Pinpointing in the Description Logic $\mathcal{EL}^+$

Franz Baader[1], Rafael Peñaloza[2,*], and Boontawee Suntisrivaraporn[1,**]

[1] Theoretical Computer Science, TU Dresden, Germany
{baader,meng}@tcs.inf.tu-dresden.de
[2] Intelligent Systems, University of Leipzig, Germany
penaloza@informatik.uni-leipzig.de

Abstract. Axiom pinpointing has been introduced in description logics (DLs) to help the user understand the reasons why consequences hold by computing minimal subsets of the knowledge base that have the consequence in question. Until now, the pinpointing approach has only been applied to the DL $\mathcal{ALC}$ and some of its extensions. This paper considers axiom pinpointing in the less expressive DL $\mathcal{EL}^+$, for which subsumption can be decided in polynomial time. More precisely, we consider an extension of the pinpointing problem where the knowledge base is divided into a *static* part, which is always present, and a *refutable* part, of which subsets are taken. We describe an extension of the subsumption algorithm for $\mathcal{EL}^+$ that can be used to compute all minimal subsets of (the refutable part of) a given TBox that imply a certain subsumption relationship. The worst-case complexity of this algorithm turns out to be exponential. This is not surprising since we can show that a given TBox may have exponentially many such minimal subsets. However, we can also show that the problem is not even output polynomial, i.e., unless P=NP, there cannot be an algorithm computing all such minimal sets that is polynomial in the size of its input *and output*. In addition, we show that finding out whether there is such a minimal subset within a given cardinality bound is an NP-complete problem. In contrast to these negative results, we also show that one such minimal subset can be computed in polynomial time. Finally, we provide some encouraging experimental results regarding the performance of a practical algorithm that computes one (small, but not necessarily minimal) subset that has a given subsumption relation as consequence.

1 Introduction

Description logics (DLs) [2] are a successful family of logic-based knowledge representation formalisms, which can be used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way. They are employed in various application domains, such as natural language processing, configuration, databases, and bio-medical ontologies, but their most notable success so far is the adoption of the DL-based language OWL [15] as standard

* Funded by the German Research Foundation (DFG) under grant GRK 446.
** Funded by the German Research Foundation (DFG) under grant BA 1122/11-1.

J. Hertzberg, M. Beetz, and R. Englert (Eds.): KI 2007, LNAI 4667, pp. 52–67, 2007.

ontology language for the semantic web. For a developer or user of a DL-based ontology, it is often quite hard to understand why a certain consequence holds, and even harder to decide how to change the ontology in case the consequence is unwanted. For example, in the current version of the medical ontology SNOMED [24], the concept *Amputation-of-Finger* is unintendedly classified as a subconcept of *Amputation-of-Arm*. Finding the axioms that are responsible for this among the more than 350,000 terminological axioms of SNOMED without support by an automated reasoning tool is not easy.

As a first step towards providing such support, Schlobach and Cornet [22] describe an algorithm for computing all the *minimal subsets* of a given knowledge base that *have* a given *consequence*. In the following, we call such a set a *minimal axiom set (MinA)*. It helps the user to comprehend why a certain consequence holds. The knowledge bases considered in [22] are so-called unfoldable $\mathcal{ALC}$-terminologies, and the unwanted consequences are the unsatisfiability of concepts. The algorithm is an extension of the known tableau-based satisfiability algorithm for $\mathcal{ALC}$ [23], where labels keep track of which axioms are responsible for an assertion to be generated during the run of the algorithm. The authors also coin the name "axiom pinpointing" for the task of computing these minimal subsets.

The problem of computing MinAs of a DL knowledge base was actually considered earlier in the context of extending DLs by default rules. In [3], Baader and Hollunder solve this problem by introducing a labeled extension of the tableau-based consistency algorithm for $\mathcal{ALC}$-ABoxes [14], which is very similar to the one described later in [22]. The main difference is that the algorithm described in [3] does not directly compute minimal subsets that have a consequence, but rather a monotone Boolean formula whose variables correspond to the axioms of the knowledge bases and whose minimal satisfying valuations correspond to the MinAs. Another difference between [3] and [22] is that in the former paper the ABox is divided into a static and a refutable part, where the elements of the static part are assumed to be always present, and subsets are built only of the refutable part of the ABox.

The approach of Schlobach and Cornet [22] was extended by Parsia et al. [20] to more expressive DLs, and the one of Baader and Hollunder [3] was extended by Meyer et al. [19] to the case of $\mathcal{ALC}$-terminologies with general concept inclusions (GCIs), which are no longer unfoldable. Axiom pinpointing has also been considered in other research areas, though usually not under this name. For example, in the SAT community, people have considered the problem of computing minimally unsatisfiable (and maximally satisfiable) subsets of a set of propositional formulae. The approaches for computing these sets developed there include special purpose algorithms that call a SAT solver as a black box [18, 7], but also algorithms that extend a resolution-based SAT solver directly [9, 26].

Whereas the previous work on pinpointing in DLs considered fairly expressive DLs that contain at least $\mathcal{ALC}$, this work is concerned with pinpointing in the inexpressive DL $\mathcal{EL}^+$, which allows for conjunction, existential restrictions, and

complex role inclusion axioms. There are several good reasons for considering $\mathcal{EL}^+$. First, several bio-medical ontologies such as SNOMED [24], the Gene Ontology [25], and large parts of Galen [21] can be expressed in $\mathcal{EL}^+$. In particular, both SNOMED and Galen require role inclusion axioms. Second, reasoning in $\mathcal{EL}^+$ and some of its extensions remains polynomial even in the presence of GCIs [1], which are required by Galen. Third, $\mathcal{EL}^+$ is the DL currently handled by our reasoner CEL [5, 4], which behaves quite well on very large ontologies such as SNOMED.

Although the polynomial-time subsumption algorithm for $\mathcal{EL}^+$ described in [1, 5] is not tableau-based, the ideas for extending tableau-based algorithms to pinpointing algorithms employed in [3, 22] can also be applied to this algorithm. However, we will see that the normalization phase employed by this algorithm introduces an additional problem. We will also consider the complexity of pinpointing in $\mathcal{EL}^+$. In contrast to the case of $\mathcal{ALC}$, where the subsumption problem is already quite complex (PSpace-complete), subsumption in $\mathcal{EL}^+$ is polynomial, which makes it easier to analyze in how far pinpointing is a source of additional complexity. Not surprisingly, it turns out that there may be exponentially many MinAs, which shows that an algorithm for computing all MinAs needs exponential time in the size of the input TBox. Even worse, we can show that it is not even possible to obtain an algorithm that is polynomial in the size of the input *and the output* (unless P=NP). In addition, even testing whether there is a MinA of cardinality $\leq n$ for a given natural number n is an NP-complete problem. On the positive side, one MinA can always be computed in polynomial time. Finally, we will provide some experimental results regarding the performance of a practical algorithm that computes one (not necessarily minimal) set that has a given consequence.

