementation of the Horn encoding used by EL$^+$SAT (Sebastiani & Vescovi, 2009; Vescovi, 2011; Sebastiani & Vescovi, 2015). BEACON also integrates the state-of-the-art tool Forqes (Ignatiev et al., 2015), also based on implicit minimal hitting set dualization, for computing smallest MUSes, which correspond to the smallest MinAs for a given subsumption relation w.r.t. a TBox. In this section, EL2MUS uses EL$^+$SAT as a front-end (for generating the Horn encoding) for comparison purposes, but the BEACON tool's front-end could be used instead.

5.1 Experimental Setup

Each considered instance represents the problem of explaining a particular subsumption relation (query) entailed in a medical ontology. Four standard medical ontologies[5] are considered: GALEN (Rector & Horrocks, 1997), the Gene Ontology (GO) (Ashburner, Ball, Blake, Botstein, Butler, Cherry, Davis, Dolinski, Dwight, Eppig, & et al., 2000), NCI (Sioutos, de Coronado, Haber, Hartel, Shaiu, & Wright, 2007) and SNOMED CT (version 2009) (Spackman et al., 1997). For GALEN, we consider two variants: FULL-GALEN and NOT-GALEN. The most important ontology is SNOMED CT and, due to its huge size, it also produces the hardest axiom pinpointing instances. For each ontology

3. Additional material and detailed results for the submission are available at http://logos.ucd.ie/web/doku.php?id=jair16-submission.
4. HgMUS is available at http://logos.ucd.ie/web/doku.php?id=hgmus.
5. GO, GALEN and NCI ontologies are freely available at http://lat.inf.tu-dresden.de/~meng/toyont.html. The SNOMED CT ontology was requested from IHTSDO under a nondisclosure license agreement.

(including the GALEN variants) 100 queries are considered; 50 random (expected to be easier) and 50 sorted (expected to have a large number of minimal explanations) queries. These queries were proposed in previous work, e.g. (Vescovi, 2011; Sebastiani & Vescovi, 2015). Hence, the experimental setup consists of a total of 500 axiom pinpointing problem instances.

As mentioned before, all the instances have been obtained from the encoding proposed in (Sebastiani & Vescovi, 2009; Vescovi, 2011; Sebastiani & Vescovi, 2015). In addition, two different experiments were considered by applying two different simplification techniques to the problem instances, both of which were proposed in (Vescovi, 2011; Sebastiani & Vescovi, 2015). Simplification techniques are of common use in axiom pinpointing, such as the so-called *reachability-based modularization* in the DL domain (Baader & Suntisrivaraporn, 2008). These techniques allow for extracting an (often small) subset of the ontology such that it contains all the MinAs and diagnoses for a given subsumption relation, and their application often plays an important role in the efficiency of axiom pinpointing tools. The first technique considered in these experiments uses the *Cone-Of-Influence* (COI) reduction. These are reduced instances in both the size of the Horn formula and the number of original axioms in the ontology, but are still quite large. Similar techniques are exploited in related work (Baader et al., 2006; Ludwig, 2014; Manthey & Peñaloza, 2015; Manthey et al., 2016). The second one considers the more effective reduction technique (referred to as x2), consisting in applying the COI technique, re-encoding the Horn formula into a reduced ontology, and encoding this ontology again into a Horn formula. This results in small Horn formulae, which will be useful to evaluate the algorithms when there are a large number of MUSes or MCSes.

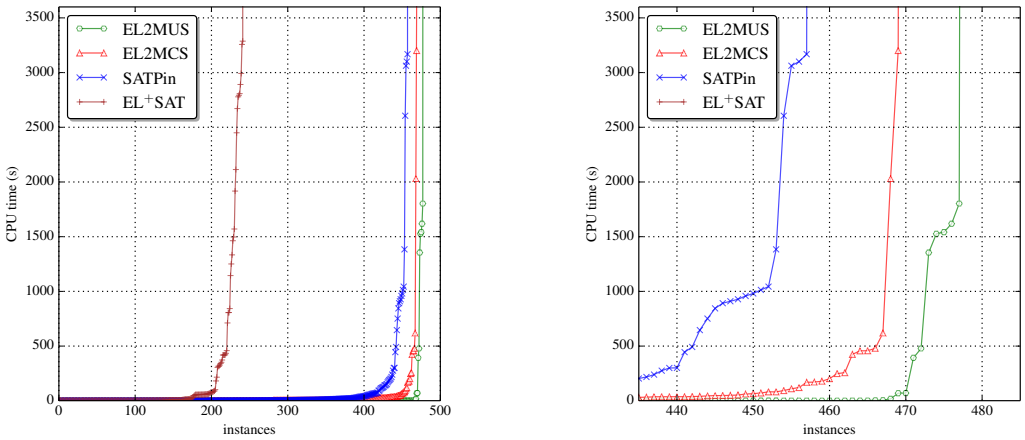
The experiments compare EL2MCS and EL2MUS to different algorithms; specifically, EL$^+$SAT (Sebastiani & Vescovi, 2009, 2015), CEL (Baader et al., 2006), Just (Ludwig, 2014) and SATPin (Manthey & Peñaloza, 2015; Manthey et al., 2016). EL$^+$SAT has been previously shown to outperform CEL, whereas SATPin has been shown to outperform the MUS enumerator MARCO (Liffiton et al., 2016).

