Lean Kernels in Description Logics

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

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

Abstract. Lean kernels (LKs) are an effective optimization for deriving the causes of unsatisfiability of a propositional formula. Interestingly, no analogous notion exists for explaining consequences of description logic (DL) ontologies. We introduce LKs for DLs using a general notion of consequence-based methods, and provide an algorithm for computing them which incurs in only a linear time overhead. As an example, we instantiate our framework to the DL $\mathcal{ALC}$. We prove formally and empirically that LKs provide a tighter approximation of the set of relevant axioms for a consequence than syntactic locality-based modules.

1 Introduction

Description logics (DLs) [6] are logic-based knowledge representation formalisms characterized by having an intuitive syntax and formal, well-understood semantics. These logics have been successfully used for representing the terminological knowledge of several domains, and are the logical formalism behind OWL 2, the standard ontology language for the Semantic Web. Along with the availability of better editors, this had led to the creation of larger ontologies; indeed, observing ontologies with tens of thousands of axioms is increasingly common.

Depending on the reasoning task of interest, not all axioms in an ontology may be relevant at any given time. To improve the efficiency, or even guarantee the feasibility of a task over large ontologies it is thus fundamental to focus only on a subset of pertinent axioms, usually called a *module* [12,14]. For example, in axiom pinpointing, where the task is to identify all the minimal subsets of axioms that entail a given consequence (called *MinAs* or *justifications*), only a small fraction of the ontology is relevant [34,37]. Similarly, error-tolerant and probabilistic reasoning can usually be restricted to a subset of relevant axioms [26,31]. Since the performance of reasoning methods depends on the size of the input ontology, a useful optimization consists in computing first a small module containing all the MinAs. Ideally, this module would contain exactly the union of all MinAs; however, computing this set is computationally expensive [30]. Thus, different approximations that are easy to compute have been proposed.

E. Blomqvist et al. (Eds.): ESWC 2017, Part I, LNCS 10249, pp. 518–533, 2017.
DOI: 10.1007/978-3-319-58068-5_32

The notion of a MinA in DLs is conceptually closely related to that of a MUS in propositional satisfiability [10,20,24,27]. Given a propositional formula in CNF, a MUS is a minimal subset of clauses that is still unsatisfiable. As in DLs, computing the union of all MUSes is computationally expensive, even for Horn formulas.[1] In this context, the lean kernel has been proposed as a tight and easier-to-compute overapproximation of the union of all MUSes [20–22]. As such, it is an effective way to improve MUS enumeration. Briefly, the lean kernel (LK) is the set of all clauses that are used in some resolution proof for unsatisfiability. Recent work has shown that the LK can be obtained by solving maximum satisfiability [25] or by finding a minimal correction subset [28], thus requiring at most a logarithmic number of calls to a witness-producing NP oracle (e.g. a SAT solver).

Interestingly, the analogous of the lean kernel has never been studied in the context of DLs or, to the best of our knowledge, any other ontology language. Perhaps one reason for this is that the notion of LK depends on a specific derivation procedure (i.e., resolution). In this paper, we introduce lean kernels for description logics. To keep our approach as general as possible, we do not focus on a specific DL or reasoning algorithm, but rather base our definitions on abstract *consequence-based methods*, of which many instances exist in the literature. We then present an algorithm for computing the LKs of all consequences derivable through these methods with only a linear time overhead. As an example of our general methods, we focus on the consequence-based algorithm for $\mathcal{ALC}$, which generalizes the well-known completion method for the light-weight DL $\mathcal{EL}^+$. We compare the LKs in this setting with locality-based modules, which have been used for optimizing axiom pinpointing in DLs, and show formally that LKs are in general strictly smaller than those modules.

Through an empirical analysis, we show that the lean kernel is typically smaller (in some cases much smaller) than locality-based modules. More precisely, we compute the LKs and the locality-based modules for all atomic subsumption relations derivable from well-known large ontologies written in $\mathcal{EL}^+$. In these instances, the size of the LK is typically less than 1% of the size of the original ontology, and in many cases one-tenth or less of the locality-based modules. Moreover, the time required to compute these LKs is small, and the set obtained often coincides with the union of all MinAs. These results show that lean kernel computation is an effective approximation of the union of all MinAs, which can be used for solving other related reasoning problems.

2 Preliminaries

Description logics (DLs) [6] are a family of knowledge representation formalisms that have been successfully used to handle the knowledge of many application domains. They are also the logical formalism underlying the standard Web Ontology Language (OWL 2). As prototypical examples, we briefly introduce $\mathcal{ALC}$ [32],

[1] Finding the union of MUSes is at least as hard as testing MUS membership, which is Σ_2^p-complete for arbitrary CNF formulas [23, Theorem 4].

$$\begin{aligned}
\top^{\mathcal{I}} &:= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &:= \emptyset \\
\neg C^{\mathcal{I}} &:= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &:= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &:= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\exists r.C)^{\mathcal{I}} &:= \{d \in \Delta^{\mathcal{I}} \mid \exists e \in C^{\mathcal{I}}.(d,e) \in r^{\mathcal{I}}\} \\
(\forall r.C)^{\mathcal{I}} &:= \{d \in \Delta^{\mathcal{I}} \mid \forall e.(d,e) \in r^{\mathcal{I}} \Rightarrow e \in C^{\mathcal{I}}\}
\end{aligned}$$

Fig. 1. Interpretation of complex concepts

the smallest propositionally closed DL, and $\mathcal{EL}^+$ [5], the logic underlying the OWL 2 EL profile.[2]

Let N_C and N_R be two disjoint sets of *concept-* and *role-names*, respectively. $\mathcal{ALC}$ *concepts* are constructed via the grammar rule

$$C ::= A \mid \top \mid \bot \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \exists r.C \mid \forall r.C, \tag{1}$$

where $A \in N_C$ and $r \in N_R$. $\mathcal{EL}$ *concepts* are obtained from the rule (1) by disallowing the constructors $\bot$ (bottom), $\neg$ (negation), $\sqcup$ (disjunction), and $\forall$ (value restrictions). Knowledge is represented by a *TBox*. An $\mathcal{ALC}$ *TBox* is a finite set of *general concept inclusions* (GCIs) $C \sqsubseteq D$ with C, D $\mathcal{ALC}$ concepts. An $\mathcal{EL}^+$ *TBox* is a finite set of GCIs formed by $\mathcal{EL}$ concepts, and *role inclusions* (RIs) $r_1 \circ \cdots \circ r_n \sqsubseteq s$, $n \geq 1$, with $r_i, s \in N_R$. We use the term *axiom* to denote both GCIs and RIs. Given an axiom $\alpha = x \sqsubseteq y$, we denote by $\mathsf{siglhs}(\alpha)$ and $\mathsf{sigrhs}(\alpha)$ the set of all symbols from N_C and N_R appearing in x and y, respectively, and $\mathsf{sig}(\alpha) = \mathsf{siglhs}(\alpha) \cup \mathsf{sigrhs}(\alpha)$.