EI subsumption alg, not only tableaux. Mention in reasoner section.

2 The Description Logic $\mathcal{EL}^+$

In DLs, *concept descriptions* are inductively defined with the help of a set of *constructors*, starting with a set $\mathsf{N_C}$ of *concept names* and a set $\mathsf{N_R}$ of *role names*. $\mathcal{EL}^+$ concept descriptions are formed using the three constructors shown in the upper part of Table 1. An $\mathcal{EL}^+$ *ontology* or *TBox* is a finite set of *general concept inclusion (GCI)* and *role inclusion (RI)* axioms, whose syntax is shown in the lower part of Table 1. The sublanguage of $\mathcal{EL}^+$ that does not allow for RIs is called $\mathcal{EL}$. We will also use the name $\mathcal{HL}$ for the sublanguage of $\mathcal{EL}$ that does not allow for existential restrictions. This name is motivated by the fact that GCIs involving $\mathcal{HL}$ concepts are basically propositional Horn clauses.

The semantics of $\mathcal{EL}^+$ is defined in terms of *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where the domain $\Delta^{\mathcal{I}}$ is a non-empty set of individuals, and the interpretation function $\cdot^{\mathcal{I}}$ maps each concept name $A \in \mathsf{N_C}$ to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and each role name $r \in \mathsf{N_R}$ to a binary relation $r^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is inductively defined, as shown in the semantics column of Table 1. An interpretation $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ if, for each inclusion axiom in $\mathcal{T}$, the conditions given in the semantics column of Table 1 are satisfied.

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

Name	Syntax	Semantics
top	$\top$	$\Delta^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
exists restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
GCI	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
RI	$r_1 \circ \cdots \circ r_n \sqsubseteq s$	$r_1^{\mathcal{I}} \circ \cdots \circ r_n^{\mathcal{I}} \subseteq s^{\mathcal{I}}$

Since there is no constructor in $\mathcal{EL}^+$ that can cause logical inconsistencies, satisfiability of concepts or consistency of the TBox are not interesting inference problems. The main inference problem for $\mathcal{EL}^+$ is the subsumption problem:

Definition 1 (concept subsumption). *Given two $\mathcal{EL}^+$ concept descriptions C, D and an $\mathcal{EL}^+$ TBox $\mathcal{T}$, C is subsumed by D w.r.t. $\mathcal{T}$ (written $C \sqsubseteq_{\mathcal{T}} D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in every model $\mathcal{I}$ of $\mathcal{T}$.*

In the following, we will restrict the attention to subsumption between concept names. This is justified by the fact that subsumption between concept descriptions can be reduced to subsumption between concept names: we have $C \sqsubseteq_{\mathcal{T}} D$ iff $A \sqsubseteq_{\mathcal{T} \cup \{A \sqsubseteq C, D \sqsubseteq B\}} B$ where A, B are new concept names not occurring in C, D and $\mathcal{T}$.

In order to describe our pinpointing algorithm for subsumption in $\mathcal{EL}^+$, we must briefly recall the known polynomial-time subsumption algorithm for $\mathcal{EL}^+$ [1, 5].[1] First, this algorithm transforms a given TBox into a *normal form* where all GCIs have one of the following forms: $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B$, $A \sqsubseteq \exists r.B$, $\exists r.A \sqsubseteq B$, and where all RIs are of the form $r \sqsubseteq s$ or $r \circ r' \sqsubseteq s$, where $r, r', s \in \mathsf{N_R}$, $n \geq 1$, and $A, A_1, \ldots, A_n, B$ are elements of $\mathsf{N_C^\top}$, i.e, concepts names or the top concept $\top$. This transformation can be achieved in linear time using simple transformation rules, which basically break down complex GCIs into simpler ones (see [1] for details).

Given a TBox $\mathcal{T}$ in normal form over the concept names $\mathsf{N_C}$ and role names $\mathsf{N_R}$, *the subsumption algorithm for $\mathcal{EL}^+$* employs completion rules to extend an initial set of assertions until no more assertions can be added. Assertions are of the form (A, B) or (A, r, B) where $A, B \in \mathsf{N_C^\top}$, and $r \in \mathsf{N_R}$. Intuitively, the assertion (A, B) expresses that $A \sqsubseteq_{\mathcal{T}} B$ holds and (A, r, B) expresses that $A \sqsubseteq_{\mathcal{T}} \exists r.B$ holds. The algorithm starts with a set of assertions $\mathcal{A}$ that contains $(A, \top)$ and (A, A) for every concept name A, and then uses the rules shown in Fig. 1 to extend $\mathcal{A}$. Note that such a rule is only applied if it really extends $\mathcal{A}$, i.e., if the assertion added by the rule is not yet contained in $\mathcal{A}$. The following theorem, which is shown in [1], summarizes the important properties of this algorithm.

[1] A polynomial-time subsumption algorithm for $\mathcal{EL}$ with GCIs was first presented in [8], and subsumption in $\mathcal{HL}$ with GCIs is basically the implication problem for propositional Horn clauses, which is known to be solvable in linear time [10].

If $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \in \mathcal{T}$	**and** $\{(X, A_1), \ldots, (X, A_n)\} \subseteq \mathcal{A}$	**then** add (X, B) to $\mathcal{A}$.
If $A \sqsubseteq \exists r.B \in \mathcal{T}$	**and** $(X, A) \in \mathcal{A}$	**then** add (X, r, B) to $\mathcal{A}$.
If $\exists r.A \sqsubseteq B \in \mathcal{T}$	**and** $\{(X, r, Y), (Y, A)\} \subseteq \mathcal{A}$	**then** add (X, B) to $\mathcal{A}$.
If $r \sqsubseteq s \in \mathcal{T}$	**and** $(X, r, Y) \in \mathcal{A}$	**then** add (X, s, Y) to $\mathcal{A}$.
If $r \circ r' \sqsubseteq s \in \mathcal{T}$	**and** $\{(X, r, Y), (Y, r', Z)\} \subseteq \mathcal{A}$	**then** add (X, s, Z) to $\mathcal{A}$.