The comparison with CEL and Just imposes a number of constraints. First, CEL only computes 10 MinAs, so all comparisons with CEL only consider reporting the first 10 MinAs/MUSes. Also, CEL uses a simplification technique similar to COI, so CEL is considered in the first experiments. Second, Just operates on selected subsets of $\mathcal{EL}^+$, since it is unable to handle general role inclusions. As a result, all comparisons with Just consider solely the problem instances for which Just can compute correct results. Just accepts the simplified x2 ontologies, so it is considered in the second series of experiments. The comparison with these tools is presented at the end of the section.

EL2MUS interfaces the SAT solver Minisat 2.2 (Eén & Sörensson, 2003) for computing maximal models. All the experiments were performed on a Linux cluster (2 GHz) and the algorithms were given a time limit of 3600s and a memory limit of 4 GB.

5.2 Assessment of SAT-Based Approaches

The first series of experiments aims at evaluating the performance of EL2MCS and EL2MUS compared to EL$^+$SAT and SATPin. All these methods work on the Horn encoding provided



(a) Full range [0, 500]

(b) Zoomed in the range [435-485]

Figure 3: Cactus plots comparing EL$^+$SAT, SATPin, EL2MCS and EL2MUS on the COI instances

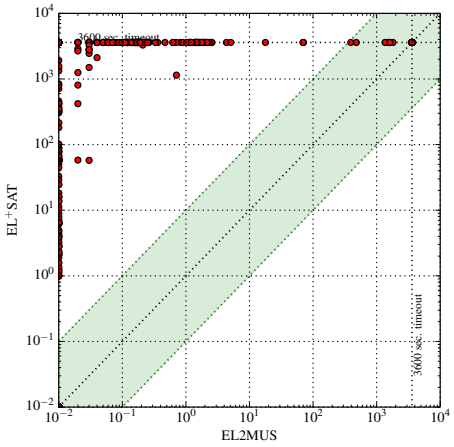
by EL$^+$SAT. Hence, the systems were run on the same Horn formulae, simplified with either the COI or x2 reduction techniques. The results are presented in separate subsections.

5.2.1 COI Instances

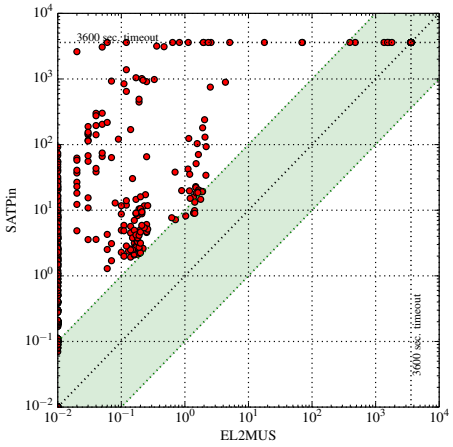
For the COI instances, Figure 3 summarizes the results for EL$^+$SAT, EL2MCS, SATPin and EL2MUS. It shows the running times for solving the different axiom pinpointing instances, i.e. enumerating all the MinAs and proving that there is no MinA left. As it can be easily observed, EL2MCS has a slight performance advantage over SATPin which, in turn solves much more instances than EL$^+$SAT. On the other hand, EL2MUS terminates for more instances than any of the other tools. Noticeably, by the time limit EL2MCS is able to solve all the instances EL$^+$SAT or SATPin can solve, and EL2MUS solves all the instances solved by EL2MCS. The differences among SATPin, EL2MCS and ELMUS can be better observed in Figure 3b, which shows a clear performance gap in favor of EL2MCS and EL2MUS.

Figure 4 shows scatter plots comparing EL2MUS with the different tools. From this figure it can be concluded that the performance of EL2MUS exceeds the performance of any of the other tools by at least one order of magnitude (and often by more), except for a few outlying instances. Figure 4d summarizes the results in the scatter plots, where the percentages shown are computed for problem instances for which at least one of the tools takes more than 0.001s. The first four rows of the table show a pairwise comparison among the methods, reporting the percentage of the instances each method takes less time than other ones in solving the instances. The last five rows of the table show the percentage of the instances for which the performance gap in favor of EL2MUS is greater than one to five orders of magnitude. Observe that EL2MUS outperforms all other tools in all of the problem instances and, for many cases, with two or more orders of magnitude improvement.