The semantics of DLs is based on *interpretations* of the form $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is a non-empty *domain* and $\cdot^{\mathcal{I}}$ maps every $A \in N_C$ to a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and every $r \in N_R$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. This function is extended to arbitrary concepts as shown in Fig. 1. The interpretation $\mathcal{I}$ *satisfies* the GCI $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; it *satisfies* the RI $r_1 \circ \cdots r_n \sqsubseteq s$ if $r_1^{\mathcal{I}} \circ \cdots r_n^{\mathcal{I}} \subseteq s^{\mathcal{I}}$. $\mathcal{I}$ is a *model* of the TBox $\mathcal{T}$ if it satisfies all axioms in $\mathcal{T}$.

One of the main reasoning problems in DLs is *subsumption* between concepts; that is, to decide whether every model of a TBox $\mathcal{T}$ also satisfies the GCI $C \sqsubseteq D$ (denoted by $C \sqsubseteq_{\mathcal{T}} D$). Without loss of generality, we focus only on *atomic subsumption*, where C and D are restricted to be concept names. It is often important to determine, in addition, the axioms that are responsible for a subsumption to follow.

Definition 1 (MinA). *Let $\mathcal{T}$ be a TBox and $A, B \in N_C$. A* MinA *for $A \sqsubseteq B$ w.r.t. $\mathcal{T}$ is a subset $\mathcal{M} \subseteq \mathcal{T}$ s.t. $A \sqsubseteq_{\mathcal{M}} B$ and for every $\mathcal{N} \subsetneq \mathcal{M}$, $A \not\sqsubseteq_{\mathcal{N}} B$.*

Example 2. Consider the TBox $\mathcal{T}_{\mathsf{exa}} := \{\mathsf{ax}_1, \ldots \mathsf{ax}_6\}$, with

$$\begin{aligned}
\mathsf{ax}_1 &= A \sqsubseteq B & \mathsf{ax}_2 &= A \sqsubseteq \exists r.A & \mathsf{ax}_3 &= \exists r.B \sqsubseteq B \\
\mathsf{ax}_4 &= B \sqsubseteq C & \mathsf{ax}_5 &= A \sqsubseteq \exists s.C & \mathsf{ax}_6 &= \exists s.A \sqsubseteq B
\end{aligned}$$

[2] https://www.w3.org/TR/owl2-overview/.

Then $A \sqsubseteq_{\mathcal{T}_{\mathsf{exa}}} C$. Moreover, $\{\mathsf{ax}_1, \mathsf{ax}_4\}$ is the only MinA for $A \sqsubseteq C$ w.r.t. $\mathcal{T}_{\mathsf{exa}}$.

The importance of the computation of MinAs, also known as *axiom pinpointing*, for ontology debugging and repair is well-documented [8,16]. Other applications of this task are error-tolerant [26] and context-based reasoning [7]; and probabilistic reasoning under distribution [31] and Bayesian semantics [11], to name just a few recent examples.

One fundamental step for handling large ontologies is to extract a small subset of axioms (or *module*) that preserves the relevant properties of the original TBox. In the case of axiom pinpointing and its associated reasoning tasks, such a module should contain all the MinAs [38].

Definition 3 (MinA-preserving module). *Let $\mathcal{T}$ be a TBox and $A, B \in N_C$. A subset $\mathcal{S} \subseteq \mathcal{T}$ is a* MinA-preserving module *for $A \sqsubseteq B$ if for every MinA $\mathcal{M}$ for $A \sqsubseteq B$ w.r.t. $\mathcal{T}$ it holds that $\mathcal{M} \subseteq \mathcal{S}$.*

To improve the efficiency of reasoning, one would start with the smallest possible MinA-preserving module, and extract all the MinAs from this set. Clearly, the smallest MinA-preserving module is formed by the union of all MinAs. However, computing this union is known to be hard, even for restricted sublogics of $\mathcal{EL}$ [30]. Thus, other approaches, like reachability- and locality-based modules [13,36], have been suggested to compute a small module more efficiently.

To improve readability, we introduce syntactic locality modules only for TBoxes that are in *normal form*; that is, where all the GCIs are of the form

$$A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B, \quad A \sqsubseteq B_1 \sqcup \cdots \sqcup B_n, \quad \exists r.A \sqsubseteq B, \quad A \sqsubseteq \exists r.A, \quad A \sqsubseteq \forall r.B \tag{2}$$

with $n \geq 0$, $A_i, A \in N_C$, and $B_i, B \in N_C$. As usual, we identify the empty conjunction with $\top$ and the empty disjunction with $\bot$. Every TBox can be transformed to normal form preserving all relevant subsumption relations in polynomial time [5,35]. Moreover, modules obtained from a normalized TBox can be easily mapped to modules of the original TBox preserving the same properties [3,4].

Definition 4 (locality-based module). *Let $\mathcal{T}$ be a TBox in normal form, and Σ a signature. An axiom $\alpha \in \mathcal{T}$ is* $\bot$-local *w.r.t. Σ if $\mathsf{siglhs}(\alpha) \not\subseteq \Sigma$; it is* $\top$-local *w.r.t. Σ if $\mathsf{sigrhs}(\alpha) \not\subseteq \Sigma$. Locality is extended to sets of axioms in the obvious way.*

Let $A, B \in N_C$ and $\mathsf{x} \in \{\bot, \top\}$. The x*-module for $\mathcal{T}$ w.r.t. $A \sqsubseteq B$, denoted $\mathcal{M}^{\mathsf{x}}_{A,B}$, is the smallest subset $\mathcal{M} \subseteq \mathcal{T}$ s.t. $\mathcal{T} \setminus \mathcal{M}$ is* x*-local w.r.t. $\{A, B\} \cup \mathsf{sig}(\mathcal{M})$. The $\bot\top^*$-module for $\mathcal{T}$ w.r.t. $A \sqsubseteq B$ is the fixpoint reached from iteratively extracting the $\bot$- and $\top$-modules for $\mathcal{T}$ w.r.t. $A \sqsubseteq B$.*

Example 5. Consider again the TBox $\mathcal{T}_{\mathsf{exa}}$ from Example 2, which is already in normal form. Clearly, $\mathsf{ax}_1, \mathsf{ax}_3$, and ax_5 are not $\bot$-local w.r.t. $\{A\}$. Moreover, $\mathsf{ax}_3, \mathsf{ax}_4$, and ax_6 are not $\bot$-local w.r.t. $\mathsf{sig}(\{\mathsf{ax}_1, \mathsf{ax}_3, \mathsf{ax}_5\})$. Hence, $\mathcal{M}^{\bot}_{A,C} = \mathcal{T}_{\mathsf{exa}}$. Similarly, $\mathcal{M}^{\top}_{A,C} = \mathcal{T}_{\mathsf{exa}}$ and thus $\mathcal{M}^{\bot\top^*}_{A,C} = \mathcal{T}_{\mathsf{exa}}$.