Fig. 1. Completion rules for subsumption in $\mathcal{EL}^+$

Theorem 1. *Given an $\mathcal{EL}^+$ ontology $\mathcal{T}$ in normal form, the subsumption algorithm terminates in time polynomial in the size of $\mathcal{T}$. After termination, the resulting set of assertions $\mathcal{A}$ satisfies $A \sqsubseteq_{\mathcal{T}} B$ iff $(A, B) \in \mathcal{A}$, for all concept names A, B occurring in $\mathcal{T}$.*

3 A Pinpointing Algorithm for $\mathcal{EL}^+$

In many applications, it makes sense to distinguish two kinds of axioms in an ontology: trusted ones whose correctness is no longer doubted, and refutable ones for which the designer or user of the ontology is not yet sure whether they are correct. For example, if an already well-established ontology is extended, one might view the newly added GCIs as refutable, but trust the GCIs of the existing ontology. From now on, we assume that TBoxes are of the form $\mathcal{T} = (\mathcal{T}_s \uplus \mathcal{T}_r)$, i.e., they are a disjoint union of a *static* TBox $\mathcal{T}_s$ (whose axioms are irrefutable) and a *refutable* TBox $\mathcal{T}_r$.

Definition 2 (MinA). *Let $\mathcal{T} = (\mathcal{T}_s \uplus \mathcal{T}_r)$ be an $\mathcal{EL}^+$ TBox and A, B concept names occurring in it such that $A \sqsubseteq_{\mathcal{T}} B$. Then, a* minimal axiom set (MinA) *for $\mathcal{T}$ w.r.t. $A \sqsubseteq B$ is a subset $\mathcal{S}$ of $\mathcal{T}_r$ such that $A \sqsubseteq_{\mathcal{T}_s \cup \mathcal{S}} B$, but $A \not\sqsubseteq_{\mathcal{T}_s \cup \mathcal{S}'} B$ for all strict subsets $\mathcal{S}' \subset \mathcal{S}$.*

If $\mathcal{T}_s = \emptyset$, then all axioms are assumed to be refutable. This is the case considered, e.g., in [22]. As an example, consider the TBox $\mathcal{T} = (\emptyset \uplus \mathcal{T}_r)$ consisting of the (refutable) GCIs

$$\text{ax}_1\colon\ A \sqsubseteq \exists r.A, \qquad \text{ax}_2\colon\ A \sqsubseteq Y, \qquad \text{ax}_3\colon\ \exists r.Y \sqsubseteq B, \qquad \text{ax}_4\colon\ Y \sqsubseteq B. \quad (1)$$

We have $A \sqsubseteq_{\mathcal{T}} B$, and it is easy to see that $\{\text{ax}_2, \text{ax}_4\}$ and $\{\text{ax}_1, \text{ax}_2, \text{ax}_3\}$ are the MinAs for $\mathcal{T}$ w.r.t $A \sqsubseteq B$.

In the following, we show how the polynomial-time subsumption algorithm presented in the previous section can be modified to a pinpointing algorithm. However, instead of computing the MinAs directly, we follow the approach introduced in [3] that computes a monotone Boolean formula from which the MinAs can be derived. To define this formula, which we will call pinpointing formula in the following, we assume that every refutable axiom $t \in \mathcal{T}_r$ is labeled with a unique propositional variable, $lab(t)$; axioms $t \in \mathcal{T}_s$ are labeled with $lab(t) := \mathtt{t}$

(for truth). Let $lab(\mathcal{T})$ be the set of all propositional variables labeling axioms in $\mathcal{T}_r$. A *monotone Boolean formula* over $lab(\mathcal{T})$ is a Boolean formula using (some of) the variables in $lab(\mathcal{T})$ and only the binary connectives conjunction and disjunction and the nullary connective $\mathtt{t}$ (for truth). As usual, we identify a propositional *valuation* with the set of propositional variables made true by this valuation. For a valuation $\mathcal{V} \subseteq lab(\mathcal{T})$, let $\mathcal{T}_\mathcal{V} := \{t \in \mathcal{T}_r \mid lab(t) \in \mathcal{V}\}$.

Definition 3 (pinpointing formula). *Given an $\mathcal{EL}^+$ TBox $\mathcal{T} = (\mathcal{T}_s \uplus \mathcal{T}_r)$ and concept names A, B occurring in it, the monotone Boolean formula ϕ over $lab(\mathcal{T})$ is a* pinpointing formula *for $\mathcal{T}$ w.r.t $A \sqsubseteq B$ if the following holds for every valuation $\mathcal{V} \subseteq lab(\mathcal{T})$: $A \sqsubseteq_{\mathcal{T}_s \cup \mathcal{T}_\mathcal{V}} B$ iff $\mathcal{V}$ satisfies ϕ.*

In our example, we can take $lab(\mathcal{T}) = \{\mathrm{ax}_1, \ldots, \mathrm{ax}_4\}$ as set of propositional variables. It is easy to see that $\mathrm{ax}_2 \wedge (\mathrm{ax}_4 \vee (\mathrm{ax}_1 \wedge \mathrm{ax}_3))$ is a pinpointing formula for $\mathcal{T}$ w.r.t. $A \sqsubseteq B$.

If we order valuations by set inclusion, then we obviously have the following relation between MinAs and minimal satisfying valuations of the pinpointing formula.

Proposition 1. *Let ϕ be a pinpointing formula for $\mathcal{T}$ w.r.t $A \sqsubseteq B$. Then*

$$\{\mathcal{T}_\mathcal{V} \mid \mathcal{V} \text{ is a minimal valuation satisfying } \phi\}$$

is the set of all MinAs for $\mathcal{T}$ w.r.t $A \sqsubseteq B$.

This shows that it is enough to design an algorithm for computing a pinpointing formula to obtain all MinAs. For example, one possibility is to bring ϕ into disjunctive normal form and then remove disjuncts implying other disjuncts. Note that this may cause an exponential blowup, which means that, in some cases, the pinpointing formula provides us with a compact representation of the set of all MinAs. Also note that this blowup is not really in the size of the pinpointing formula but rather in the number of variables. Thus, if the size of the pinpointing formula is already exponential in the size of the TBox $\mathcal{T}$ (which may well happen), computing all MinAs from it is still "only" exponential in the size of $\mathcal{T}$. More about the complexity of computing all MinAs from a given pinpointing formula can be found in Section 4.

