Debugging $\mathcal{EL}^+$ Ontologies through Horn MUS Enumeration

Alexey Ignatiev[1], Joao Marques-Silva[1], Carlos Mencía[2], and Rafael Peñaloza[3]

[1] University of Lisbon, Portugal ({aignatiev,jpms}@ciencias.ulisboa.pt)
[2] University of Oviedo, Spain (cmencia@gmail.com)
[3] Free University of Bozen-Bolzano, Italy (penaloza@inf.unibz.it)

Abstract. In description logics (DLs), axiom pinpointing refers to the problem of enumerating the minimal subsets of axioms from an ontology that entail a given consequence. Recent developments on axiom pinpointing for the light-weight DL $\mathcal{EL}^+$ are based on translating this problem into the enumeration of all minimally unsatisfiable subformulas (MUSes) of a propositional formula, and using advanced SAT-based techniques for solving it. Further optimizations have been obtained by targeting the MUS enumerator to the specific properties of the formula obtained.
In this paper we describe different improvements that have been considered since the translation was first proposed. Through an empirical study, we analyse the behaviour of these techniques and how it depends on different characteristics of the original pinpointing problem, and the translated SAT formula.

1 Introduction

Description logics, in particular those from the $\mathcal{EL}$ family, have been successfully used to represent the knowledge of many application domains, forming large ontologies. And their popularity is continuously increasing. Ontology development, however, is very error-prone and it is not uncommon to find unwanted (or unexpected) consequences being entailed. To understand and correct the causes of these consequences on existing ontologies—some of which have hundreds of thousands of axioms—without the help of any automated tools would be impossible. Axiom pinpointing refers to the task of finding the precise axioms in an ontology that cause a consequence to follow [28], or that need to be corrected for avoiding it.

Recent years have witnessed remarkable improvements in axiom pinpointing technologies, specially for logics in the $\mathcal{EL}$ family [2,3,6–8,22,23,36,37]. Among these, the use of SAT-based methods [2,3,36] has been shown to outperform other alternative approaches very significantly. These methods reduce the axiom pinpointing problem to a propositional Horn formula, and apply highly-optimized SAT tools, along with some *ad-hoc* optimizations to enumerate all the (minimal) subontologies that entail the consequence. Recently, it was shown that the use of techniques developed specifically to handle Horn formulas can further improve the performance of these methods [1].

In this paper, we propose an integrated tool for analysing ontologies that integrates different methods for solving axiom pinpointing and other related problems. Specifically, we show that by considering the shape of the Horn formula obtained, we can explain and repair consequences from ontologies, and also find justifications of a desired size, among other tasks. An experimental analysis shows the behaviour of these techniques, and how it depends on the characteristics of the input problem, and in particular of the output it produces.

2 Preliminaries

We assume some basic knowledge of the DL $\mathcal{EL}^+$ [5] and propositional logic [10], but we briefly recall the main notions needed for this paper.

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

In the DL $\mathcal{EL}^+$, *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 knowledge of the domain is stored in a *TBox*: a finite set of *general concept inclusions* (GCIs) of the form $C \sqsubseteq D$, where C and D are $\mathcal{EL}^+$ concepts, and *role inclusions* (RIs) of the form $r_1 \circ \cdots \circ r_n \sqsubseteq s$, 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.

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 $A \in \mathsf{N_C}$ to a set $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ and every $r \in \mathsf{N_R}$ to a binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$. 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 composition of binary relations. $\mathcal{I}$ is a *model* of $\mathcal{T}$ iff $\mathcal{I}$ satisfies all its GCIs and RIs.

The main reasoning problem in $\mathcal{EL}^+$ is to decide subsumption between concepts. A concept C is *subsumed* by D w.r.t. $\mathcal{T}$ (denoted $C \sqsubseteq_\mathcal{T} D$) if for every model $\mathcal{I}$ of $\mathcal{T}$ it holds that $C^\mathcal{I} \subseteq D^\mathcal{I}$. *Classification* refers to the task of deciding all the subsumption relations between concept names appearing in $\mathcal{T}$. 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). *A* 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 $C \not\sqsubseteq_{\mathcal{T} \setminus \mathcal{D}} D$.*[4]

It is well known that MinAs and diagnoses are closely related by minimal hitting set duality [21,35]. More precisely, the minimal hitting sets of the set of all MinAs is exactly the set of all repairs, and *vice versa*.

[4] MinAs are also often called *justifications* in the literature [17,34].