In the following, the term *locality-based module* (LBM) refers to any of the three kinds of modules defined above. LBMs are MinA-preserving modules that can be computed in polynomial time. It has been shown, through various empirical studies, that these modules are typically small for realistic ontologies, in particular for $\mathcal{EL}^+$ [4,37]. In the next section we consider a new notion of module that has been previously considered in the context of propositional logic.

3 Lean Kernels

Intuitively, the lean kernel for a consequence c—e.g. a subsumption relation—is the set of all axioms that can appear in some proof for c. In general, the notion of a proof depends not only on the logic, but also on the decision method used. We now define lean kernels based on a general notion of consequence-based methods.

Abstracting from particularities, a *consequence-based method* is an algorithm that works on a set $\mathcal{A}$ of *assertions*, and uses rules to extend this set. The algorithm has two phases. First, the *normalization* phase transforms all the axioms into a suitable normal form. The *saturation* phase initializes the set $\mathcal{A}$ and extends it through rule applications. A *rule* is of the form $(\mathcal{B}_0, \mathcal{S}) \to \mathcal{B}_1$, where $\mathcal{B}_0, \mathcal{B}_1$ are finite sets of assertions, and $\mathcal{S}$ is a finite set of axioms in normal form. This rule is *applicable* to a set of axioms $\mathcal{T}$ and a set of assertions $\mathcal{A}$ if $\mathcal{B}_0 \subseteq \mathcal{A}$, $\mathcal{S} \subseteq \mathcal{T}$, and $\mathcal{B}_1 \not\subseteq \mathcal{A}$. Its *application* extends $\mathcal{A}$ to $\mathcal{A} \cup \mathcal{B}_1$. $\mathcal{A}$ is *saturated* if no rule is applicable to it. The method *terminates* if $\mathcal{A}$ is saturated after finitely many rule applications. After termination, the consequences of $\mathcal{T}$ can be read directly from $\mathcal{A}$; that is, to decide whether a consequence c follows from $\mathcal{T}$ it suffices to verify whether an assertion from the distinguished set $\mathsf{check}(c)$ appears in $\mathcal{A}$. The set $\mathsf{check}(c)$ contains the assertions that suffice for deciding that the consequence c holds. Given a rule $R = (\mathcal{B}_0, \mathcal{S}) \to \mathcal{B}_1$, we will use $pre(R)$, $ax(R)$ and $res(R)$ to denote the sets $\mathcal{B}_0$ of premises, $\mathcal{S}$ of axioms that trigger R, and $\mathcal{B}_1$ of assertions resulting of its applicability, respectively.

A simple example of a consequence-based method is the algorithm for reasoning with $\mathcal{ALC}$ TBoxes presented in [35]. Before describing this algorithm we introduce some necessary notation. A *literal* is either a concept name or a negated concept name. In the following, H, K denote (possibly empty) conjunctions of literals, and M, N are (possibly empty) disjunctions of concept names. For simplicity, we will often treat these conjunctions and disjunctions as sets.

The consequence-based algorithm for $\mathcal{ALC}$ works on assertions of the form (H, M) and (H, N, r, K). Intuitively, these assertions express $H \sqsubseteq_{\mathcal{T}} M$ and $H \sqsubseteq_{\mathcal{T}} N \sqcup \exists r.K$, respectively. The normalization phase transforms all GCIs to be of the form (2) introduced before. The saturation phase initializes $\mathcal{A}$ to contain the assertions (H, A) for all concept names A and all conjuctions of literals H from the normalized TBox, such that $A \in H$. The rules applied during saturation are depicted in the upper part of Table 1. After termination, for every two concept names A, B, it holds that $A \sqsubseteq_{\mathcal{T}} B$ iff $(A, B) \in \mathcal{A}$ or $(A, \bot) \in \mathcal{A}$. Thus, in this case, if the desired consequence c is the subsumption $A \sqsubseteq B$, then $\mathsf{check}(c) = \{(A, B)\}$.

Table 1. $\mathcal{ALC}$ and $\mathcal{EL}^+$ consequence-based algorithm rules $(\mathcal{B}_0, \mathcal{S}) \rightarrow \mathcal{B}_1$

$\mathcal{B}_0$	$\mathcal{S}$	$\mathcal{B}_1$
$(H \sqcap \neg A, N \sqcup A)$	$\emptyset$	(H, N)
$(H, N_1 \sqcup A_1), \ldots, (H, N_n \sqcup A_n)$	$A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B$	$(H, \bigsqcup_{i=1}^{n} N_i \sqcup B)$
$(H, N \sqcup A)$	$A \sqsubseteq \exists r.B$	(H, N, r, B)
$(H, M, r, K), (K, N \sqcup A)$	$\exists r.A \sqsubseteq B$	$(H, M \sqcup B, r, K \sqcap \neg A)$
$(H, M, r, K), (K, \bot)$	$\emptyset$	(H, M)
$(H, M, r, K), (H, N \sqcup A)$	$A \sqsubseteq \forall r.B$	$(H, M \sqcup N, r, K \sqcap B)$
$(A_0, \emptyset, r_1, A_1), \ldots (A_{n-1}, \emptyset, r_n, A_n)$	$r_1 \circ \cdots \circ r_n \sqsubseteq s$	$(A_0, \emptyset, s, A_n)$

We emphasize that this is only one of many consequence-based algorithms available. The completion-based algorithm for $\mathcal{EL}^+$ [5] is obtained by restricting the assertions to be of the form (A, B) and $(A, \emptyset, r, B)$ with $A, B \in N_C \cup \{\top\}$ and $r \in N_R$, and adding the rule in the last row of Table 1. Other examples include LTUR approach for Horn clauses [29], and methods for more expressive and Horn DLs [9,18,19]. For the rest of this section, we consider an arbitrary, but fixed, consequence-based method, that is sound and complete for deciding consequences from a set of axioms.

For the following definition, we need to weaken the notion of applicability of a rule. The rule $(\mathcal{B}_0, \mathcal{S}) \rightarrow \mathcal{B}_1$ is *weakly applicable* to $\mathcal{T}$ and $\mathcal{A}$ if $\mathcal{B}_0 \subseteq \mathcal{A}$ and $\mathcal{S} \subseteq \mathcal{T}$. In other words, the last condition of applicability is ignored.

Definition 6 (proof). *A* proof *for a consequence c is a finite sequence of rules $\mathcal{P} = (R_1, \ldots, R_n)$ such that: (i) for all $i, 1 \leq i \leq n$, R_i is weakly applicable after $R_1, \ldots, R_{i-1}$ have been applied, (ii) $\mathsf{check}(c) \cap res(R_n) \neq \emptyset$, and (iii) for every $i, 1 \leq i < n$, there is a non-initial assertion $b \in res(R_i)$ and a $j > i$ where $b \in pre(R_j)$. $\mathsf{Pf}(c)$ denotes the set of all proofs for c.*

Given a proof $\mathcal{P} = (R_1, \ldots, R_n)$, we denote by $\mathcal{T}_\mathcal{P}$ the set of all axioms appearing in $\mathcal{P}$; that is, $\mathcal{T}_\mathcal{P} := \bigcup_{i=1}^{n} ax(R_i)$. Using this notation, it is now possible to define a general notion of the lean kernel, which corresponds to the set of all axioms appearing in at least one proof for the consequence.