For the moment, let us assume that the TBox $\mathcal{T} = (\mathcal{T}_s \uplus \mathcal{T}_r)$ is already in normal form. Recall that the axioms in $\mathcal{T}_r$ have a unique propositional variable as label, whereas the axioms in $\mathcal{T}_s$ have label $\mathtt{t}$. In *the pinpointing extension* of the subsumption algorithm for $\mathcal{EL}^+$, assertions a are labeled with monotone Boolean formulae $lab(a)$. The initial assertions $(A, \top)$ and (A, A) receive label $\mathtt{t}$. The definition of rule application is modified as follows. Assume that the preconditions of a rule from Fig. 1 are satisfied for the set of assertions $\mathcal{A}$ w.r.t. the TBox $\mathcal{T}$. Let ϕ be the conjunction of the labels of the GCIs from $\mathcal{T}$ and the assertions from $\mathcal{A}$ occurring in the precondition. If the assertion in the consequence of the rule does not yet belong to $\mathcal{A}$, then it is added with label ϕ. If the assertion is already there with label ψ, then its label is changed to $\psi \vee \phi$ if

this formula is not equivalent to ψ; otherwise (i.e., if ϕ implies ψ) the rule is not applied.

It is easy to see that this modified algorithm always terminates, though not necessarily in polynomial time. In fact, there are polynomially many assertions that can be added to $\mathcal{A}$. If the label of an assertion is changed, then the new label is a more general monotone Boolean formula, i.e., it has more models than the original label. Since there are only exponentially many models, the label of a given assertion can be changed only exponentially often. Accordingly, the size of the label of an assertion can grow only exponentially. Equivalence of monotone Boolean formulae is an NP-complete problem. However, given formulae over n propositional variables whose size is exponential in n, equivalence can be tested in time exponential in n. Thus, there are at most exponentially many rule applications, each of which takes at most exponential time. This yields an exponential time bound for the execution of the pinpointing algorithm.

Regarding correctness of the pinpointing algorithm, it is easy to see that the set of assertions $\mathcal{A}$ obtained after termination is identical to the one obtained by the unmodified algorithm. In addition, we can show that, for all assertions $(A, B) \in \mathcal{A}$, the formula $lab((A, B))$ is a pinpointing formula for $\mathcal{T}$ w.r.t $A \sqsubseteq B$.[2]

Theorem 2. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T} = (\mathcal{T}_s \uplus \mathcal{T}_r)$ in normal form, the pinpointing algorithm terminates in time exponential in the size of $\mathcal{T}$. After termination, the resulting set of assertions $\mathcal{A}$ satisfies the following two properties for all concept names A, B occurring in $\mathcal{T}$:*

1. *$A \sqsubseteq_{\mathcal{T}} B$ iff $(A, B) \in \mathcal{A}$, and*
2. *$lab((A, B))$ is a pinpointing formula for $\mathcal{T}$ w.r.t $A \sqsubseteq B$.*

As an example, consider the TBox $\mathcal{T}$ consisting of the refutable GCIs given in (1) and of no irrefutable axioms. The pinpointing algorithm proceeds as follows. Since $\mathrm{ax}_2 : A \sqsubseteq Y \in \mathcal{T}$ and $(A, A) \in \mathcal{A}$ with label $\mathtt{t}$, the assertion (A, Y) is added to $\mathcal{A}$ with label ax_2 (actually with label $\mathrm{ax}_2 \wedge \mathtt{t}$, which is equivalent to ax_2). Since $\mathrm{ax}_1 : A \sqsubseteq \exists r.A \in \mathcal{T}$ and $(A, A) \in \mathcal{A}$ with label $\mathtt{t}$, (A, r, A) is added to $\mathcal{A}$ with label ax_1. Since $\mathrm{ax}_4 : Y \sqsubseteq B \in \mathcal{T}$ and $(A, Y) \in \mathcal{A}$ with label ax_2, (A, B) is added to $\mathcal{A}$ with label $\mathrm{ax}_2 \wedge \mathrm{ax}_4$. Finally, since $\mathrm{ax}_3 : \exists r.Y \sqsubseteq B \in \mathcal{T}$, $(A, Y) \in \mathcal{A}$ with label ax_2, and $(A, r, A) \in \mathcal{A}$ with label ax_1, the label of $(A, B) \in \mathcal{A}$ is modified from $\mathrm{ax}_2 \wedge \mathrm{ax}_4$ to $(\mathrm{ax}_2 \wedge \mathrm{ax}_4) \vee (\mathrm{ax}_1 \wedge \mathrm{ax}_2 \wedge \mathrm{ax}_3)$. This final label of (A, B) is a pinpointing formula for $\mathcal{T}$ w.r.t. $A \sqsubseteq B$.

As described until now, our pinpointing algorithm for $\mathcal{EL}^+$ can only deal with normalized TBoxes, i.e., the pinpointing formula ϕ it yields contains propositional variables corresponding to the normalized GCIs. We now show that the algorithm for computing pinpointing formulae can easily be extended to one dealing also with non-normalized TBoxes. First, note that the relationship between original axioms and normalized axioms is many to many: one axiom in the original TBox can give rise to several axioms in the normalized one, and one axiom in the normalized TBox can come from several axioms in the original

[2] A proof of this fact in a more general setting can be found in [6].

TBox. For example, consider the GCIs $A \sqsubseteq B_1 \sqcap B_2, A \sqsubseteq B_2 \sqcap B_3$, which are normalized to $A \sqsubseteq B_1, A \sqsubseteq B_2, A \sqsubseteq B_3$. Each original GCI gives rise to two normalized ones, and the normalized GCI $A \sqsubseteq B_2$ has two sources, i.e., it is present in the normalized TBox if the first *or* the second original GCI is present in the input TBox.

Now, assume that $\widehat{\mathcal{T}}$ is an unnormalized input TBox, and that $\mathcal{T}$ is the corresponding normalized TBox where we view all axioms in $\mathcal{T}$ as being refutable. Let ϕ be a pinpointing formula for $\mathcal{T}$ w.r.t. $A \sqsubseteq B$, where A, B are concept names occurring in $\widehat{\mathcal{T}}$ (and thus also in $\mathcal{T}$). We can now modify ϕ to a *pinpointing formula for the original TBox* $\widehat{\mathcal{T}}$ as follows. Assume that the refutable axioms in $\widehat{\mathcal{T}}$ are associated with unique propositional variables, and the irrefutable ones in $\widehat{\mathcal{T}}$ with $\mathtt{t}$. Each normalized axiom in $\mathcal{T}$ has a finite number of original axioms as sources. We modify ϕ by replacing the propositional variable for each normalized axiom by the disjunction of the labels of its sources. Note, in particular, that the propositional variable of a normalized axiom that has an irrefutable axiom as source is replaced by a formula that is equivalent to $\mathtt{t}$.

4 The Complexity of Computing All MinAs

In this section we will show several hardness results regarding the computation of all MinAs. We can actually show all of them already for the sublanguage $\mathcal{HL}$ of $\mathcal{EL}^+$. Of course, these results then also hold for $\mathcal{EL}$ and $\mathcal{EL}^+$.