Example 2. Consider the TBox $\mathcal{T}_{\mathsf{exa}} = \{A \sqsubseteq \exists r.A, A \sqsubseteq Y, \exists r.Y \sqsubseteq B, Y \sqsubseteq B\}$. There are two MinAs for $A \sqsubseteq B$ w.r.t. $\mathcal{T}_{\mathsf{exa}}$, 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 relation are $\{A \sqsubseteq Y\}$, $\{A \sqsubseteq \exists r.A, Y \sqsubseteq B\}$, and $\{\exists r.Y \sqsubseteq B, Y \sqsubseteq B\}$.

2.2 Propositional Satisfiability

In propositional logic, we consider a set of Boolean (or propositional) *variables* X. A *literal* is either a variable $x \in X$ or its negation ($\neg x$). The former are called *positive* literals, and the latter are *negative*. A finite disjunction of literals is called a *clause*. Finally, a CNF formula (or *formula* for short) is a finite conjunction of clauses. One special class of formulas are *Horn formulas*. These are CNF formulas whose clauses contain at most one positive literal.

Clauses and formulas are interpreted via truth assignments. A *truth assignment* is a mapping μ: $X \to \{0, 1\}$. This truth assignment is extended to literals, clauses, and formulas in the obvious manner. If μ satisfies the formula $\mathcal{F}$, then μ is called a *model* of $\mathcal{F}$. *Propositional satisfiability* refers to the problem of deciding whether a given formula $\mathcal{F}$ has a model or not. If it does not have a model, then $\mathcal{F}$ is called *unsatisfiable*. As it is well known, this problem is in general NP-complete. However, when considering only Horn formulas, satisfiability is decidable in polynomial time [13, 16, 27].

Just as in the case of description logics, one is sometimes interested in understanding (and correcting) the causes for unsatisfiability of a formula. For this reason, the following subsets are usually considered [21, 25].

Definition 3 (MUS, MCS). *Let $\mathcal{F}$ be an unsatisfiable formula. A subformula $\mathcal{M} \subseteq \mathcal{F}$ is called* minimally unsatisfiable subset *(MUS) of $\mathcal{F}$ iff $\mathcal{M}$ is unsatisfiable and for all $c \in \mathcal{M}, \mathcal{M} \setminus \{c\}$ is satisfiable. The formula $\mathcal{C} \subseteq \mathcal{F}$ is a* minimal correction subset *(MCS) iff $\mathcal{F} \setminus \mathcal{C}$ is satisfiable and for all $c \in \mathcal{C}, \mathcal{F} \setminus (\mathcal{C} \setminus \{c\})$ is unsatisfiable.*

MUSes and MCSes are related by hitting set duality [9, 11, 32, 39]. Notice that MUSes and MCSes are closely related to MinAs and diagnoses from Definition 1, respectively. Indeed, although $\mathcal{EL}^+$ and propositional logic differ in expressivity and in the inferences of interest, in both cases the goal is to find minimal information that explains, or removes, the inference.

A generalization of the notion of a MUS is that of a *group-MUS* [21]. In this case, the clauses of the unsatisfiable formula $\mathcal{F}$ are partitioned into groups, and the goal is not to find the specific clauses that cause unsatisfiability, but the groups to which they belong. This notion is formalized next.

Definition 4 (Group-MUS). *Given an explicitly partitioned unsatisfiable CNF formula $\mathcal{F} = \mathcal{G}_0 \cup ... \cup \mathcal{G}_k$, a* group-MUS *of $\mathcal{F}$ is a set of groups $\mathfrak{G} \subseteq \{\mathcal{G}_1, ..., \mathcal{G}_k\}$, such that $\mathcal{G}_0 \cup \bigcup_{\mathcal{G}_i \in \mathfrak{G}} \mathcal{G}_i$ is unsatisfiable, and for every $\mathcal{G}_j \in \mathfrak{G}$, $\mathcal{G}_0 \cup \bigcup_{\mathcal{G}_i \in \mathfrak{G}, i \neq j} \mathcal{G}_i$ is satisfiable.*

Notice that MUSes are a special cases of group-MUSes in which the group $\mathcal{G}_0$ is empty, and all other groups are singletons, containing a clauses from $\mathcal{F}$. One can also define the generalization of MCSes to *group-MCSes* in the obvious way. In the following, we will often call a formula whose clauses have been partitioned in groups as in Definition 4 a *group formula*.