Definition 7 (lean kernel). *The* lean kernel *for a consequence c is the set $LK(c) := \bigcup_{\mathcal{P} \in \mathsf{Pf}(c)} \mathcal{T}_\mathcal{P}$.*

Notice that if there is a proof $\mathcal{P}$ for a consequence c, then the subset of axioms $\mathcal{T}_\mathcal{P}$ already entails c. Since the consequence-based algorithm for $\mathcal{ALC}$ is sound and complete for deciding subsumptions entailed by a TBox, it follows that the lean kernel is a MinA-preserving module. In fact, this is true for any consequence-based method $\mathcal{C}$.

Theorem 8. *Let $\mathcal{T}$ be a set of axioms, c a consequence, and $\mathcal{C}$ a sound and complete consequence-based method. Then $LK(c)$ is a MinA-preserving module for $\mathcal{T} \models c$.*

Table 2. A proof for $A \sqsubseteq C$ in $\mathcal{T}_{\mathsf{exa}}$.

	$\mathcal{B}_0$	$\mathcal{S}$	$\mathcal{B}_1$
R_1	(A, A)	$A \sqsubseteq B$	(A, B)
R_2	(A, A)	$A \sqsubseteq \exists r.A$	$(A, \emptyset, r, A)$
R_3	$(A, \emptyset, r, A), (A, B)$	$\exists r.B \sqsubseteq B$	$(A, B, r, A \sqcap \neg A)$
R_4	$(A, B, r, A \sqcap \neg A), (A \sqcap \neg A, \bot)$	$\emptyset$	(A, B)
R_5	(A, B)	$B \sqsubseteq C$	(A, C)

Proof. Let $\mathcal{M}$ be a MinA for c w.r.t. $\mathcal{T}$. Then, there is a sequence of rule applications that uses only the axioms in $\mathcal{M}$ and eventually adds an assertion from $\mathsf{check}(c)$ to $\mathcal{A}$. This sequence can be minimized by iteratively removing all superfluous rule applications, thus yielding a proof $\mathcal{P}$. If $\mathcal{T}_{\mathcal{P}} \subsetneq \mathcal{M}$, then $\mathcal{M}$ cannot be a MinA. Thus, we get that $\mathcal{M} \subseteq LK(c)$. □

Example 9. In our running example, there are two proofs for $A \sqsubseteq_{\mathcal{T}_{\mathsf{exa}}} C$ (modulo reordering of the rules) w.r.t. the consequence-based algorithm: one that uses the axioms $\mathsf{ax}_1, \mathsf{ax}_4$, and another one that uses ax_1–ax_4 as shown in Table 2. Thus $LK(A \sqsubseteq_{\mathcal{T}_{\mathsf{exa}}} C) = \{\mathsf{ax}_1, \ldots, \mathsf{ax}_4\}$.

Notice that the lean kernel from this example contains some axioms that do not belong to any MinA; specifically, ax_2 and ax_3 are not fundamental for deriving this consequence. On the other hand, this LK is a strict subset of the $\bot\top^*$-module for the same consequence (see Example 5). As we show next, this property holds in general for $\mathcal{ALC}$ TBoxes in normal form.

Theorem 10. *Let $\mathcal{T}$ be an $\mathcal{ALC}$ TBox and $A, B \in N_C$. Then, w.r.t. the completion algorithm, $LK(A \sqsubseteq_{\mathcal{T}} B) \subseteq \mathcal{M}^{\bot\top^*}_{A,B}$.*

Proof. To obtain this result, it suffices to show that $LK(A \sqsubseteq_{\mathcal{T}} B) \subseteq \mathcal{M}^{\bot}_{A,B}$ and $LK(A \sqsubseteq_{\mathcal{T}} B) \subseteq \mathcal{M}^{\top}_{A,B}$ hold. Assume that $LK(A \sqsubseteq_{\mathcal{T}} B) \not\subseteq \mathcal{M}^{\bot}_{A,B}$. Then there exists a proof $\mathcal{P} \in \mathsf{Pf}(c)$ such that $\mathcal{T}_{\mathcal{P}} \not\subseteq \mathcal{M}^{\bot}_{A,B}$; let α be the first axiom appearing in $\mathcal{P}$ such that $\alpha \notin \mathcal{M}^{\bot}_{A,B}$. Then α is $\bot$-local w.r.t. $\Sigma := \mathsf{sig}(\mathcal{M}^{\bot}_{A,B}) \cup \{A, B\}$; i.e., $\mathsf{siglhs}(\alpha) \not\subseteq \Sigma$. By construction, for a rule to be weakly applicable to the axiom α, $\mathsf{siglhs}(\alpha)$ must have been already derived. (This connection between axioms and rules is a property of this specific algorithm.) Thus, α cannot be $\bot$-local. An analogous but dual argument can be used to show that $LK(A \sqsubseteq_{\mathcal{T}} B) \subseteq \mathcal{M}^{\top}_{A,B}$ also holds. □

If we consider assertions and axioms as propositional variables, the rules in a consequence-based method can be seen as implications. In particular, they can be seen as (generalized) Horn clauses. This insight was exploited in [33] to encode the execution of the $\mathcal{EL}^+$ completion algorithm in a Horn formula. In [34], the notion of *COI module* was introduced based on this encoding. As formally defined, the COI module for a given consequence is in general a superset of its

LK, as it considers also all possible derivations of initial assertions, which can be seen as tautologies, and are disregarded by our definition of proof. However, this notion can be adapted to correspond to the LKs defined here.

4 Computing Lean Kernels

We now describe a method for computing LKs based on modifying consequence-based algorithms to keep track of the relevant axioms used in the derivation of the consequences. To achieve this, we first provide a unique label to each axiom in $\mathcal{T}$, which will be used to identify it. At the normalization phase, every normalized axiom α obtained is labeled with the set $\mathsf{lab}(\alpha)$ of the original axioms that produce it. To label all the derived assertions with the set of the relevant axioms that generate them, we modify the rule applicability condition, as well as the result of applying it.

First, all assertions a obtained at initialization are labeled with the empty set $\mathsf{lab}(a) = \emptyset$. The rule $R = (\mathcal{B}_0, \mathcal{S}) \rightarrow \mathcal{B}_1$ is *LK-applicable* to $\mathcal{T}$ and $\mathcal{A}$ if $\mathcal{B}_0 \subseteq \mathcal{A}$, $\mathcal{S} \subseteq \mathcal{T}$, and there exists some non-initial $b \in \mathcal{B}_1$ such that $b \notin \mathcal{A}$ or $\mathsf{lab}(R) := \bigcup_{a \in \mathcal{B}_0} \mathsf{lab}(a) \cup \bigcup_{\alpha \in \mathcal{S}} \mathsf{lab}(\alpha) \not\subseteq \mathsf{lab}(b)$. Its *application* extends $\mathcal{A}$ to $\mathcal{A} \cup \mathcal{B}_1$, sets $\mathsf{lab}(b) = \mathsf{lab}(R)$ for all new assertions b, and modifies the label of all previously existing assertions b from $\mathcal{B}_1$ to $\mathsf{lab}(b) \cup \mathsf{lab}(R)$. $\mathcal{A}$ is *LK-saturated* if no rule is LK-applicable to it. The LK of the consequence c is obtained as the union of the labels of all assertions in $\mathsf{check}(c)$. Given a consequence-based algorithm, we call the variant using these applicability conditions its *LK extension*.