If we want to compute all MinAs, then in the worst case an exponential runtime cannot be avoided since there may be *exponentially many MinAs* for a given TBox. The following example shows that this is already the case for $\mathcal{HL}$ TBoxes.

Example 1. For all $n \geq 1$, the size of the $\mathcal{HL}$ TBox

$$\mathcal{T}_n := \{B_{i-1} \sqsubseteq P_i \sqcap Q_i, P_i \sqsubseteq B_i, Q_i \sqsubseteq B_i \mid 1 \leq i \leq n\}$$

is linear in n, and we have $B_0 \sqsubseteq_{\mathcal{T}_n} B_n$. Assume that all axioms in $\mathcal{T}_n$ are refutable. Then, there are 2^n MinAs for $\mathcal{T}_n$ w.r.t. $B_0 \sqsubseteq B_n$ since, for each $i, 1 \leq i \leq n$, it is enough to have $P_i \sqsubseteq B_i$ or $Q_i \sqsubseteq B_i$ in the set.

In Section 5 we will show that a single MinA can be computed in polynomial time. However, as soon as we want to know more about the properties of the set of *all* MinAs, this cannot be achieved in polynomial time (unless P=NP). For example, determining whether there is a MinA whose cardinality is bounded by a given natural number n is NP-hard.

Theorem 3. *Given an $\mathcal{HL}$ TBox $\mathcal{T} = (\mathcal{T}_s \uplus \mathcal{T}_r)$, concept names A, B occurring in $\mathcal{T}$, and a natural number n, it is NP-complete to decide whether or not there is a MinA for $\mathcal{T}$ w.r.t. $A \sqsubseteq B$ of cardinality $\leq n$. This already holds in the case where $\mathcal{T}_s = \emptyset$.*

Proof. The problem is in NP since one can simply guess a subset $\mathcal{S}$ of $\mathcal{T}_r$ with cardinality n, and then check in polynomial time whether $A \sqsubseteq_{\mathcal{T}_s \cup \mathcal{S}} B$. Clearly, such a set exists iff there is a MinA of cardinality $\leq n$.

NP-hardness can be shown by a reduction of the NP-hard *hitting set problem* [13]: given a collection $S_1, \ldots S_k$ of sets and a natural number n, is there a set S of cardinality $\leq n$ such that $S \cap S_i \neq \emptyset$ for $i = 1, \ldots, k$. Such a set S is called a *hitting set*. In the reduction, we use a concept name P for every element $p \in S_1 \cup \ldots \cup S_n$ as well as the additional concept names $A, B, Q_1, \ldots, Q_k$. Given $S_1 = \{p_{11}, \ldots, p_{1\ell_1}\}, \ldots, S_k = \{p_{k1}, \ldots, p_{k\ell_k}\}$, we define the TBox $\mathcal{T} = (\emptyset \uplus \mathcal{T}_r)$ with

$$\begin{aligned}\mathcal{T}_r := \{&P_{ij} \sqsubseteq Q_i \mid 1 \leq i \leq k, 1 \leq j \leq \ell_i\} \cup \\ &\{A \sqsubseteq P_{ij} \mid 1 \leq i \leq k, 1 \leq j \leq \ell_i\} \cup \{Q_1 \sqcap \ldots \sqcap Q_k \sqsubseteq B\}.\end{aligned}$$

It is easy to see that $S_1, \ldots, S_k$ has a hitting set of cardinality $\leq n$ iff there is a MinA for $\mathcal{T}$ w.r.t. $A \sqsubseteq B$ of cardinality $\leq n + k + 1$. □

Given the fact that a TBox may have exponentially many MinAs, it is clear that it is not possible to enumerate all MinAs in time polynomial in the size of the input. However, in complexity theory one also considers other kinds of complexity measures for the complexity of enumeration problems [16]. One possibility is to ask whether there is an algorithm that enumerates all MinAs in time polynomial in the size of the input *and the output*, i.e., in the size of the TBox and the number of MinAs. We will call such an algorithm *output polynomial*. One advantage of an output polynomial algorithm is that it runs in polynomial time in case there are only polynomially many outputs.

The pinpointing algorithm for $\mathcal{EL}^+$ described in Section 3 uses as a subprocedure the enumeration of all minimal valuations satisfying a given monotone Boolean formula. Unfortunately, already this problem is known not to have an output polynomial solution (unless P=NP). A proof of this fact can be found in the technical report [11]; since this result is not included in the corresponding journal paper [12], we provide our own proof for the sake of completeness.

Theorem 4. *There is no output polynomial algorithm for computing all minimal satisfying valuations of monotone Boolean formulae, unless P=NP.*

To prove this theorem, it is enough to show (see [17]) that the following decision problem is NP-hard:

Lemma 1. *Given a monotone Boolean formula ϕ and a set $\mathcal{M}$ of minimal valuations satisfying ϕ, deciding whether there exists a minimal valuation $\mathcal{V} \notin \mathcal{M}$ satisfying ϕ is NP-hard in the size of ϕ and $\mathcal{M}$.*

Proof. The proof is by reduction of the NP-hard *hypergraph 2-coloring problem* [13]: given a collection $H = \{E_1, \ldots, E_m\}$ of subsets of a set of vertices V, each of them of size 3, is there a set C such that $C \cap E_i \neq \emptyset$ and $(V \setminus C) \cap E_i \neq \emptyset$ for $i = 1, \ldots, m$.[3]

[3] In other words, both C and its complement must be hitting sets for $E_1, \ldots, E_m$.

Let $V = \{v_1, \ldots, v_n\}$ and $E_i = \{v_{i1}, v_{i2}, v_{i3}\}$ for all $i = 1, \ldots, m$. We represent every $v_i \in V$ by a propositional variable p_i, and construct the monotone Boolean formula $\phi := \psi \vee \bigvee_{i=1}^{m} \psi_i$, where

$$\psi = \bigwedge_{i=1}^{m} p_{i1} \vee p_{i2} \vee p_{i3} \quad \text{and} \quad \psi_i = p_{i1} \wedge p_{i2} \wedge p_{i3}$$

and the set $\mathcal{M} := \{\mathcal{V}_i := \{p_{i1}, p_{i2}, p_{i3}\} \mid 1 \leq i \leq m$ and no strict subset of $\mathcal{V}_i$ satisfies $\psi\}$.

It is easy to see that the formula ϕ as well as the set $\mathcal{M}$ can be constructed in time polynomial in the size of V and H. Moreover, every valuation $\mathcal{V}_i \in \mathcal{M}$ satisfies the formula ψ_i, and hence also ϕ. It is minimal since no strict subset of $\mathcal{V}_i$ satisfies (i) any of the ψ_j (which require valuations of size at least 3 to be satisfied) nor (ii) ψ since otherwise the condition in the definition of $\mathcal{M}$ would be violated. This shows that ϕ and $\mathcal{M}$ indeed form an instance of the problem considered in the lemma.