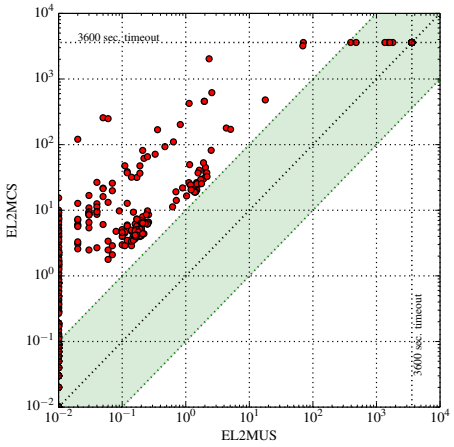
The results presented so far focus on the ability of the tools for completing the enumeration of the MinAs, and do not provide information about their performance when a complete enumeration is not possible. Table 2 shows, for each family of instances (each with 100), the number of instances solved (#Sol.) and the total number of MinAs computed (#Mi-



(a) Comparison with EL$^+$SAT



(b) Comparison with SATPin



(c) Comparison with EL2MCS

% wins	EL$^+$SAT	SATPin	EL2MCS
EL$^+$SAT	–	20.29%	17.66%
SATPin	79.71%	–	19.13%
EL2MCS	82.34%	80.41%	–
EL2MUS	100.0%	100.0%	100.0%
$> 10^1$x	98.09%	96.78%	98.41%
$> 10^2$x	97.55%	72.07%	58.07%
$> 10^3$x	96.46%	47.75%	14.09%
$> 10^4$x	74.05%	06.49%	00.00%
$> 10^5$x	31.10%	00.45%	00.00%

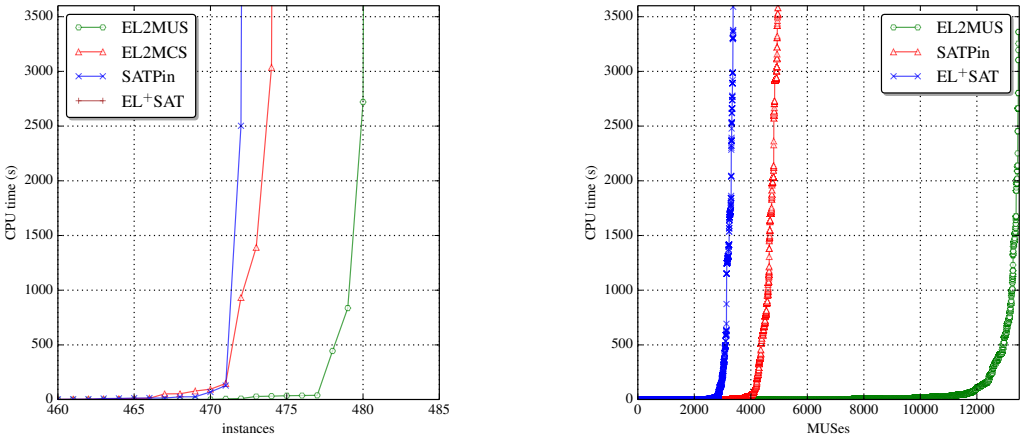
(d) Summary table

Figure 4: Scatter plots for COI instances

nAs) by the time limit by each of the methods (including this information for the instances that were not solved). Since EL2MCS and EL2MUS compute diagnoses as well, the total number of them is also reported (#Diag.) for these tools. The hardest instances come from the SNOMED CT ontology, where EL2MUS is able to report more MinAs in about a factor of 10 than any other tool by the time limit . For the other ontologies, both EL2MCS and EL2MUS are able to solve all the problem instances, SATPin is able to solve most of them and EL$^+$SAT computes most of the MinAs by the time limit, although it is often unable to terminate. Noticeably, the few SNOMED CT instances that cannot be solved by EL2MCS and EL2MUS contain a very large number of diagnoses (that is, MCSes). This illustrates the main drawback of EL2MCS which, in contrast to the other tools is unable to report a single MinA by the time limit in these cases as a result of being unable to complete the enumeration of the diagnoses. At the same time, the results suggest that the number of diagnoses for a given subsumption relation plays an important role in the difficulty of axiom pinpointing. EL2MUS stands out in both its capability for computing diagnoses and MinAs for these hardest instances compared to EL2MCS and any other tool respectively.

Table 2: SAT-based approaches. COI instances

Ontology	EL$^+$SAT		SATPin		EL2MCS			EL2MUS		
	#Sol.	#MinA	#Sol.	#MinA	#Sol.	#MinA	#Diag.	#Sol.	#MinA	#Diag.
GENE	87	1126	100	1134	100	1134	20011	100	1134	20011
NCI	77	664	93	661	100	678	78989	100	678	78989
NOT-GALEN	34	159	100	159	100	159	702	100	159	702
FULL-GALEN	17	137	100	139	100	139	776	100	139	776
SNOMED CT	27	1002	65	973	70	439	78839	78	9349	3199135
Total	242	3088	458	3066	470	2549	179317	**478**	**11459**	**3299613**



