Axiom Pinpointing

Rafael Peñaloza

IKR3 Research Lab, University of Milano-Bicocca,
rafael.penaloza@unimib.it

Abstract. Axiom pinpointing refers to the task of finding the specific axioms in an ontology which are responsible for a consequence to follow. This task has been studied, under different names, in many research areas, leading to a reformulation and reinvention of techniques. In this chapter, we present a general overview to axiom pinpointing, providing the basic notions, different approaches for solving it, and some variations and applications which have been considered in the literature. This should serve as a starting point for researchers interested in related problems, with an ample bibliography for delving deeper into the details.

1 Introduction

Intelligent applications need to represent and handle knowledge effectively. For that reason, many different knowledge representation languages have been developed, providing formal semantics and reasoning methods for deriving implicit consequences from explicitly represented elements (also called axioms). As these knowledge bases or ontologies grow, they become harder to maintain and verify; and when—inevitably—errors occur, they are harder to understand and correct. Indeed, it is nowadays common to encounter knowledge bases with tens-of-thousands of axioms, and detecting the handful of them responsible for a given consequence would be impossible without the help of an automated tool.

Axiom pinpointing refers to the task of identifying the axioms in a knowledge base that are responsible for a given consequence. Assuming that the representation language is monotonic (that is, adding new knowledge does not remove any previous consequences), a relevant set of axioms is nothing more than a subset-minimal subontology which still entails the consequence under consideration. Such a set is called a justification. It is not difficult to see (and will be exemplified in the following sections) that there may exist multiple justifications for a single consequence. It is thus important to try to compute them all, for a full understanding of the derivation.

Understanding the causes of a consequence are not just instrumental for knowledge engineers to understand the ontologies they work on. Axiom pinpointing is also a relevant step for repairing modeling errors which could have been introduced through misunderstanding, automated knowledge extraction, or merely typing or methodological errors. Identifying the potentially faulty axioms is the first step towards correcting the error. A third usecase for axiom

pinpointing—beyond analysis and repair—is explainability: if an intelligent system makes a decision based on a reasoning process, it is important to be able to explain the reasoning behind this decision to all the stakeholders involved.

We provide a general overview of axiom pinpointing over many different representation languages. Although we use terminology and results primarily developed in the context of description logics [4], we try to keep the presentation as general as possible to include other well-known monotonic formalisms like databases, logic programming, and propositional logic. Our goal is to describe what the main reasoning tasks associated to axiom pinpointing are, provide the basic templates for solving them, and present a few variants and applications from the literature. This general description should serve as a first step towards a unified description of the tasks for different areas of knowledge representation, and aid in a common development of new methods and tools.

2 Axiom Pinpointing

To make the presentation as general as possible, we consider an abstract notion of an *ontology language*, which has four components: a class $\mathcal{A}$ of well-formed *axioms*; a class $\mathcal{C}$ of *consequences*; a class $\mathfrak{O} \subseteq \mathscr{P}_{\mathsf{fin}}(\mathcal{A})$ of valid *ontologies*, where $\mathscr{P}_{\mathsf{fin}}(X)$ is the class of all *finite* subsets of X, such that if $\mathcal{O} \subseteq \mathcal{O}'$ and $\mathcal{O}' \in \mathfrak{O}$, then $\mathcal{O} \in \mathfrak{O}$ (that is, every subset of an ontology is also an ontology); and an *entailment relation* $\models \subseteq \mathfrak{O} \times \mathcal{C}$, expressed in infix notation, such that for every two ontologies $\mathcal{O}, \mathcal{O}' \in \mathfrak{O}$ and consequence $c \in \mathcal{C}$, if $\mathcal{O} \models c$ and $\mathcal{O} \subseteq \mathcal{O}'$, then $\mathcal{O}' \models c$; that is, the entailment relation must be monotonic w.r.t. the ontology.

We note that in some existing work on axiom pinpointing and related topics, an ontology is often defined to be just a finite set of axioms, which is a special case of our definition. We decided to use this more general notion to account for syntactic restrictions that are common in description logics. For example, it allows for acyclic TBoxes, but also for the syntactic restrictions imposed to guarantee decidability of reasoning in $\mathcal{SROIQ}$ ontologies [4, 27]. It also allows for other languages not typically considered ontological, such as propositional formulas in conjunctive normal form (CNF) or constraint satisfaction problems, to name just two examples.