Example 11. An example execution of the LK extension of the $\mathcal{ALC}$ algorithm over $\mathcal{T}_{\mathsf{exa}}$ appears in Table 3. Each step shows the preconditions for a rule application, along with the assertion added and its label. For simplicity, we removed all steps that derive obvious tautologies. In step 4, the assertion (A, B) is not added again; rather its label is extended to the set $\{\mathsf{ax}_1, \mathsf{ax}_2, \mathsf{ax}_3\}$. Notice, moreover, that the rule at step 4 would not be applicable in the original algorithm, but becomes LK-applicable, as more axioms are detected as potentially relevant.

It is easy to see that the LK extension of a consequence-based algorithm has essentially the same asymptotic run-time behavior as the original algorithm.

Table 3. Execution of the LK extension over $\mathcal{T}_{\mathsf{exa}}$.

step	$\mathcal{B}_0$	$\mathcal{S}$	$\mathcal{B}_1$	label
1	(A, A)	$A \sqsubseteq B$	(A, B)	$\{\mathsf{ax}_1\}$
2	(A, A)	$A \sqsubseteq \exists r.A$	$(A, \emptyset, r, A)$	$\{\mathsf{ax}_2\}$
3	$(A, \emptyset, r, A), (A, B)$	$\exists r.B \sqsubseteq B$	$(A, B, r, A \sqcap \neg A)$	$\{\mathsf{ax}_1, \mathsf{ax}_2, \mathsf{ax}_3\}$
4	$(A, B, r, A \sqcap \neg A), (A \sqcap \neg A, \bot)$	$\emptyset$	(A, B)	$\{\mathsf{ax}_1, \mathsf{ax}_2, \mathsf{ax}_3\}$
5	(A, B)	$B \sqsubseteq C$	(A, C)	$\{\mathsf{ax}_1, \ldots, \mathsf{ax}_4\}$
6	(A, A)	$A \sqsubseteq \exists s.C$	$(A, \emptyset, s, C)$	$\{\mathsf{ax}_5\}$

Indeed, the extension generates the same set of assertions. The main difference is that, while the original algorithm generates each assertion only once, its LK extension may update its label several times. Notice, however, that each update strictly extends the set of axioms in the label. Thus, each label can be updated at most $|\mathcal{T}|$ times, and the run-time of the LK extension is increased by a linear factor. Another important feature of this extension is that the resulting labels do not depend on the order in which the rules are applied.

Theorem 12. *If the LK extension of a consequence-based method is executed until LK saturation, then* $\mathsf{lab}(\mathsf{check}(c))$ *is the lean kernel for* c*.*

Proof. Let $\mathcal{P}$ be a proof for c. Then $\mathcal{P}$ is a sequence of weakly applicable rules. Consider the LK application of the same sequence. If one rule $(\mathcal{B}_0, \mathcal{S}) \to \mathcal{B}_1$ is not LK-applicable, it means that its application would not change the label of the assertions in $\mathcal{B}_1$. Condition (iii) in Definition 6 guarantees that all axioms in $\mathcal{T}_\mathcal{P}$ appear in the labels of the elements of $\mathsf{check}(c)$ obtained after the execution of this sequence; otherwise, the rule applications in which the missing axioms appear could be removed from the sequence. Hence, $\mathcal{T}_\mathcal{P} \subseteq \mathsf{lab}(\mathsf{check}(c))$.

Conversely, for every axiom $\alpha \in \mathsf{lab}(\mathsf{check}(c))$ there exists a sequence of rule LK-applications that eventually adds the axiom α to this label. Since every LK-applicable rule is also weakly applicable, such a sequence can be trivially transformed into a proof. Thus, $\mathsf{lab}(\mathsf{check}(c)) \subseteq LK(c)$. □

Notice that the runtime behaviour of the LK extension is governed by the underlying decision procedure. For instance, our approach would compute the LK of all atomic subsumption relations following from an $\mathcal{ALC}$ TBox in exponential time. This is in contrast to LBM and other MinA-preserving modules (see e.g. [4]), which can be computed very efficiently, at the cost of losing soundness and potentially including many superfluous axioms.

Example 13. Let $\mathcal{T}'_{\mathsf{exa}} := \mathcal{T}_{\mathsf{exa}} \cup \{\exists s.A \sqsubseteq B_i, B_i \sqsubseteq C \mid i \in I\}$, where I is an arbitrarily large set of indices and $\mathcal{T}_{\mathsf{exa}}$ is the TBox from Example 2. Then the $\bot\top^*$-module for $\mathcal{T}'_{\mathsf{exa}}$ w.r.t. $A \sqsubseteq C$ is $\mathcal{T}'_{\mathsf{exa}}$, while $LK(A \sqsubseteq_{\mathcal{T}'_{\mathsf{exa}}} C) = \{\mathsf{ax}_1, \ldots, \mathsf{ax}_4\}$.

Moreover, the TBox $\mathcal{T}''_{\mathsf{exa}} := \mathcal{T}'_{\mathsf{exa}} \setminus \{\mathsf{ax}_4\}$ does not entail $A \sqsubseteq C$, and hence $LK(A \sqsubseteq_{\mathcal{T}''_{\mathsf{exa}}} C) = \emptyset$. However, the $\bot\top^*$-module for $\mathcal{T}''_{\mathsf{exa}}$ w.r.t. $A \sqsubseteq C$ is the set $\{\mathsf{ax}_1\} \cup \{\exists s.A \sqsubseteq B_i, B_i \sqsubseteq C \mid i \in I\}$.

This example shows that LBMs may not provide much information about the entailments or their axiomatic causes. As we see in the following section, this phenomenon can be observed in realistic TBoxes used in practice.

5 Experiments

We performed an experimental study aimed at assessing the sizes of the LKs in practice, and comparing them with the $\bot$-, $\top$- and $\bot\top^*$-modules (see Definition 4). A prototype for computing the different modules was implemented in C++ for the DL $\mathcal{EL}^+$ and several experiments were run on a Linux cluster

Table 4. Ontologies considered in the experiments

$\mathcal{T}$	# GCIs	# RIs	$\lvert\text{class}(\mathcal{T})\rvert$
GENE	20465	1	164743
NCI	46800	0	252519
NOT-GALEN	3937	442	27980
FULL-GALEN	35530	1014	453674
SNOMED-CT	307692	12	5333580

(2 GHz, 128 GB) on different well-known bio-medical ontologies. These ontologies are: GENE, NCI, NOT-GALEN, FULL-GALEN and SNOMED-CT (v. 2009). Table 4 shows the number of GCIs, RIs and atomic subsumption relations in these ontologies. For each ontology, except SNOMED, the different modules were computed for all atomic subsumption relations entailed by the ontology. For SNOMED, both the LK and $\bot$-modules were computed for all atomic subsumption relations. Computing all $\top$-modules was infeasible; hence we considered the 240 concept names subsuming the largest number of concepts. These yield the $\top$-modules for around 2.7 million subsumption relations, for which also the $\bot\top^*$-modules were computed.