(a) Solved instances

(b) Reported MUSes

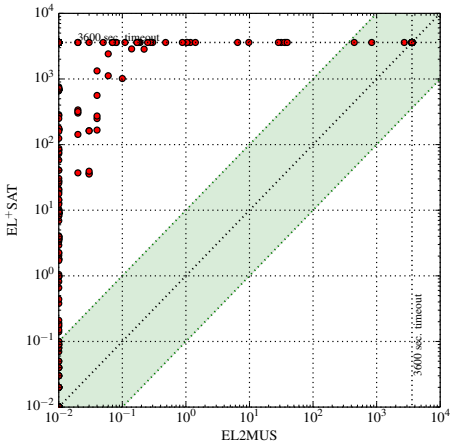
Figure 5: Cactus plots comparing EL$^+$SAT, SATPin, EL2MCS and EL2MUS on the x2 instances

5.2.2 x2 Instances

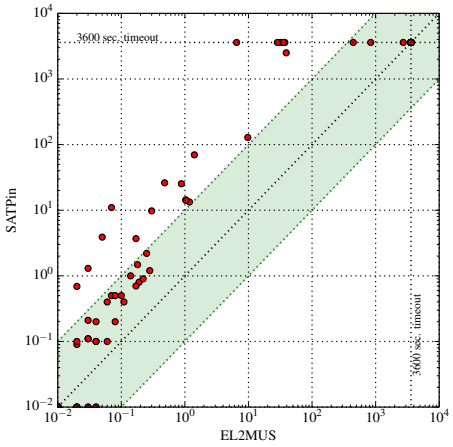
The x2 instances contain exactly the same MinAs and diagnoses than the COI instances, but these are significantly simpler in terms of size. Thus, whereas the COI instances can serve to assess the scalability of each approach, the x2 instances highlight the expected performance in representative settings for axiom pinpointing.

Figure 5a summarizes the performance of the tools EL$^+$SAT, SATPin, EL2MCS and EL2MUS. Due to its poor performance, EL$^+$SAT is not included in this plot. This tool terminated on only 317 instances. As in the previous experiment, EL2MUS exhibits an observable performance edge in terms of solved instances.

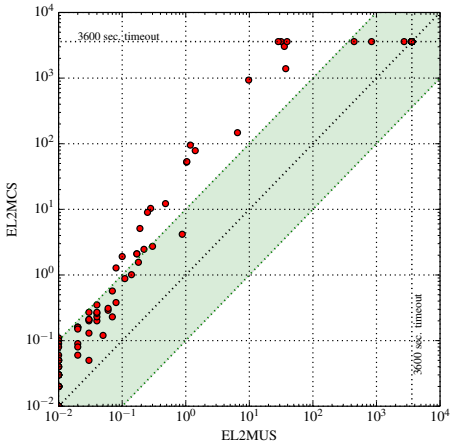
A pairwise comparison between the different tools is summarized in Figure 6. Although not as impressive as for the COI instances, EL2MUS still consistently outperforms all other tools. Figure 6d summarizes the results, where as before the percentages shown are computed for problem instances for which at least one of the tools takes more than 0.001s. Observe that, for these easier instances, SATPin becomes competitive with EL2MUS. Nevertheless, for instances taking more than 0.1s, EL2MUS outperforms SATPin on 100% of the instances. Thus, the 67.69% shown in the table result from instances for which both SATPin and EL2MUS take at most 0.04s. The summary table also lists the number of computed MUSes/MinAs for the 19 instances for which EL2MUS does *not* terminate (all of the other tools also do not terminate for these 19 instances). EL2MUS computes 9948 MUSes in total. As can be observed from the table, the other tools lag behind, and com-



(a) Comparison with EL$^+$SAT



(b) Comparison with SATPin



(c) Comparison with EL2MCS

% wins	EL$^+$SAT	SATPin	EL2MCS
EL$^+$SAT	–	00.00%	00.00%
SATPin	100.0%	–	91.55%
EL2MCS	100.0%	08.45%	–
EL2MUS	100.0%	67.69%	99.32%

	EL$^+$SAT	SATPin	EL2MCS
# MUSes	788	1484	0
Δ MUSes	9160	8864	9948

(d) Summary table

Figure 6: Scatter plots for x2 instances

pute significantly fewer MUSes. Also, as noted earlier in the paper, the main issue with EL2MCS is demonstrated with these results; for these 19 instances, EL2MCS is unable to compute any MUSes. The comparison with the other tools, EL$^+$SAT and SATPin, reveals that EL2MUS computes respectively in excess of a factor of 10 and of 5 more MUSes.

EL2MUS not only terminates on more instances than any other approach and computes more MUSes for the unsolved instances; it also reports the sequences of MUSes much faster. Figure 5b shows, for each computed MUS over the whole set of instances, the time each MUS was reported. This figure compares EL$^+$SAT, SATPin and EL2MUS, as these are the only methods able to report MUSes from the beginning. The results confirm that EL2MUS is able to find many more MUSes in less time than the alternatives.