During the last years, highly-optimized methods for the enumeration of (group-)MUSes of propositional formulas have been developed and implemented. Taking advantage of these developments, it has been proposed to translate the problem of enumerating MinAs and diagnoses in DLs—and in particular in $\mathcal{EL}^+$—to MUS and MCS enumeration in propositional logic. In the following section we briefly recall the basic ideas of this translation and present a few further insights for improving the overall efficiency of MinA enumeration.

3 The SAT Encoding

The main problem we consider in this section is the enumeration of the MinAs and diagnoses for a given subsumption relation w.r.t. an $\mathcal{EL}^+$ TBox $\mathcal{T}$. Our approach consists of three main components: The first one *classifies* the TBox $\mathcal{T}$ and encodes the classification procedure into a set of Horn clauses $\mathcal{H}$. Given a subsumption relation entailed by $\mathcal{T}$, which we aim to analyze, the second component creates and simplifies an unsatisfiable Horn formula, and partitions it in a suitable manner to reduce the DL enumeration problems into group-MUS and group-MCS enumerations. Finally, the third component computes group-MUSes and group-MCSes, corresponding to MinAs and diagnoses, respectively. Each of its components is explained in more detail next.

3.1 Classification and Horn Encoding

During the classification of $\mathcal{T}$, a Horn formula $\mathcal{H}$ is created according to the method introduced in EL$^+$SAT [36, 37]. To this end, each axiom $a_i \in \mathcal{T}$ is initially assigned a unique selector variable $s_{[a_i]}$. The classification of $\mathcal{T}$ is done in two phases (see [5, 7] for more details).

First, $\mathcal{T}$ is normalized so that it only contains GCIs of the forms

$$(A_1 \sqcap ... \sqcap A_k) \sqsubseteq B \qquad A \sqsubseteq \exists r.B \qquad \exists r.A \sqsubseteq B,$$

where $A, A_i, B \in \mathsf{N_C} \cup \{\top\}$ and $k \geq 1$, and RIs of the form

$$r_1 \circ ... \circ r_n \sqsubseteq s$$

with $r, r_i, s \in \mathsf{N_R}$ and $n \geq 1$. The normalization process runs in linear time and 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 clauses $s_{[a_i]} \rightarrow s_{[a_{ik}]}$, with $1 \leq k \leq m_i$, are added to the Horn formula $\mathcal{H}$.

Then, the normalized TBox $\mathcal{T}_\mathcal{N}$ is saturated through the exhaustive application of the *completion rules* shown in Table 1, resulting in the extended TBox $\mathcal{T}'$.

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

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

Each of the rows in Table 1 constitute a so-called completion rule. Their application is sound and complete for inferring (atomic) subsumptions [5]. Whenever a rule r can be applied (with antecedents $\mathsf{ant}(r)$) leading to inferring an axiom a_i, the Horn clause $(\bigwedge_{\{a_j \in \mathsf{ant}(r)\}} s_{[a_j]}) \rightarrow s_{[a_i]}$ is added to $\mathcal{H}$.

The completion algorithm—and hence the construction of the formula $\mathcal{H}$—terminates after polynomially many rule applications. The result of this construction is a Horn formula that encodes all possible derivations that can be obtained through applications of the completion algorithm, to infer any atomic subsumption relation; that is, any entailment $X \sqsubseteq_{\mathcal{T}} Y$, with $X, Y \in \mathsf{N_C}$.

3.2 Generation of Group Horn Formulas

After classifying $\mathcal{T}$, the extended TBox $\mathcal{T}'$ contains all atomic subsumptions that can be derived from $\mathcal{T}$. For a given such entailment $A \sqsubseteq B \in \mathcal{T}'$, we might be interested in computing their MinAs or diagnoses. Each of these *queries* will result in a group Horn formula defined as: $\mathcal{H}_G = \{\mathcal{G}_0, \mathcal{G}_1, ..., \mathcal{G}_{|\mathcal{T}|}\}$, where $\mathcal{G}_0 = \mathcal{H} \cup \{(\neg s_{[A \sqsubseteq B]})\}$ and for each axiom a_i $(i > 0)$ in the original TBox $\mathcal{T}$, the group $\mathcal{G}_i = \{(s_{[a_i]})\}$ is defined with a single unit clause defined by the selector variable of the axiom. By construction, $\mathcal{H}_G$ is unsatisfiable. Moreover, its group-MUSes correspond to the MinAs for $A \sqsubseteq_{\mathcal{T}} B$ (see [2,3] for the full details). Equivalently, due to the hitting set duality between MinAs and diagnoses, which also holds for group-MUSes and group-MCSes, the group-MCSes of $\mathcal{H}_G$ correspond to the diagnoses for $A \sqsubseteq_{\mathcal{T}} B$.