To aid the understanding of the notions presented here, we will use a very simple ontology language dealing with reachability in finite graphs. Specifically, given a countable set V of *vertices*, let $\mathcal{A} = \mathcal{C} = \{(v, w) \mid v, w \in V\}$; that is, axioms and consequences are given by ordered pairs of vertices (called *edges*), and $\mathfrak{O} = \mathscr{P}_{\mathsf{fin}}(\mathcal{A})$. Intuitively, an ontology is a finite graph, and $\mathcal{O} \models (v, w)$ iff w is reachable from v in the graph $\mathcal{O}$. For example, the graph $\mathcal{G}$ in Figure 1 (a) is one such ontology, and $\mathcal{G} \models (u, w)$ but $\mathcal{G} \not\models (w, u)$. Importantly, this is only an *example* of a very simple ontology language, but many of the intuitions obtained from it apply also to more complex langauges.

For any given ontology language, the main reasoning task is to decide *entailments*. Formally, given an ontology $\mathcal{O} \in \mathfrak{O}$ and a consequence $c \in \mathcal{C}$, we are interested in deciding whether $\mathcal{O} \models c$ holds. Depending on the specific language
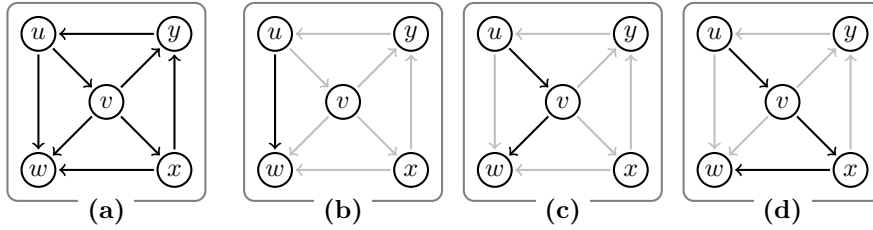
Fig. 1. The ontology $\mathcal{G}$ depicted as a graph **(a)**, and three justifications for the consequence (u, w) **(b)**–**(d)**.

considered, different methods can be developed to solve this reasoning task, and its computational complexity may vary. In fact, already within the family of description logics, we can find examples where the complexity of entailment checking varies from polynomial time up to doubly-exponential time. Moreover, there exist ontology languages with an undecidable entailment problem, but for the scope of this work we focus only on cases where this problem is decidable with complexity $\mathfrak{C}$. Once that adequate methods for deciding entailments have been developed, one is often interested in solving more complex reasoning tasks.

Axiom pinpointing is a non-standard reasoning task, which focuses on identifying the axioms that are responsible for a consequence to follow from an ontology. Formally, *axiom pinpointing* is the task of identifying one or all the justifications for a given consequence, where a justification is a minimal sub-ontology that still entails the consequence.

Definition 1 (justification). *Let $\mathcal{O}$ be an ontology and c a consequence such that $\mathcal{O} \models c$. The sub-ontology $\mathcal{M} \subseteq \mathcal{O}$ is called a* justification *for c w.r.t. $\mathcal{O}$ iff (i) $\mathcal{M} \models c$ and (ii) for every $\mathcal{M}' \subsetneq \mathcal{M}$, $\mathcal{M}' \not\models c$.*

Here we are using the standard name from description logics, but it is worth noting that justifications are known with different names by different communities. For example, they are also known as MUSes in SAT [36], MESCs in CSP [43], causes in DBs [41], and MIPS, MUPS, and MinAs in DLs [45, 51].

Consider for example the graph $\mathcal{G}$ from Figure 1 (a), which entails the consequence (u, w); that is, w is reachable from u. This consequence has three justifications, which are the three subgraphs (b)–(d) of the same figure. Note that there exist other sub-ontologies that still entail (u, w), but these contain at least one of the justifications depicted, and hence are not subset-minimal. In general, when an ontology corresponds to a graph, a justification for the consequence (u, v) is a simple path from u to v.