Finally, Table 3 summarizes the results for each family of problem instances. As can be observed, all the tools exhibit a clear improvement w.r.t. the COI instances (see Table 2). The simplified instances allow EL$^+$SAT, SATPin, EL2MCS and EL2MUS for solving 75, 15, 5 and 3 more instances respectively. The improvement of EL2MCS and EL2MUS in terms of solved instances is not as dramatic as for the other tools, since the open instances are really challenging, as suggested by the huge number of diagnoses reported by EL2MUS for the

Table 3: SAT-based approaches. x2 instances

Ontology	EL$^+$SAT		SATPin		EL2MCS			EL2MUS		
	#Sol.	#MinA	#Sol.	#MinA	#Sol.	#MinA	#Diag.	#Sol.	#MinA	#Diag.
GENE	99	1134	100	1134	100	1134	20011	100	1134	20011
NCI	86	672	100	678	100	678	78989	100	678	78989
NOT-GALEN	77	159	100	159	100	159	702	100	159	702
FULL-GALEN	24	137	100	139	100	139	776	100	139	776
SNOMED CT	31	1570	73	2847	75	919	22330370	81	11364	32430103
Total	317	3672	473	4957	475	3029	22430848	**481**	**13474**	**32530581**

SNOMED CT instances. These very large numbers confirm that the number of diagnoses serves as a very accurate indicator of the difficulty of the axiom pinpointing problems. It should be noted that EL2MUS is able to solve more COI instances (478) than any other tool x2 instances, and the same applies for the number of reported MinAs. Noticeably, EL2MCS improves to a great extent regarding its capability for computing diagnosis and EL2MUS is able to compute a factor of 10 more diagnoses for the unsolved instances w.r.t the COI instances. In the x2 instances EL2MUS is able to compute around 33% more diagnoses than EL2MCS.

These experimental results confirm the superiority of EL2MUS over any other SAT-based axiom pinpointing tool. Not only EL2MUS is able to complete the enumeration of the MinAs for more instances than the other tools, but it also shows a clear advantage in enumerating MinAs for the problem instances that cannot be solved by the time limit, as well as in computing diagnoses.

5.3 Assessment of Other Axiom Pinpointing Tools

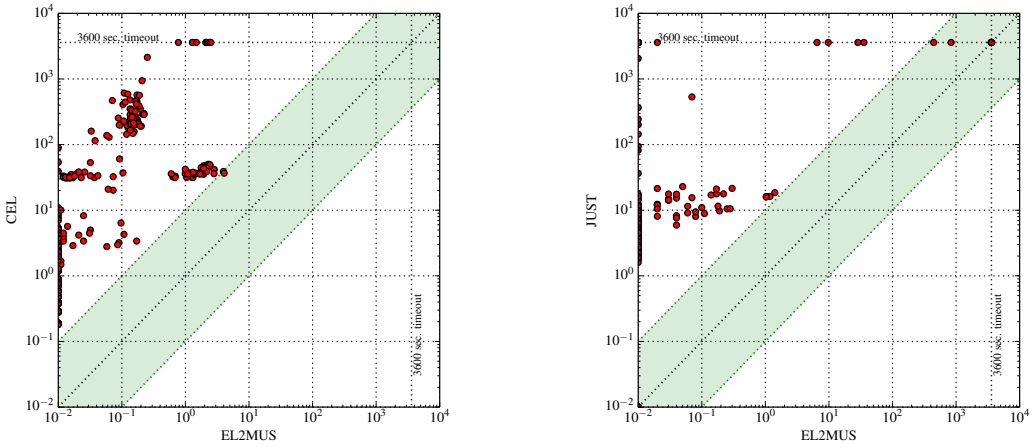
This section compares EL2MUS to DL-based axiom pinpointing tools, namely CEL (Baader et al., 2006) and Just (Ludwig, 2014). Figure 7 shows scatter plots comparing EL2MUS with CEL and Just, respectively for the COI and x2 instances.

As indicated earlier, CEL only computes 10 MinAs, and so the run times shown are for computing the first 10 MinA/MUSes. For problem instances with less than 10 MinAs, the reported run time corresponds to the time taken by the algorithms in terminating after a complete enumeration of the MinAs. As can be observed, the performance edge of EL2MUS is clear, with the performance gap exceeding 1 order of magnitude almost without exception.

Just (Ludwig, 2014) is a recent state-of-the-art axiom pinpointing tool for the less expressive $\mathcal{ELH}$ DL. Thus, not all subsumption relations can be represented and analyzed. The results shown are for the subsumption relations for which Just gives the correct results. In total, 382 instances could be considered and are shown in the plot. As before, the performance edge of EL2MUS is clear, with the performance gap exceeding 1 order of magnitude without exception. In this case, since the x2 instances are in general much simpler, the performance gap is even more significant.

6. Conclusions

Axiom pinpointing is a fundamental reasoning task for the development and maintainment of description logic ontologies, as it constitutes the basis for debugging and repair of the



(a) EL2MUS vs. CEL on the COI instances

(b) EL2MUS vs. Just on the x2 instances

Figure 7: Comparison of EL2MUS with CEL and with Just

specified knowledge. This paper builds on earlier work relating axiom pinpointing in the $\mathcal{EL}$ family of lightweight DLs and propositional satisfiability. The present work makes several contributions towards efficient SAT-based axiom pinpointing in these logics, which can be applied in existing large-scale ontologies.