Figure 2 compares the sizes of the LKs against the size of locality-based modules, shown as a percentage. Theorem 10 guarantees that the size of the lean kernel never exceeds the size of the $\bot$, $\top$ and $\bot\top^*$-modules. A result of 10^2 means that the size of the lean kernel matches the size of the other module, with smaller values indicating more significant size reductions. Notice that in most cases the LK is smaller than both $\bot$- and $\top$-modules, often achieving a significant reduction. The difference is more significant w.r.t. $\top$-modules, where the LK is usually at least two orders of magnitude smaller. Compared to $\bot$-modules, LKs are usually smaller in a factor or 2 or more. Despite their large size, $\top$-modules appear to be useful as shown in the sizes of $\bot\top^*$-modules. Recall that $\bot\top^*$-modules are the fixpoint from the iterative application of $\bot$- and $\top$-modules. It is thus guaranteed that $\bot\top^*$-modules will not be greater than those modules. Noticeably, for GENE $\bot\top^*$-modules are in general equal to the LK; only in 4.5% of the cases is the LK strictly smaller. For NCI, $\bot\top^*$-modules match the LKs for all subsumption

Table 5. Module sizes

	$\bot$			$\top$			$\bot\top^*$			LK		
$\mathcal{T}$	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
GENE	2	18.33	68	1	2012.04	8786	1	5.70	66	1	5.45	52
NCI	1	36.70	398	1	2431.16	11572	1	7.08	85	1	7.08	85
NOT-GALEN	1	88.06	495	1	3433.56	3796	1	23.47	300	1	13.78	80
FULL-GALEN	1	9727.15	15543	1	34244.03	35733	1	8940.22	24586	1	68.90	416
SNOMED-CT	1	51.71	264	3269	216213.80	307704	1	41.46	264	1	40.19	220

Table 6. Running times (in seconds). For LKs, the time refers to the total time for computing all LKs; for LBMs the time refers to each subsumption relation.

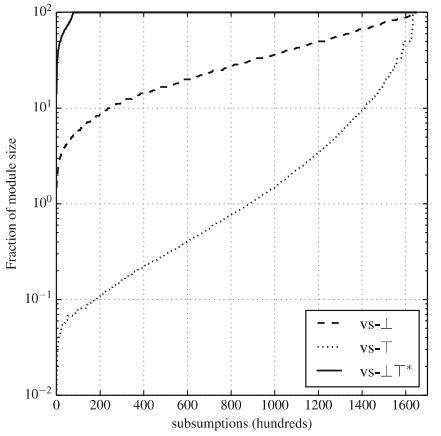
$\mathcal{T}$	LKs	$\bot$			$\top$			$\bot\top^*$		
		Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
GENE	1.98	0.00	0.01	0.04	0.00	0.01	0.04	0.00	0.01	0.06
NCI	3.79	0.00	0.01	0.10	0.01	0.01	0.09	0.01	0.02	0.13
NOT-GALEN	4.22	0.00	0.01	0.02	0.00	0.02	0.04	0.00	0.01	0.03
FULL-GALEN	461.03	0.00	0.36	1.12	0.01	1.60	2.90	0.02	2.46	8.06
SNOMED-CT	11200.53	0.11	0.65	5.34	0.15	2081.77	5209.57	0.26	3.91	28.92

relations. For NOT-GALEN and FULL-GALEN, LKs are significantly smaller than $\bot\top^*$-modules, especially in the latter case. Regarding SNOMED, the reduction w.r.t. $\bot$-modules is slightly smaller than in other cases: LKs are usually 75% of the $\bot$-modules. $\top$-modules are in general quite large, and $\bot\top^*$-modules improve over $\bot$-modules, getting close to the LKs in most cases.

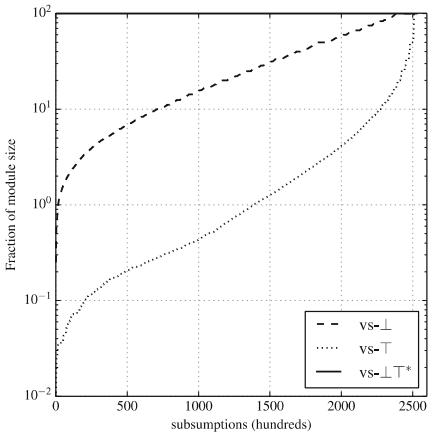
These results are confirmed in Table 5, which shows, for each ontology the minimum, average and maximum size of each module over the selected 2.7 million instances of SNOMED and all the atomic subsumptions of the other ontologies. Observe that $\top$-modules usually represent a large fraction of the ontology, while $\bot$-modules, and especially $\bot\top^*$-modules, are quite small for GENE, NCI, and SNOMED representing in general less than one percent of the ontology. For NOT-GALEN and FULL-GALEN, $\bot$- and $\bot\top^*$-modules are not so small; in the latter case representing around half of the ontology in most cases. Noticeably, LK modules are in general a small fraction of the ontologies, representing less than 1% of it in most cases.

For the five ontologies we computed the 99.9% confidence interval for the mean size of the LKs and $\bot\top^*$-modules. In all cases, excepting only NCI, these intervals do not overlap. Thus, we can conclude that the difference in size between LKs and $\bot\top^*$-modules is highly statistically significant ($p < 0.001$). In NCI, this difference does not exist as both kinds of modules coincide.

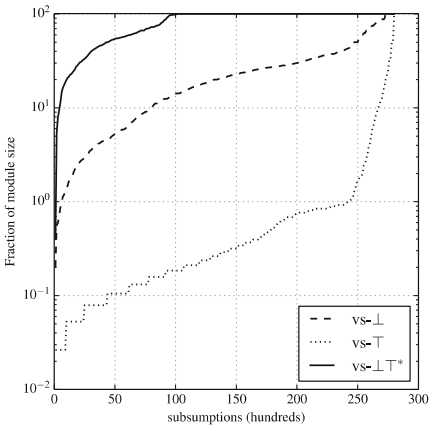
In order to get a more detailed view on the ability of the different modules to approximate the union of MinAs (UMinAs), we computed UMinAs for all the atomic subsumption relations of GENE and NOT-GALEN using BEACON [1]. The results show that LKs match UMinAs for all subsumption relations in GENE, whereas $\bot\top^*$-modules match UMinAs in 95.49% of the cases, being in average around 3.26% larger than UMinAs. For NOT-GALEN, both the LKs and the $\bot\top^*$-modules equal UMinAs for a similar percentage of the subsumption relations (43.78% and 41.28% respectively), and in average LKs are around 154.41% larger than UMinAs whereas $\bot\top^*$ modules are around 362.06% larger than UMinAs. These experiments confirm earlier results on the accuracy of locality-based modules, and reveal that LKs constitute tighter and more stable approximations.