To improve the efficiency of these enumeration problems, the formula $\mathcal{H}_G$ is often simplified through different techniques. In particular, two simplification techniques based on syntactic modularization were proposed in [36, 37]. These techniques have been shown to reduce the size of the formulas to a great extent.

3.3 Computation of Group-MUSes and Group-MCSes

The last step in the process is to enumerate all the group-MUSes or group-MCSes of the formula $\mathcal{H}_G$ constructed in Section 3.2. Previous work has proposed the use of a general purpose MUS enumerator, or other techniques focused on propositional formulas, together with some *ad-hoc* optimizations [23, 24, 36, 37]. In contrast, we exploit the fact that $\mathcal{H}_G$ is a Horn formula, which can be treated

Algorithm 1: EMUS [30] / MARCO [20]

Input: $\mathcal{F}$ a CNF formula
Output: Reports the set of MUSes (and MCSes) of $\mathcal{F}$

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

more efficiently via optimized unit propagation techniques. Thus, we enumerate group-MUSes and group-MCSes using the state-of-the-art HGMUS enumerator [3]. HGMUS exploits hitting set dualization between (group-) MCSes and (group-) MUSes and, hence, it shares ideas also explored in MaxHS [12], EMUS/MARCO [19,30], and many other systems.

As shown in Algorithm 1, these methods rely on a two (SAT) solvers approach. The formula $\mathcal{Q}$ is defined over a set of selector variables corresponding to the clauses in $\mathcal{F}$. This formula is used to enumerate subsets of $\mathcal{F}$. Iteratively, the algorithm computes a maximal model P of $\mathcal{Q}$ and tests whether the subformula $\mathcal{F}' \subseteq \mathcal{F}$ containing the clauses associated to P is satisfiable. If this formula is satisfiable, then $\mathcal{F} \setminus \mathcal{F}'$ is an MCS of $\mathcal{F}$. Otherwise, $\mathcal{F}'$ can be reduced to an MUS, and the result of this reduction is reported. To avoid observing the same MUS or MCS in a later iteration, all such sets found are blocked by adding the respective clauses to $\mathcal{Q}$.

HGMUS shares the main organization of Algorithm 1, with $\mathcal{F} = \mathcal{G}_0$ and $\mathcal{Q}$ defined over selector variables for groups $\mathcal{G}_i$ of $\mathcal{H}_G$, with $i > 0$. However, it also includes some specific features for handling Horn formulas more efficiently. First, it uses *linear time unit resolution* (LTUR) [27] to check satisfiability of the formula in linear time. Additionally, HGMUS integrates a dedicated insertion-based MUS extractor as well as an efficient algorithm for computing maximal models. The latter is based on a recently proposed reduction from maximal model computation to MCSes computation [26].

3.4 Additional Features

Although the main goal of this work is to enumerate the MinAs and diagnoses of a given consequence, it is important to notice that the construction of the

Horn formula $\mathcal{H}_G$ described earlier in this section can be used, together with other advanced techniques from propositional satisfiability, to provide additional services to axiom pinpointing. We describe some of these next.

Diagnosing Multiple Subsumption Relations Simultaneously Definition 1 considers only one subsumption relation that needs to be understood or removed. However, in a typical knowledge engineering workflow one will often be interested in looking at several consequences simultaneously. For example, if several errors are detected, then one wants to find a maximal subset of the TBox that does not imply any of those errors. Once that the formula $\mathcal{H}_G$ is constructed, it is possible to look at several atomic subsumptions as follows. Given a set of atomic subsumptions $A_i \sqsubseteq B_i \in \mathcal{T}'$, $1 \leq i \leq n$, one needs simply to add all the unit clauses $(\neg s_{[A_i \sqsubseteq B_i]})$ to $\mathcal{G}_0$ in $\mathcal{H}_G$. In this case, the formula becomes unsatisfiable as soon as *any* of the atomic subsumptions is derivable. Thus, a group-MCS for this formula corresponds to a diagnosis that eliminates all these subsumption relations. Analogously, a group-MUS corresponds to axioms that entail at least one of these consequences.