First, the paper shows that given the propositional Horn encoding of the classification of an ontology proposed in earlier work (Sebastiani & Vescovi, 2009, 2015), axiom pinpointing in the $\mathcal{EL}^+$ corresponds to enumerating all the group-MUSes of a Horn formula. This reduction serves to capitalize on state-of-the-art SAT-based approaches for addressing group-MUS enumeration as a back-end for an efficient enumeration of all MinAs of a given subsumption relation.

Building on recent work on MUS enumeration, the paper proposes two approaches for enumerating the group-MUSes of Horn formulae. In particular, these approaches yield new methods for enumerating all MinAs and solving other problems related to axiom pinpointing. Both methods rely on the well-known hitting set duality relationship between MUSes and MCSes of propositional formulae. The first method, known as EL2MCS, follows an explicit hitting set dualization scheme, in which all the group-MCSes are computed first and then the group-MUSes are obtained by computing the minimal hitting sets of this set of group-MCSes. The second method, called HgMUS, develops an implicit hitting set dualization approach, interleaving the computation of group-MCSes and group-MUSes of the Horn formula. HgMUS builds on partial MUS enumerators, such as other MUS tools like MARCO and eMUS, but incorporates a number of features specifically designed for dealing with Horn formulae which improve the overall performance of the method.

It is interesting to notice that, although the work on axiom pinpointing in DLs usually focuses on finding or enumerating MinAs, there exist several related tasks where computing repairs is more important. For example, in error-tolerant reasoning (Ludwig & Peñaloza, 2014; Bienvenu & Rosati, 2013; Bienvenu, Bourgaux, & Goasdoué, 2014) and iterative ontology update (Peñaloza & Thuluva, 2015), one is interested in analysing some properties on the set of all repairs for a consequence. Given the close connection between the repairs of a consequence and the MCSes obtained from the Horn encoding, the two methods presented

here are directly capable of enumerating repairs, making them specially suitable in those circumstances.

Both methods were implemented, and their performance compared to other existing techniques for axiom pinpointing in lightweight DLs over a large set of benchmarks derived from existing ontologies developed for the bio-medical domains. The experimental results obtained indicate very remarkable improvements of the proposed methods over the state of the art. Both EL2MCS and HgMUS are able to solve more instances than any other method. The results also show a clear advantage of HgMUS over EL2MCS, specially for instances where there is a very large number of MCSes/diagnoses, where EL2MCS is often unable to deliver a single MUS/MinA, whereas HgMUS is able to compute many solutions. In a few of these hard instances, HgMUS was able to generate all MinAs; a feat that had not been accomplished by any tool before.

The results obtained with these tools suggest that propositional satisfiability can contribute very significantly to the area of description logics in general, and in particular for axiom pinpointing and other non-standard reasoning tasks. This positive outcome encourages further research on establishing deeper relationships between DL reasoning problems and SAT. This deeper interleaving of both areas could result in the development of more efficient algorithms for solving a wide range of problems in Description Logics. In the short term, one of the important questions to be answered is whether some of the techniques developed in this paper can be extended to cover more expressive DLs. It would be also interesting to understand how to adapt the methods to solve other related reasoning tasks efficiently.

Acknowledgments

This work was funded in part by SFI grant BEACON (09/IN.1/I2618), by DFG grant DFG HO 1294/11-1, and by Spanish grant TIN2013-46511-C2-2-P. The contribution of the researchers associated with the SFI grant BEACON is also acknowledged.

References

Arif, M. F., Mencía, C., Ignatiev, A., Manthey, N., Peñaloza, R., & Marques-Silva, J. (2016). BEACON: an efficient SAT-based tool for debugging *EL+* ontologies. In *SAT*, pp. 521–530.

Arif, M. F., Mencía, C., & Marques-Silva, J. (2015a). Efficient axiom pinpointing with EL2MCS. In *KI*, pp. 225–233.

Arif, M. F., Mencía, C., & Marques-Silva, J. (2015b). Efficient MUS enumeration of Horn formulae with applications to axiom pinpointing. In *SAT*, pp. 324–342.

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., & et al. (2000). Gene ontology: tool for the unification of biology. *Nat. Genet.*, *25*(1), 25–29.

Baader, F., Brandt, S., & Lutz, C. (2005). Pushing the $\mathcal{EL}$ envelope. In *IJCAI*, pp. 364–369.

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., & Hollunder, B. (1995). Embedding defaults into terminological knowledge representation formalisms. *J. Autom. Reasoning*, *14*(1), 149–180.

Baader, F., Lutz, C., & Suntisrivaraporn, B. (2006). CEL - A polynomial-time reasoner for life science ontologies. In *IJCAR*, pp. 287–291.