To complete the proof of NP-hardness of this problem, it remains to be shown that there is a minimal valuation $\mathcal{V} \notin \mathcal{M}$ satisfying ϕ iff there is a set $C \subseteq V$ such that $C \cap E_i \neq \emptyset$ and $(V \setminus C) \cap E_i \neq \emptyset$ for all $1 \leq i \leq m$.

For the *if direction*, let C be such a set, which we assume without loss of generality to be minimal with respect to set inclusion. We define the valuation $\mathcal{V}_C := \{p_i \mid v_i \in C\}$ and claim that it is the minimal valuation we are looking for. For every $1 \leq i \leq m$, $C \cap E_i \neq \emptyset$ implies that there is a $1 \leq j \leq 3$ such that $v_{ij} \in C$, which means that $p_{ij} \in \mathcal{V}_C$. This shows that $\mathcal{V}_C$ satisfies ψ and thus also ϕ. In addition, since $(V \setminus C) \cap E_i \neq \emptyset$, there is a $1 \leq k \leq 3$ such that $v_{ik} \notin C$. Thus, $\mathcal{V}_C$ is different from all the valuations $\mathcal{V}_i \in \mathcal{M}$, and it does not satisfy any of the formulae ψ_i.

To show that $\mathcal{V}_C$ is minimal, assume that $\mathcal{V}' \subset \mathcal{V}_C$. Since C is minimal, the set $C' := \{v_i \mid p_i \in \mathcal{V}'\} \subset C$ is such that there is a $1 \leq i \leq m$ with $C' \cap E_i = \emptyset$. This implies that $\mathcal{V}'$ does not satisfy $p_{i1} \vee p_{i2} \vee p_{i3}$, and hence it does not satisfy ψ. As a subset of $\mathcal{V}_C$, it also does not satisfy any of the formulae ψ_i, and thus it does not satisfy ϕ. This shows that $\mathcal{V}_C$ is a minimal valuation satisfying ϕ that does not belong to $\mathcal{M}$.

For the *only-if direction*, assume that there is a minimal valuation $\mathcal{V} \notin \mathcal{M}$ satisfying ϕ. This valuation cannot satisfy any of the formulae ψ_i. Indeed, (i) for $\mathcal{V}_i \in \mathcal{M}$ this would imply that $\mathcal{V}$ is a superset of one of the valuations in $\mathcal{M}$, which contradicts either the minimality of $\mathcal{V}$ or the fact that it does not belong to $\mathcal{M}$; (ii) for $\mathcal{V}_i \notin \mathcal{M}$ there would be a smaller valuation satisfying ψ, which contradicts the minimality of $\mathcal{V}$.

Since $\mathcal{V}$ is a model of ϕ, it must thus satisfy ψ. Define the set $C_\mathcal{V} := \{v_i \mid p_i \in \mathcal{V}\}$. Since $\mathcal{V}$ satisfies ψ, for every $1 \leq i \leq m$ there is a $1 \leq j \leq 3$ such that $p_{ij} \in \mathcal{V}$, and thus $v_{ij} \in C_\mathcal{V} \cap E_i$. On the other hand, since $\mathcal{V}$ does not satisfy any of the formulae ψ_i, for every $1 \leq i \leq m$ there must also be a $1 \leq l \leq 3$ such that $p_{i,k} \notin \mathcal{V}$, which means that $E_i \not\subseteq C_\mathcal{V}$ and hence $(V \setminus C) \cap E_i \neq \emptyset$. □

Theorem 4 follows from this lemma since an output polynomial algorithm whose runtime is bounded by the polynomial $P(|\phi|, |\mathcal{M}|)$ (where ϕ is the input and $\mathcal{M}$

Algorithm 1. Compute one MinA for $\mathcal{T} = (\emptyset \uplus \{t_1, \ldots, t_n\})$ w.r.t. $A \sqsubseteq B$.

```
1: if A ⋢_T B then
2:    return no MinA
3: S := {t_1, ..., t_n}
4: for 1 ≤ i ≤ n do
5:    if A ⊑_{S\{t_i}} B then
6:       S := S \ {t_i}
7: return S
```

the output) could be used to decide the problem introduced in the lemma in polynomial time as follows: given ϕ and $\mathcal{M}$, run the algorithm for time at most $P(|\phi|, |\mathcal{M}|)$ and check whether the generated valuations are exactly those in $\mathcal{M}$.

Theorem 4 shows that an algorithm for computing all MinAs based on computing the pinpointing formula and then producing its minimal satisfying valuations cannot be output polynomial. However, we can also use Theorem 4 to show that there cannot be any algorithm for computing MinAs that is output polynomial.

Theorem 5. *There is no output polynomial algorithm that computes, for a given $\mathcal{HL}$ TBox $\mathcal{T} = (\mathcal{T}_s \uplus \mathcal{T}_r)$ and concept names A, B occurring in $\mathcal{T}$, all MinAs for $\mathcal{T}$ w.r.t. $A \sqsubseteq B$, unless P=NP.*

Proof. We show that the problem of computing minimal valuations of monotone Boolean formulae can be reduced in polynomial time to the problem of computing MinAs of an $\mathcal{HL}$ TBox. Given a monotone Boolean formula ϕ, we introduce one concept name B_ψ for every subformula of ψ of ϕ, and one additional concept name A. We define TBoxes $\mathcal{T}_\psi$ for the subformulae ψ of ϕ by induction: if $\psi = p$ is a propositional variable, then $\mathcal{T}_\psi := \{A \sqsubseteq B_p\}$; if $\psi = \psi_1 \wedge \psi_2$, then $\mathcal{T}_\psi := \{B_{\psi_1} \sqcap B_{\psi_2} \sqsubseteq B_\psi\}$; if $\psi = \psi_1 \vee \psi_2$, then $\mathcal{T}_\psi := \{B_{\psi_1} \sqsubseteq B_\psi, B_{\psi_2} \sqsubseteq B_\psi\}$.