Computing Smallest MinAs Alternatively to enumerating all the possible MinAs, one may want to compute only those of the minimum possible size. This would be the case, for instance, when the MinA will be reviewed by a human expert, and retrieving large subsets of the TBox $\mathcal{T}$ would make the task of understanding them harder. To enable this functionality, one can simply integrate a state-of-the-art solver for the smallest MUS (SMUS) problem such as FORQES [15]. Notice that the decision version of the SMUS problem is known to be Σ_2^{p}-complete for general CNF formulas [14,18], but this complexity bound is lowered to NP-completeness for Horn formulas [7,28]. As HGMUS, FORQES is based on the hitting set dualization between (group) MUSes and (group) MCSes. The tool iteratively computes *minimum* hitting sets of a set of MCSes of a formula detected so far. While these minimum hitting sets are satisfiable, they are grown into a maximal satisfiable subformula (MSS), whose complement is an MCS which is added to the set of MCSes. The process terminates when an unsatisfiable minimum hitting set is identified, representing a smallest MUS of the formula.

All these features have been implemented in the system BEACON [1], which is available at `http://logos.ucd.ie/web/doku.php?id=beacon-tool`. The performance of this system has been analyzed, in comparison to other existing axiom pinpointing tools, in previous work [1]. In the following section we perform an empirical analysis aiming at understanding the behaviour of BEACON in relation to different characteristics of the input problem and the propositional formula obtained.

Table 2: Summary of Instances

		Full-Galen	Not-Galen	GO	NCI	Snomed-CT
full	# clauses	3880668	148104	294783	342826	36530904
	# axioms	36544	4379	20466	46800	310024
COI	# clauses (avg)	637680	17888	3900	7693	3722424
	# clauses (max)	1234308	35787	40857	44294	11325511
	# axioms (avg)	118	24	13	15	80
	# axioms (max)	242	78	30	43	199
x2	# clauses (avg)	3834	378	109	67	6261
	# clauses (max)	10076	1580	348	314	33643
	# axioms (avg)	118	24	13	15	80
	# axioms (max)	242	78	30	43	199

4 Experimental Results

In this section, we present some experimental results aimed to understanding the properties of our approach and its behaviour on different kinds of instances. To this end, we took the instances originally proposed by Sebastiani and Vescovi [37], which have become *de facto* benchmarks for axiom pinpointing tools in $\mathcal{EL}^+$. The experimental setup considers 500 subsumption relations that follow from five well-known $\mathcal{EL}^+$ bio-medical ontologies. These are GALEN [31] (FULL-GALEN and NOT-GALEN), the Gene Ontology (GO) [4], NCI [38], and SNOMED-CT [40]. Specifically, for each of these ontologies, 100 subusmption relations were chosen: 50 were randomly selected from the space of all entailed subsumption relations, and 50 were selected as those that appear most often in the head of a Horn clause in the encoding. This choice was made as a heuristic for instances that should contain more MinAs and be harder to solve (see [37]) for the full details.

For each of these 500 instances, we considered three variants: the full formula, as obtained through the construction described in Section 3, and the smaller formulas obtained after applying the *cone of influence* (COI) and *x2* optimizations from [37]. Intuitively, the COI technique traverses the formula backwards, considering the clauses containing assertions that were used for deriving the queried subsumption relation. The COI module consists of the set of (original) axioms that were found in the computation of the COI formula. The x2 technique encodes the subontology represented by the COI module, which usually results in a smaller Horn formula, containing the same axiom variables. Overall, this gives us a corpus of 1500 Horn formulas of varying size and structure, and providing a large range of hardness (see Table 2). For our experiments, all these formulas were fed to the back-end engine used by BEACON; that is, to HgMUS. The experiments were run on a Linux cluster (2Ghz) with a time limit of 3600s and 4Gbytes of memory.

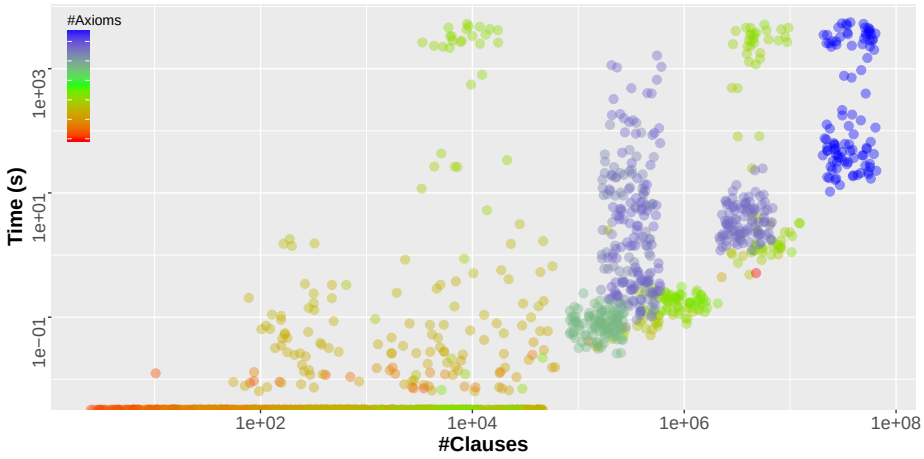

Fig. 1: Plot comparing the size of the Horn formula (x-axis) and the number of axioms (gradient) to the time needed to solve an instance. Notice that all scales are logarithmic.