Baader, F., & Peñaloza, R. (2010). Axiom pinpointing in general tableaux. *J. Log. Comput.*, *20*(1), 5–34.

Baader, F., Peñaloza, R., & Suntisrivaraporn, B. (2007). Pinpointing in the description logic $\mathcal{EL}^+$. In *KI*, pp. 52–67.

Baader, F., & Suntisrivaraporn, B. (2008). Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}^+$. In *KR-MED*.

Bacchus, F., Davies, J., Tsimpoukelli, M., & Katsirelos, G. (2014). Relaxation search: A simple way of managing optional clauses. In *AAAI*, pp. 835–841.

Bailey, J., & Stuckey, P. J. (2005). Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In *PADL*, pp. 174–186.

Bakker, R. R., Dikker, F., Tempelman, F., & Wognum, P. M. (1993). Diagnosing and solving over-determined constraint satisfaction problems. In *IJCAI*, pp. 276–281.

Belov, A., Lynce, I., & Marques-Silva, J. (2012). Towards efficient MUS extraction. *AI Commun.*, *25*(2), 97–116.

Bienvenu, M., Bourgaux, C., & Goasdoué, F. (2014). Querying inconsistent description logic knowledge bases under preferred repair semantics. In Brodley, C. E., & Stone, P. (Eds.), *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 996–1002. AAAI Press.

Bienvenu, M., & Rosati, R. (2013). Tractable approximations of consistent query answering for robust ontology-based data access. In Rossi, F. (Ed.), *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*. IJCAI/AAAI.

Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.). (2009). *Handbook of Satisfiability*, Vol. 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

Birnbaum, E., & Lozinskii, E. L. (2003). Consistent subsets of inconsistent systems: structure and behaviour. *J. Exp. Theor. Artif. Intell.*, *15*(1), 25–46.

Cook, S. A. (1971). The complexity of theorem-proving procedures. In *STOC*, pp. 151–158.

de Siqueira N., J. L., & Puget, J.-F. (1988). Explanation-based generalisation of failures. In *ECAI*, pp. 339–344.

Dowling, W. F., & Gallier, J. H. (1984). Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *J. Log. Program.*, *1*(3), 267–284.

Eén, N., & Sörensson, N. (2003). An extensible SAT-solver. In *SAT*, pp. 502–518. MiniSat 2.2 is available from: https://github.com/niklasso/minisat.git.

Gent, I. (2013). Optimal implementation of watched literals and more general techniques. *Journal of Artificial Intelligence Research*, *48*, 231–252.

Giunchiglia, E., & Maratea, M. (2006). Solving optimization problems with DLL. In *ECAI*, pp. 377–381.

Grégoire, É., Lagniez, J., & Mazure, B. (2014). An experimentally efficient method for (MSS, CoMSS) partitioning. In *AAAI*, pp. 2666–2673.

Hëbert, C., Bretto, A., & Crëmilleux, B. (2007). A data mining formalization to improve hypergraph minimal transversal computation.. *Fundammenta Informaticase*, *80*(4), 415–433.

Heras, F., Morgado, A., & Marques-Silva, J. (2015). MaxSAT-based encodings for group MaxSAT. *AI Commun.*, *28*(2), 195–214.

Ignatiev, A., Previti, A., Liffiton, M., & Marques-Silva, J. (2015). Smallest MUS extraction with minimal hitting set dualization. In *CP*.

Itai, A., & Makowsky, J. A. (1987). Unification as a complexity measure for logic programming. *J. Log. Program.*, *4*(2), 105–117.

Junker, U. (2004). QuickXplain: Preferred explanations and relaxations for over-constrained problems. In *AAAI*, pp. 167–172.

Kalyanpur, A., Parsia, B., Horridge, M., & Sirin, E. (2007). Finding all justifications of OWL DL entailments. In *ISWC*, pp. 267–280.

Kalyanpur, A., Parsia, B., Sirin, E., & Grau, B. C. (2006). Repairing unsatisfiable concepts in owl ontologies... pp. 170–184.

Lahiri, S. K., Nieuwenhuis, R., & Oliveras, A. (2006). SMT techniques for fast predicate abstraction. In *CAV*, pp. 424–437.

Liffiton, M. H., & Malik, A. (2013). Enumerating infeasibility: Finding multiple MUSes quickly. In *CPAIOR*, pp. 160–175.

Liffiton, M. H., Previti, A., Malik, A., & Marques-Silva, J. (2016). Fast, flexible MUS enumeration. *Constraints*, *21*(2), 223–250.

Liffiton, M. H., & Sakallah, K. A. (2008). Algorithms for computing minimal unsatisfiable subsets of constraints. *J. Autom. Reasoning*, *40*(1), 1–33.

Ludwig, M. (2014). Just: a tool for computing justifications w.r.t. ELH ontologies. In *ORE*.

Ludwig, M., & Peñaloza, R. (2014). Error-tolerant reasoning in the description logic $\mathcal{EL}$. In *JELIA*, pp. 107–121.