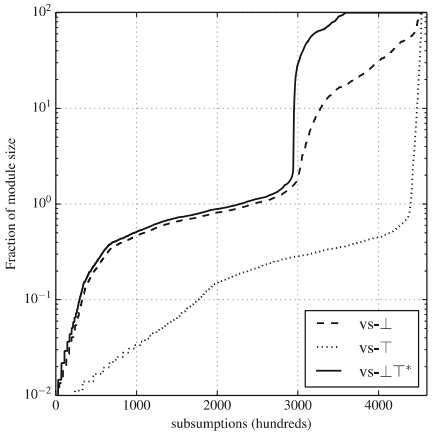
(a) GENE



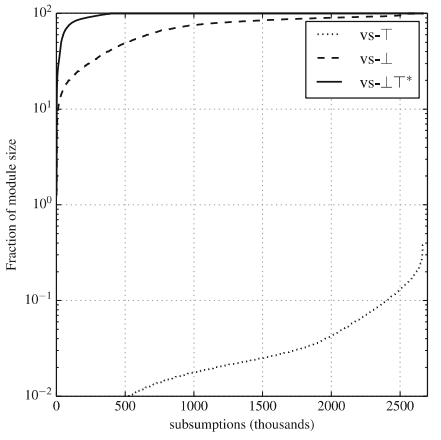
(b) NCI



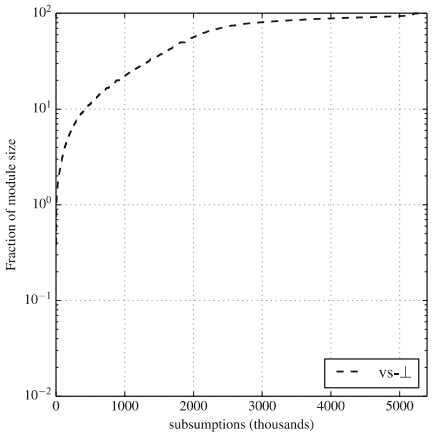
(c) NOT-GALEN



(d) FULL-GALEN



(e) SNOMED-CT (all)



(f) SNOMED-CT (ß2.7M)

Fig. 2. LK module size w.r.t. locality-based modules (in percentage)

The time required to compute the modules, after parsing the ontology, is shown in Table 6. In the case of $\top$-modules for SNOMED, only the selected instances are reported. Recall that the LK algorithm computes the LKs for *all* atomic subsumptions simultaneously. Thus, e.g., it takes less than 2s. to obtain the LKs of *all* 16743 consequences from GENE. In contrast, the times for $\bot$- and $\top$-modules are for each concept name appearing in either the left or the right-hand side of any atomic subsumptions respectively. The times for $\bot\top^*$-modules consider each of the subsumption relations. In some cases, computing $\bot\top^*$-modules took less time than computing the corresponding $\top$-modules (especially in SNOMED). The reason is that in these cases, $\top$-modules are large and expensive to compute for the whole ontology, but in the computation of $\bot\top^*$-modules, we compute $\top$-modules of usually small subsets of the ontologies. Our results show that it is feasible to compute all LKs for very large ontologies. In addition, the LK computation could be performed as a preprocessing step for enhancing other reasoning services. The columns for $\bot$-, $\top$-, and $\bot\top^*$-modules show the minimum, average, and maximum time required to compute these modules. Notice that, in the case of GENE, it takes over 185s. to compute all the $\bot$-modules and over 1647s. to compute all the $\bot\top^*$-modules (amortizing the parsing time over all of them). For the larger ontologies, i.e. FULL-GALEN and SNOMED, computing $\bot\top^*$ modules is, in average, significantly more time consuming than computing $\bot$-modules. If one is interested in analyzing only one consequence from an ontology, a better strategy would be to first compute the $\bot$-module, and then find the LKs for the consequences of that subontology. This goal-directed approach yields LKs in very short time.

6 Conclusions

We have introduced the notion of lean kernels for description logics, as an effective way to approximate the union of all MinAs for a given consequence. Our definition of LKs is based on a general notion of a consequence-based algorithm, and is thus applicable to a large variety of logical formalisms and reasoning approaches. We have shown how the consequence-based decision method can be transformed into a procedure for computing LKs, with only a linear overhead. As an example for our formalism, we have instantiated the definitions to well-known reasoning algorithms for the DLs $\mathcal{ALC}$ and $\mathcal{EL}^+$.

From a theoretical point of view, we have shown that LKs based on these methods are MinA-preserving modules; i.e., they contain all the axioms appearing in some MinA for a given consequence. More interestingly, they are contained in the different versions of locality-based modules that have been proposed in the literature. While the computation of LKs for an $\mathcal{ALC}$ ontology requires exponential time in the worst case, the $\mathcal{EL}^+$ completion algorithm can be adapted to find the lean kernels of all atomic subsumptions entailed from an $\mathcal{EL}^+$ ontology in polynomial time.

To evaluate the effectiveness of LKs as a means to approximate the union of all MinAs, we computed the LKs and the three variants of locality-based

modules for all atomic subsumptions derivable from large bio-medical ontologies that are commonly used as benchmarks for $\mathcal{EL}^+$ systems. Overall, more than 6 million subsumption relations were analyzed. Our experiments show that LKs are often (much) smaller than LBMs, and in most cases only a small fraction of the original ontology. Interestingly, for the Gene Ontology LKs in fact coincide with the union of all MinAs. Moreover, they can be effectively computed. Thus, lean kernel computation can improve the runtime of axiom pinpointing and other related reasoning tasks. In future work we will analyze the practical benefits of computing these modules for those tasks.

It is known that syntactic LBMs are also MinA preserving, and can be computed in polynomial time even for very expressive DLs where reasoning is 2ExpTime-hard [2–4,17]. However, an additional reasoning step is required to verify whether the consequence follows from the ontology. Following our approach, we can compute the LKs of all relevant consequences of an ontology simultaneously, while reasoning. In that sense, LKs are more closely related to *semantic* locality-based modules [13]. It has been observed in [15] that the difference between syntactic and semantic LBMs is not statistically significant. Our results thus suggest that the size difference between semantic LBMs and LKs is also statistically significant, although a full empirical analysis is needed to justify this conclusion.

Acknowledgements. We would like to thank Beatriz Peñaloza for her help on statistical methods. Carlos Mencía is supported by grant TIN2016-79190-R.