As explained before, our goal is not to compare the performance of HgMUS against other proposed approaches. Such a comparison can be found in previous work [1,3]. Likewise, the effect of the two proposed optimizations (COI and x2) has been analysed in detail in [37] (albeit, in a different system). One conclusion obtained from the latter analysis is that the COI optimization improves the results over the full formula, and x2 performs better than COI.

A simple analysis shows that the same behaviour is observed when HgMUS is used as the underlying Horn MUS enumeration tool. One possible explanation for the improvements obtained through COI and x2 is that these optimizations produce smaller formulas with smaller axioms, which are easier to handle by HgMUS. Recall, in addition, that ontology size is a very good predictor for hardness of inferences in DLs [33]. Through Figure 1, we observe that this intuition is correct, but there are other factors influencing the performance of the enumerator. The figure compares the time required to solve an instance (y-axis) against the size of the formula and number of axioms used in that instance (shown in the gradient). While it is true that time tends to increase as either of these factors grows, the correlation is not very high. See for example the cluster of instances with over 50,000 clauses and 1000 axioms that are solved in no time (bottom line of the figure).

A better predictor for the time needed to find all MinAs seems to be the size of the *output*. As seen in Figure 2, the time required to solve an instance increases with both, the number of MinAs found in that instance (shown through the gradient, where a colder color means a larger number of MinAs), and the average size of the MinAs found (shown through the size of the dots in the plot). Notice that there is a large concentration of big, cold dots at the top line of the

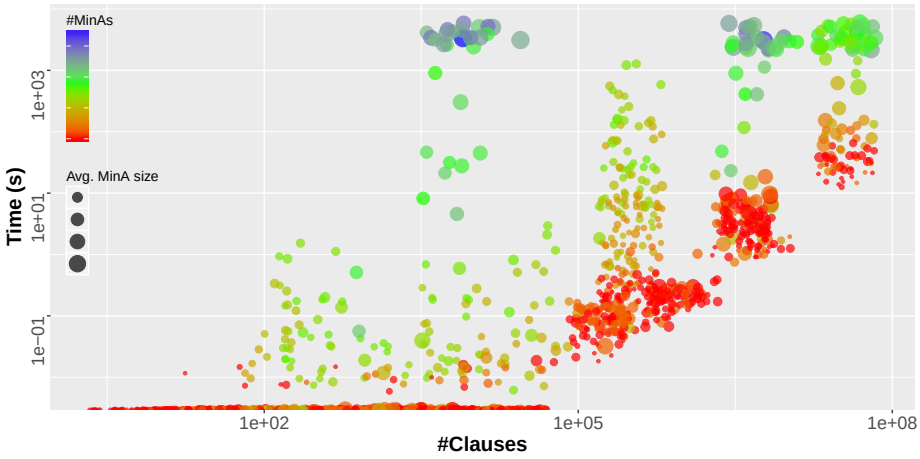

Fig. 2: Plot comparing the size of the Horn formula (x-axis), the number of MinAs (gradient), and the average MinA size (size) to the time needed to solve an instance. Notice that the scales of the axes are logarithmic.

plot. Some of these correspond to the 73 instances that timed-out. Interestingly, in one instance we were able to enumerate over 3200 MinAs before the allocated time of 3600s. was spent. For comparison, notice that the average number of MinAs found over all the instances is below 20, and when restricted to only those instances fully solved, this average drops down to 6.5.

Finally, we verify whether the theoretical *output-complexity hardness* of MinA enumeration is observed also in practice. In a nutshell, it is known that as more MinAs are found, it becomes harder to find a new one, or to decide whether no additional solutions exist [29]. As shown in Figure 3, the maximum delay between solutions increases with the number of MinAs. However, the relation to the average delay is not so direct.

In order to understand these relationships in more detail, we will need to design experiments aimed at finding the differentiating properties of the hard and simple instances observed. In addition, we will need to develop better optimizations that will allow us to fully solve the missing instances, at least over the reduced formulas. Both of these steps will be the focus of future work.