Manthey, N., & Peñaloza, R. (2015). Exploiting SAT technology for axiom pinpointing. Tech. rep. LTCS 15-05, Chair of Automata Theory, Institute of Theoretical Computer Science, Technische Universität Dresden. Available from https://ddll.inf.tu-dresden.de/web/Techreport3010.

Manthey, N., Peñaloza, R., & Rudolph, S. (2016). Efficient axiom pinpointing in EL using SAT technology. In *Proceedings of the 29th International Workshop on Description Logics*, Vol. 1577 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Marques-Silva, J., Heras, F., Janota, M., Previti, A., & Belov, A. (2013a). On computing minimal correction subsets. In *IJCAI*, pp. 615–622.

Marques-Silva, J., Janota, M., & Belov, A. (2013b). Minimal sets over monotone predicates in Boolean formulae. In *CAV*, pp. 592–607.

Mencía, C., Previti, A., & Marques-Silva, J. (2015). Literal-based MCS extraction. In *IJCAI*, pp. 1973–1979.

Meyer, T. A., Lee, K., Booth, R., & Pan, J. Z. (2006). Finding maximally satisfiable terminologies for the description logic $\mathcal{ALC}$. In *AAAI*, pp. 269–274.

Minoux, M. (1988). LTUR: A simplified linear-time unit resolution algorithm for Horn formulae and computer implementation. *Inf. Process. Lett.*, *29*(1), 1–12.

Moodley, K., Meyer, T., & Varzinczak, I. J. (2011). Root justifications for ontology repair. In *RR*, pp. 275–280.

Morgado, A., Liffiton, M. H., & Marques-Silva, J. (2012). MaxSAT-based MCS enumeration. In *HVC*, pp. 86–101.

Murakami, K., & Uno, T. (2011). Efficient algorithms for dualizing large-scale hypergraphs. *CoRR*, *abs/1102.3813*.

Nguyen, H. H., Alechina, N., & Logan, B. (2012). Axiom pinpointing using an assumption-based truth maintenance system. In *DL*.

O'Sullivan, B., Papadopoulos, A., Faltings, B., & Pu, P. (2007). Representative explanations for over-constrained problems. In *AAAI*, pp. 323–328.

Parsia, B., Sirin, E., & Kalyanpur, A. (2005). Debugging OWL ontologies. In *WWW*, pp. 633–640.

Peñaloza, R. (2009). *Axiom pinpointing in description logics and beyond*. Ph.D. thesis, Dresden University of Technology.

Peñaloza, R., & Sertkaya, B. (2010). On the complexity of axiom pinpointing in the EL family of description logics. In *KR*.

Peñaloza, R., & Thuluva, A. S. (2015). Iterative ontology updates using context labels. In Papini, O., Benferhat, S., Garcia, L., Mugnier, M., Fermé, E. L., Meyer, T., Wassermann, R., Hahmann, T., Baclawski, K., Krisnadhi, A., Klinov, P., Borgo, S., Kutz, O., & Porello, D. (Eds.), *Proceedings of the Joint Ontology Workshops 2015*, Vol. 1517 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Previti, A., & Marques-Silva, J. (2013). Partial MUS enumeration. In *AAAI*, pp. 818–825.

Rector, A. L., & Horrocks, I. R. (1997). Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Workshop on Ontological Engineering*, pp. 414–418.

Reiter, R. (1987). A theory of diagnosis from first principles. *Artif. Intell.*, *32*(1), 57–95.

Rosa, E. D., & Giunchiglia, E. (2013). Combining approaches for solving satisfiability problems with qualitative preferences. *AI Commun.*, *26*(4), 395–408.

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

Schlobach, S., Huang, Z., Cornet, R., & van Harmelen, F. (2007). Debugging incoherent terminologies. *J. Autom. Reasoning*, *39*(3), 317–349.

Sebastiani, R., & Vescovi, M. (2009). Axiom pinpointing in lightweight description logics via Horn-SAT encoding and conflict analysis. In *CADE*, pp. 84–99.

Sebastiani, R., & Vescovi, M. (2015). Axiom pinpointing in large $\mathcal{EL}^+$ ontologies via SAT and SMT techniques. Tech. rep. DISI-15-010, DISI, University of Trento, Italy. Under Journal Submission. Available as http://disi.unitn.it/~rseba/elsat/elsat_techrep.pdf.

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. *J. Biomed. Inform.*, *40*(1), 30–43.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *J. Web Sem.*, *5*(2), 51–53.

Slaney, J. (2014). Set-theoretic duality: A fundamental feature of combinatorial optimisation. In *ECAI*, pp. 843–848.

Spackman, K. A., Campbell, K. E., & Côté, R. A. (1997). SNOMED RT: a reference terminology for health care. In *AMIA*.

Vescovi, M. (2011). *Exploiting SAT and SMT Techniques for Automated Reasoning and Ontology Manipulation in Description Logics*. Ph.D. thesis, University of Trento.