References

1. Arif, M.F., Mencía, C., Ignatiev, A., Manthey, N., Peñaloza, R., Marques-Silva, J.: BEACON: An efficient SAT-based tool for debugging EL+ ontologies. In: SAT, pp. 521–530 (2016)
2. Romero, A.A.: Ontology module extraction and applications to ontology classification. Ph.D. thesis, University of Oxford, UK (2015)
3. Romero, A.A., Kaminski, M., Grau, B.C., Horrocks, I.: Ontology module extraction via datalog reasoning. In: AAAI, pp. 1410–1416 (2015)
4. Romero, A.A., Kaminski, M., Cuenca Grau, B., Horrocks, I.: Module extraction in expressive ontology languages via datalog reasoning. JAIR **55**, 499–564 (2016)
5. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: IJCAI, pp. 364–369 (2005)
6. 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)
7. Baader, F., Knechtel, M., Peñaloza, R.: Context-dependent views to axioms and consequences of semantic web ontologies. J. Web Semant. **12–13**, 22–40 (2012)
8. Baader, F., Suntisrivaraporn, B.: Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}^+$. In: KR-MED (2008)
9. Bate, A., Motik, B., Grau, B.C., Simancik, F., Horrocks, I.: Extending consequence-based reasoning to SRIQ. In: KR, pp. 187–196 (2016)
10. Belov, A., Lynce, I., Marques-Silva, J.: Towards efficient MUS extraction. AI Commun. **25**(2), 97–116 (2012)

11. Ceylan, İİ., Peñaloza, R.: The bayesian description logic $\mathcal{BEL}$. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) IJCAR 2014. LNCS (LNAI), vol. 8562, pp. 480–494. Springer, Cham (2014). doi:10.1007/978-3-319-08587-6_37
12. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: Just the right amount: Extracting modules from ontologies. In: WWW, pp. 717–726 (2007)
13. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. J. Artif. Intell. Res. (JAIR) **31**, 273–318 (2008)
14. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Extracting modules from ontologies: A logic-based approach. In: Stuckenschmidt, H., Parent, C., Spaccapietra, S. (eds.) Modular Ontologies. LNCS, vol. 5445, pp. 159–186. Springer, Heidelberg (2009). doi:10.1007/978-3-642-01907-4_8
15. Vescovo, C., Klinov, P., Parsia, B., Sattler, U., Schneider, T., Tsarkov, D.: Empirical study of logic-based modules: Cheap is cheerful. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013. LNCS, vol. 8218, pp. 84–100. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41335-3_6
16. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of OWL DL entailments. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 267–280. Springer, Heidelberg (2007). doi:10.1007/978-3-540-76298-0_20
17. Kaminski, M., Nenov, Y., Grau, B.C.: Datalog rewritability of disjunctive datalog programs and its applications to ontology reasoning. In: AAAI, pp. 1077–1083 (2014)
18. Kazakov, Y.: Consequence-driven reasoning for Horn SHIQ ontologies. In: Boutilier, C. (ed.) IJCAI 2009, pp. 2040–2045 (2009)
19. Kazakov, Y., Krötzsch, M., Simancik, F.: The incredible ELK - from polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies. JAR **53**(1), 1–61 (2014)
20. Büning, H.K., Kullmann, O.: Minimal unsatisfiability and autarkies. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185, pp. 339–401. IOS Press (2009)
21. Kullmann, O.: Investigations on autark assignments. Discrete Appl. Math. **107**(1–3), 99–137 (2000)
22. Kullmann, O., Lynce, I., Marques-Silva, J.: Categorisation of clauses in conjunctive normal forms: Minimally unsatisfiable sub-clause-sets and the lean kernel. In: Biere, A., Gomes, C.P. (eds.) SAT 2006. LNCS, vol. 4121, pp. 22–35. Springer, Heidelberg (2006). doi:10.1007/11814948_4
23. Liberatore, P.: Redundancy in logic I: CNF propositional formulae. Artif. Intell. **163**(2), 203–232 (2005)
24. Liffiton, M.H., Previti, A., Malik, A., Marques-Silva, J.: Fast, flexible MUS enumeration. Constraints **21**(2), 223–250 (2016)
25. Liffiton, M., Sakallah, K.: Searching for autarkies to trim unsatisfiable clause sets. In: Kleine Büning, H., Zhao, X. (eds.) SAT 2008. LNCS, vol. 4996, pp. 182–195. Springer, Heidelberg (2008). doi:10.1007/978-3-540-79719-7_18
26. Ludwig, M., Peñaloza, R.: Error-tolerant reasoning in the description logic $\mathcal{EL}$. In: JELIA, pp. 107–121 (2014)
27. Marques-Silva, J., Ignatiev, A., Mencía, C., Peñaloza, R.: Efficient reasoning for inconsistent horn formulae. In: Michael, L., Kakas, A. (eds.) JELIA 2016. LNCS (LNAI), vol. 10021, pp. 336–352. Springer, Cham (2016). doi:10.1007/978-3-319-48758-8_22

28. Marques-Silva, J., Ignatiev, A., Morgado, A., Manquinho, V.M., Lynce, I.: Efficient autarkies. In: ECAI, pp. 603–608 (2014)
29. Minoux, M.: LTUR: A simplified linear-time unit resolution algorithm for horn formulae and computer implementation. Inf. Process. Lett. **29**(1), 1–12 (1988)
30. Peñaloza, R., Sertkaya, B.: On the complexity of axiom pinpointing in the $\mathcal{EL}$ family of description logics. In: KR (2010)
31. Riguzzi, F., Bellodi, E., Lamma, E., Zese, R.: Probabilistic description logics under the distribution semantics. Semant. Web **6**(5), 477–501 (2015)
32. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. Artif. Intell. **48**(1), 1–26 (1991)
33. Sebastiani, R., Vescovi, M.: Axiom pinpointing in lightweight description logics via horn-SAT encoding and conflict analysis. In: Schmidt, R.A. (ed.) CADE 2009. LNCS (LNAI), vol. 5663, pp. 84–99. Springer, Heidelberg (2009). doi:10.1007/978-3-642-02959-2_6
34. Sebastiani, R., Vescovi, M.: Axiom pinpointing in large $\mathcal{EL}^+$ ontologies via SAT and SMT techniques. Technical Report DISI-15-010, DISI, University of Trento, Italy, April 2015. http://disi.unitn.it/rseba/elsat/elsat_techrep.pdf
35. Simancik, F., Kazakov, Y., Horrocks, I.: Consequence-based reasoning beyond horn ontologies. In: IJCAI 2011, pp. 1093–1098 (2011). IJCAI/AAAI
36. Suntisrivaraporn, B.: Module extraction and incremental classification: A pragmatic approach for $\mathcal{EL}^+$ ontologies. In: ESWC, pp. 230–244 (2008)
37. Suntisrivaraporn, B.: Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies. Ph.D. thesis, TU Dresden (2009)
38. Suntisrivaraporn, B., Qi, G., Ji, Q., Haase, P.: A modularization-based approach to finding all justifications for OWL DL entailments. In: Domingue, J., Anutariya, C. (eds.) ASWC 2008. LNCS, vol. 5367, pp. 1–15. Springer, Heidelberg (2008). doi:10.1007/978-3-540-89704-0_1