5 Conclusions

In this paper, we have presented a new approach for enumerating MinAs in the light-weight DL $\mathcal{EL}^+$ through the enumeration of Horn group-MUSes. The approach is based on a previously explored translation from $\mathcal{EL}^+$ to Horn formulas. One of the differentiating features of our proposal is that it uses a dedicated Horn enumeration tool, which is able to exploit the linear time unit resolution algorithm available for this logic.

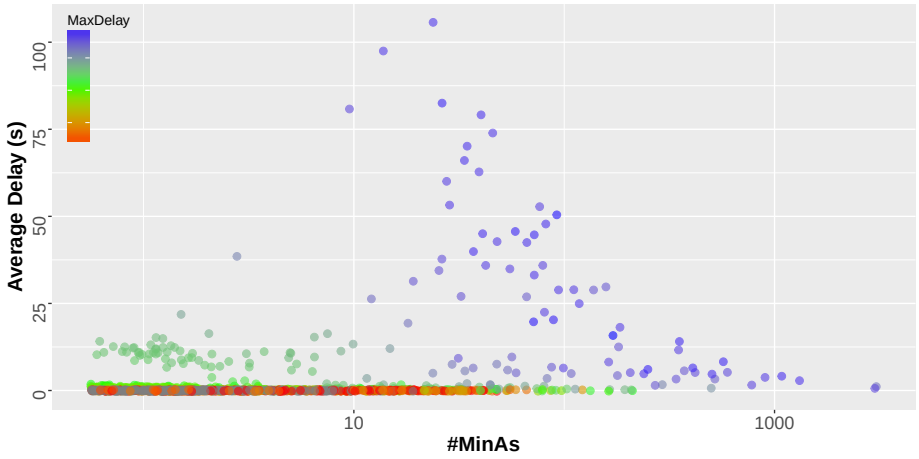

Fig. 3: Plot comparing the number of MinAs (x-axis) to the average and maximum delay observed between solutions.

By using the properties of propositional Horn formulas, we show that it is possible not only to efficiently enumerate all MinAs for large $\mathcal{EL}^+$ ontologies, but also solve other associated problems, like repairing an error, or finding justifications of minimal size. Interestingly, the effectiveness of our methods depends only on the existence of a Horn formula, and not on the properties of $\mathcal{EL}^+$; thus, it should be possible to produce efficient axiom pinpointing and repair methods for other DLs as well.

Through an empirical evaluation, we observe that the size of the output is an important contributing factor to the total time spent by our method. Since we cannot avoid generating this output, it is unclear how our methods can be improved to avoid such a bottleneck. However, we plan to further analyse the behaviour of the MinA enumeration, and extend it with an analysis of the other related reasoning tasks, to identify potential improvements, or cases that can be solved easily.

References

1. M. F. Arif, C. Mencía, A. Ignatiev, N. Manthey, R. Peñaloza, and J. Marques-Silva. BEACON: an efficient sat-based tool for debugging EL+ ontologies. In N. Creignou and D. L. Berre, editors, *Proceedings of the 19th International Conference on Theory and Applications of Satisfiability Testing (SAT 2016)*, volume 9710 of *Lecture Notes in Computer Science*, pages 521–530. Springer, 2016.
2. M. F. Arif, C. Mencía, and J. Marques-Silva. Efficient axiom pinpointing with EL2MCS. In *KI*, pages 225–233, 2015.
3. M. F. Arif, C. Mencía, and J. Marques-Silva. Efficient MUS enumeration of Horn formulae with applications to axiom pinpointing. In *SAT*, pages 324–342, 2015.