Obviously, the size of $\mathcal{T}_\phi$ is linear in the size of ϕ. In $\mathcal{T}_\phi$, we declare the GCIs $A \sqsubseteq B_p$ with p a propositional variable to be refutable, and the other GCIs to be irrefutable. With this division of $\mathcal{T}_\phi$ into a static and a refutable part, it is easy to see that there is a 1–1-correspondence between the minimal satisfying valuations of ϕ and the MinAs for $\mathcal{T}_\phi$ w.r.t. $A \sqsubseteq B_\phi$. In particular, given a MinA $\mathcal{S}$, the corresponding valuation $\mathcal{V}_\mathcal{S}$ consists of all p such that $A \sqsubseteq B_p \in \mathcal{S}$. Thus, if we could compute all MinAs with an output polynomial algorithm, we could do the same for all minimal satisfying valuations. □

5 Computing One MinA

For the sake of simplicity, we restrict the attention in this section to the case where all axioms in the TBox are assumed to be refutable. Note, however, the results could easily be extended to the general case.

A single MinA can be computed in polynomial time by the simple Algorithm 1, which goes through all axioms (in a given fixed order) and throws away those

that are not needed to obtain the desired subsumption relationship. Since the algorithm performs $n + 1$ subsumption tests (where n is the cardinality of $\mathcal{T}$), and each such test takes only polynomial time, the overall complexity of this algorithm is polynomial. It is easy to see that its output (in case $A \sqsubseteq_{\mathcal{T}} B$) is indeed a MinA for $\mathcal{T}$ w.r.t. $A \sqsubseteq B$.

Theorem 6. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T} = (\emptyset \uplus \mathcal{T}_r)$, Algorithm 1 terminates in time polynomial in the size of $\mathcal{T}$, and yields a MinA for $\mathcal{T}$ w.r.t. $A \sqsubseteq B$ if $A \sqsubseteq_{\mathcal{T}} B$.*

Although it requires only polynomial time, computing one MinA using Algorithm 1 may still be impractical for very large TBoxes like SNOMED. In fact, the algorithm has to make as many calls of the subsumption algorithm as there are axioms in the TBox (in the case of SNOMED, more than 350,000). Here we propose an *improved algorithm* that proceeds in two steps: (i) first compute a small (though not necessarily minimal) subset of the TBox from which the subsumption relationship follows; (ii) then minimize this set using Algorithm 1. Of course, this approach makes sense only if the algorithm used in step (i) is efficient and produces fairly small sets. It wouldn't help to use the trivial algorithm that always produces the whole TBox. In the following, we denote by nMinA such a (not necessarily minimal) subset obtained by step (i).

An algorithm that realize step (i) and runs in polynomial time can easily be obtained from the pinpointing algorithm sketched in Section 3 by strengthening the preconditions of rule applicability. The only modification is the following: if an assertion in the consequence of a rule already belongs to the current set of assertions, then this rule is not applied, i.e., once an assertion is there with some label, the label remains unchanged. Thus, every assertion (A, B) in the final set has a conjunction of propositional variables as its label, which clearly corresponds to a subset of the TBox from which the subsumption relationship $A \sqsubseteq B$ follows. In general, this subset is not minimal, however. (Because of the space constraints, we cannot give an example demonstrating this.)

As described until now, this modified algorithm works on normalized TBoxes. To get an appropriate subset of the original axioms, one can use a greedy strategy for producing a set of original axioms that covers a given set $\mathcal{S}$ of normalized axioms in the following sense. For each original axiom t, let $\mathcal{S}_t$ be the set of normalized axioms t gives rise to. The set $\mathcal{T}'$ of original axioms *covers* $\mathcal{S}$ if $\mathcal{S} \subseteq \bigcup_{t \in \mathcal{T}'} \mathcal{S}_t$. The use of a greedy strategy adds another possible source of non-minimality. (We use a non-optimal greedy strategy to keep the algorithm polynomial. In fact, even determining whether there is a cover set of size $\leq n$ is another NP-complete problem [13].)

Our preliminary experimental results confirm that this algorithm is indeed more practical than Algorithm 1. Based on the *refined algorithm* underlying the CEL reasoner [5, 4], we have implemented the practical algorithm described above for computing *exactly one* MinA for each subsumption relationship in $\mathcal{EL}^+$. The experiments were run on a variant of the Galen Medical Knowledge

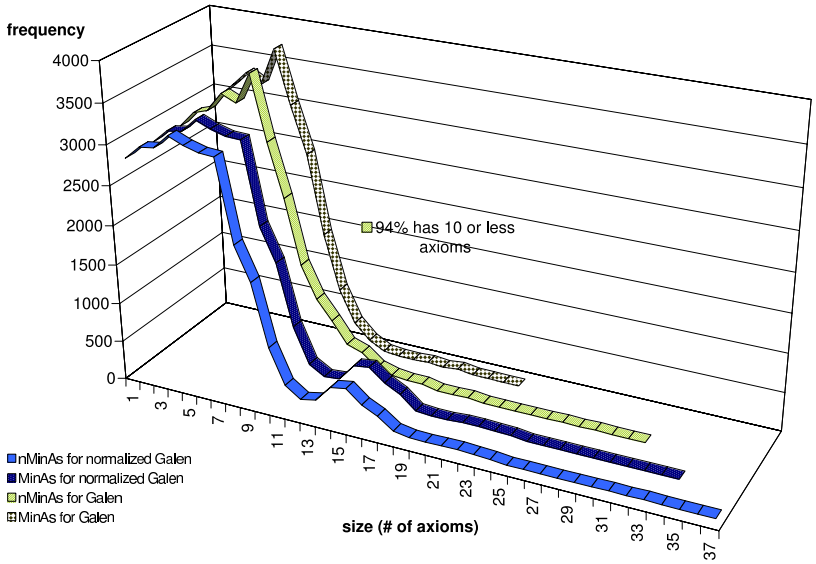

Fig. 2. Statistical data on the sizes of all computed axiom sets

Base [21],[4] which is a TBox consisting of more than 4,000 axioms. On the normalized version of this TBox, CEL needs about 14 sec to compute all subsumption relationships between concept names occurring in this TBox. Overall, over 27,000 subsumption relationships are computed. The overhead for computing for all of these subsumption relationships (possibly non-minimal) subsets from which they already follow was a bit more than 50%: the modified pinpointing algorithm described above needed about 23 sec. Going from the nMinAs for the normalized TBox to the corresponding nMinAs for the original Galen TBox with the greedy strategy took 0.27 sec. Finally, the overall time required for minimizing these sets using Algorithm 1 (with CEL [4] as the subsumption reasoner) was 9:45 min. For these last two numbers one should take into account, however, that these involved treating more than 27,000 such sets. For a single such set, the average post-processing time was negligible (on average 21 milliseconds). Also note that applying Algorithm 1 directly to the whole TBox for just one subsumption relationship (between *Renal-Artery* and *Artery-Which-Has-Laterality*) took more than 7 hours.