4. M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, and et al. Gene ontology: tool for the unification of biology. *Nat. Genet.*, 25(1):25–29, 2000.
5. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *IJCAI*, pages 364–369, 2005.
6. F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL - A polynomial-time reasoner for life science ontologies. In *IJCAR*, pages 287–291, 2006.
7. F. Baader, R. Peñaloza, and B. Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}^+$. In *KI*, pages 52–67, 2007.
8. F. Baader and B. Suntisrivaraporn. Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}^+$. In *KR-MED*, 2008.
9. J. Bailey and P. J. Stuckey. Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In *PADL*, pages 174–186, 2005.
10. A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.
11. E. Birnbaum and E. L. Lozinskii. Consistent subsets of inconsistent systems: structure and behaviour. *J. Exp. Theor. Artif. Intell.*, 15(1):25–46, 2003.
12. J. Davies and F. Bacchus. Solving MAXSAT by solving a sequence of simpler SAT instances. In *CP*, pages 225–239, 2011.
13. W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *J. Log. Program.*, 1(3):267–284, 1984.
14. A. Gupta. *Learning Abstractions for Model Checking*. PhD thesis, Carnegie Mellon University, June 2006.
15. A. Ignatiev, A. Previti, M. Liffiton, and J. Marques-Silva. Smallest MUS extraction with minimal hitting set dualization. In *CP*, 2015.
16. A. Itai and J. A. Makowsky. Unification as a complexity measure for logic programming. *J. Log. Program.*, 4(2):105–117, 1987.
17. A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all justifications of OWL DL entailments. In *ISWC*, pages 267–280, 2007.
18. P. Liberatore. Redundancy in logic I: CNF propositional formulae. *Artif. Intell.*, 163(2):203–232, 2005.
19. M. H. Liffiton and A. Malik. Enumerating infeasibility: Finding multiple MUSes quickly. In *CPAIOR*, pages 160–175, 2013.
20. M. H. Liffiton, A. Previti, A. Malik, and J. Marques-Silva. Fast, flexible MUs enumeration. *Constraints*, 2015. Online version: http://link.springer.com/article/10.1007/s10601-015-9183-0.
21. M. H. Liffiton and K. A. Sakallah. Algorithms for computing minimal unsatisfiable subsets of constraints. *J. Autom. Reasoning*, 40(1):1–33, 2008.
22. M. Ludwig. Just: a tool for computing justifications w.r.t. ELH ontologies. In *ORE*, 2014.
23. N. Manthey and R. Peñaloza. Exploiting SAT technology for axiom pinpointing. Technical Report LTCS 15-05, Chair of Automata Theory, Institute of Theoretical Computer Science, Technische Universität Dresden, April 2015. Available from https://ddll.inf.tu-dresden.de/web/Techreport3010.
24. N. Manthey, R. Peñaloza, and S. Rudolph. Efficient axiom pinpointing in EL using SAT technology. In M. Lenzerini and R. Peñaloza, editors, *Proceedings of the 29th International Workshop on Description Logics, (DL 2016)*, volume 1577 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016.
25. J. Marques-Silva, F. Heras, M. Janota, A. Previti, and A. Belov. On computing minimal correction subsets. In *IJCAI*, pages 615–622, 2013.

26. C. Mencía, A. Previti, and J. Marques-Silva. Literal-based MCS extraction. In *IJCAI*, pages 1973–1979, 2015.
27. M. Minoux. LTUR: A simplified linear-time unit resolution algorithm for Horn formulae and computer implementation. *Inf. Process. Lett.*, 29(1):1–12, 1988.
28. R. Peñaloza. *Axiom pinpointing in description logics and beyond*. PhD thesis, Dresden University of Technology, 2009.
29. R. Peñaloza and B. Sertkaya. On the complexity of axiom pinpointing in the EL family of description logics. In *KR*, 2010.
30. A. Previti and J. Marques-Silva. Partial MUS enumeration. In *AAAI*, pages 818–825, 2013.
31. A. L. Rector and I. R. Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Workshop on Ontological Engineering*, pages 414–418, 1997.
32. R. Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.
33. V. Sazonau, U. Sattler, and G. Brown. Predicting performance of OWL reasoners: Locally or globally? In C. Baral, G. D. Giacomo, and T. Eiter, editors, *Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2014)*. AAAI Press, 2014.
34. S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. pages 355–362. Morgan Kaufmann, 2003.
35. S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *IJCAI*, pages 355–362, 2003.
36. R. Sebastiani and M. Vescovi. Axiom pinpointing in lightweight description logics via Horn-SAT encoding and conflict analysis. In *CADE*, pages 84–99, 2009.
37. R. Sebastiani and M. Vescovi. Axiom pinpointing in large $\mathcal{EL}^+$ ontologies via SAT and SMT techniques. Technical Report DISI-15-010, DISI, University of Trento, Italy, April 2015. Under Journal Submission. Available as http://disi.unitn.it/~rseba/elsat/elsat_techrep.pdf.
38. N. Sioutos, S. de Coronado, M. W. Haber, F. W. Hartel, W. Shaiu, and L. W. Wright. NCI thesaurus: A semantic model integrating cancer-related clinical and molecular information. *J. Biomed. Inform.*, 40(1):30–43, 2007.
39. J. Slaney. Set-theoretic duality: A fundamental feature of combinatorial optimisation. In *ECAI*, pages 843–848, 2014.
40. K. A. Spackman, K. E. Campbell, and R. A. Côté. SNOMED RT: a reference terminology for health care. In *AMIA*, 1997.