Thus, from the point of view of runtime, our practical algorithm behaves quite well on Galen. The same can be said about the quality of its results. Figure 2 displays the distribution graphs of the sizes of all computed nMinAs and their corresponding MinAs. The average size of an axiom set computed by the algorithm before using Algorithm 1 to minimize it was 5 (with maximum size 31),

[4] Since Galen uses expressivity not available in $\mathcal{EL}^+$, we have simplified it by removing inverse role axioms and treating functional roles as ordinary ones.

which is quite small and thus means that this set can directly be given to the user as an explanation for the subsumption relationship. Also, the computed nMinAs were almost minimal: on average, the possibly non-minimal sets computed by the algorithm were only 2.59% larger than the minimal ones. When considering the normalized TBox (i.e., without translating back to the original TBox), this number was even better (0.1%). This means that in most cases it is probably not necessary to further minimize the sets using Algorithm 1. If demanded by the user for a specific subsumption relationship it can still be done without taking much time.

6 Additional and Future Work on Pinpointing

The pinpointing extension of the subsumption algorithm for $\mathcal{EL}$ described in Section 3 as well as the pinpointing algorithm for $\mathcal{ALC}$ described in [3] are instances of a general approach for modifying "tableau-like" reasoning procedures to pinpointing procedures [6].

Instead of computing minimal subsets that have a given consequence, one sometimes also wants to compute maximal subsets that do not have a given consequence. Given the pinpointing formula ϕ, these sets correspond to maximal valuations that do not satisfy ϕ. The complexity results from Section 4 hold accordingly for such maximal sets. However, we currently do not know how to obtain a practical algorithm computing one such set (i.e., the results of Section 5 cannot be transferred to the case of maximal sets). Another open problem is the question of whether Theorem 5 also holds in the special case where the static TBox is empty.

Finally, space optimizations shall be studied to cater for large ontologies such as SNOMED with up to ten million subsumptions, and thus nMinAs.

References

[1] Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Kaelbling, L.P., Saffiotti, A. (eds.) Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), Edinburgh (UK), pp. 364–369. Morgan Kaufmann, Los Altos (2005)

[2] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)

[3] Baader, F., Hollunder, B.: Embedding defaults into terminological knowledge representation formalisms. J. of Automated Reasoning 14, 149–180 (1995)

[4] Baader, F., Lutz, C., Suntisrivaraporn, B.: CEL—a polynomial-time reasoner for life science ontologies. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 287–291. Springer, Heidelberg (2006)

[5] Baader, F., Lutz, C., Suntisrivaraporn, B.: Is tractable reasoning in extensions of the description logic $\mathcal{EL}$ useful in practice. Journal of Logic, Language and Information, Special Issue on Method for Modality on M4M (to appear, 2007)

[6] Baader, F., Penaloza, R.: Axiom pinpointing in general tableaux. LTCS-Report LTCS-07-01, Germany, See (2006), `http://lat.inf.tu-dresden.de/research/reports.html`

[7] Bailey, J., Stuckey, P.J.: Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In: Hermenegildo, M.V., Cabeza, D. (eds.) Practical Aspects of Declarative Languages. LNCS, vol. 3350, pp. 174–186. Springer, Heidelberg (2005)

[8] Brandt, S.: Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In: de Mántaras, R.L., Saitta, L. (eds.) Proc. of the 16th Eur. Conf. on Artificial Intelligence (ECAI 2004), pp. 298–302 (2004)

[9] Davydov, G., Davydova, I., Büning, H.K.: An efficient algorithm for the minimal unsatisfiability problem for a subclass of CNF. Ann. of Mathematics and Artificial Intelligence 23(3–4), 229–245 (1998)

[10] Dowling, W.F., Gallier, J.H.: Linear-time algorithms for testing the satisfiability of propositional horn formulae. Journal of Logic Programming 1(3), 267–284 (1984)

[11] Eiter, T., Gottlob, G.: Identifying the minimal transversals of a hypergraph and related problems. Technical Report CD-TR 91/16, Christian Doppler Labor für Expertensysteme, TU-Wien (1991)

[12] Eiter, T., Gottlob, G.: Identifying the minimal transversals of a hypergraph and related problems. SIAM J. Comput. 24(6), 1278–1304 (1995)

[13] Garey, M.R., Johnson, D.S.: Computers and Intractability — A guide to NP-completeness. W. H. Freeman and Company, San Francisco (1979)

[14] Hollunder, B.: Hybrid inferences in KL-ONE-based knowledge representation systems. In: Proc. of the German Workshop on Artificial Intelligence, pp. 38–47. Springer, Heidelberg (1990)

[15] Horrocks, I., Patel-Schneider, P.F., Van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. Journal of Web Semantics 1(1), 7–26 (2003)

[16] Johnson, D.S., Yannakakis, M., Papadimitriou, C.H.: On generating all maximal independent sets. Inf. Process. Lett. (1988)

[17] Kavvadias, D.J., Sideri, M., Stavropoulos, E.C.: Generating all maximal models of a Boolean expression. Inf. Process. Lett. (2000)

[18] Liffiton, M.H., Sakallah, K.A.: On finding all minimally unsatisfiable subformulas. In: Bacchus, F., Walsh, T. (eds.) SAT 2005. LNCS, vol. 3569, pp. 173–186. Springer, Heidelberg (2005)

[19] Meyer, T., Lee, K., Booth, R., Pan, J.Z.: Finding maximally satisfiable terminologies for the description logic $\mathcal{ALC}$. In: Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI 2006), AAAI Press/The MIT Press (2006)

[20] Parsia, B., Sirin, E., Kalyanpur, A.: Debugging OWL ontologies. In: Ellis, A., Hagino, T. (eds.) Proc. of the 14th International Conference on World Wide Web (WWW'05), pp. 633–640. ACM Press, New York (2005)

[21] Rector, A., Horrocks, I.: Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In: Proceedings of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI'97), Stanford, CA, AAAI Press (1997)

[22] Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: Gottlob, G., Walsh, T. (eds.) Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003), Acapulco, Mexico, pp. 355–362. Morgan Kaufmann, Los Altos (2003)

[23] Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. Artificial Intelligence 48(1), 1–26 (1991)

[24] Spackman, K.A., Campbell, K.E., Cote, R.A.: SNOMED RT: A reference terminology for health care. J. of the American Medical Informatics Association (Fall Symposium Supplement), 640–644 (1997)
[25] The Gene Ontology Consortium. Gene Ontology: Tool for the unification of biology. Nature Genetics, 25, 25–29 (2000)
[26] Zhang, L., Malik, S.: Validating SAT solvers using an independent resolution-based checker: Practical implementations and other applications. In: Proc. of the Conference on Design, Automation and Test in Europe (DATE'03), pp. 10880–10885. IEEE Computer Society Press, Los Alamitos (2003)