Technically, it is not difficult to come out with algorithms which compute one or all justifications. The simplest approach is to exploit the existing reasoners and find justifications through repeated entailment checks. This approach is known as *black-box* because it uses the reasoner without any modification. A black-box method for computing one justification is described in Algorithm 1. It tries to remove each axiom from the ontology, as long as it still entails the consequence.

Algorithm 1: Compute one justification

    Data: Ontology $\mathcal{O}$, consequence c
    Result: A justification $\mathcal{M}$ for c w.r.t. $\mathcal{O}$
1 $\mathcal{M} \leftarrow \mathcal{O}$
2 for $\alpha \in \mathcal{O}$ **do**
3   **if** $\mathcal{M} \setminus \{\alpha\} \models c$ **then**
4 $\mathcal{M} \leftarrow \mathcal{M} \setminus \{\alpha\}$

5 Return $\mathcal{M}$

An invariant of the **for** loop is that $\mathcal{M} \models c$, hence the resulting set satisfies the first condition from Definition 1. Moreover, by monotonicity of $\models$ the resulting set cannot have any superfluous axioms, meaning that it is a justification. The resulting justification obtained through this algorithm depends on the order in which the axioms are selected for removal, but independently of the order, each axiom needs to be tested exactly once.

To find *all* justifications, one can simply enumerate all possible sub-ontologies $\mathcal{M} \subseteq \mathcal{O}$ and verify for each of them that they entail c and that no strict subset entails c as well. Any such $\mathcal{M}$ is a justification. This method requires to check exponentially many times (i.e., once for each subset of $\mathcal{O}$) whether an ontology is a justification. Obviously, the number of checks can be greatly reduced simply by taking into account the monotonicity of the entailment relation: if a set $\mathcal{O}' \subseteq \mathcal{O}$ is such that $\mathcal{O}' \models c$, then there is no need to control any strict superset of $\mathcal{O}'$. Indeed, any such superset still entails c, but will not be minimal. Conversely, if $\mathcal{O}' \not\models c$, then no subset of $\mathcal{O}'$ can be a justification because it cannot entail c. Other optimisations can be considered. For example, [29] presents an enumeration method based on Reiter's Hitting Set Tree method [49], which guides the search for new justifications. However, as we will see later, one cannot avoid testing exponentially many sets. In the following section we analyse this, and other complexity issues in more detail.

3 Complexity

In this section, we consider computational complexity issues related to axiom pinpointing. We have already seen two basic algorithms for computing one or all justifications, but it remains unclear whether more advanced techniques can provide improvements in terms of worst-case complexity. Although our goal is to understand the computational problem of *finding* the justifications, to analyse the complexity we consider their decision variants.

Definition 2 (is-just,all-just). IS-JUST *is the problem of deciding, given an ontology $\mathcal{M}$ and a consequence c, whether $\mathcal{M}$ is a justification of c.* ALL-JUST *is the problem of deciding, given an ontology $\mathcal{O}$, a consequence c, and a set $\{\mathcal{M}_1, \ldots, \mathcal{M}_n\}$ of sub-ontologies of $\mathcal{O}$, whether $\mathcal{M}_1, \ldots, \mathcal{M}_n$ are* all *the justifications of c w.r.t. $\mathcal{O}$.*

3.1 One Justification

Recall that the first condition for a justification is that it still entails the consequence c. Hence, entailment is a sub-problem of IS-JUST. In particular this means that IS-JUST is necessarily at least as hard as entailment. More formally, if entailment is hard for the complexity class $\mathfrak{C}$, then IS-JUST must be $\mathfrak{C}$-hard as well. Note also that Algorithm 1 can be easily modified to decide IS-JUST as well: if the test in line 3 succeeds at any point, then the input ontology $\mathcal{O}$ is *not* a justification and the method can exit with failure; if the whole loop runs without exiting, then $\mathcal{O}$ was a justification. In the worst case, this algorithm has to make one call to the entailment test for each axiom in $\mathcal{O}$. This means that IS-JUST can be decided through polynomially many entailment tests; that is, IS-JUST is in $P^{\mathfrak{C}}$, if entailment is in $\mathfrak{C}$. Note that if $\mathfrak{C}$ is at least PSPACE, then the polynomial enumeration can be absorbed into the $\mathfrak{C}$ oracle.

Proposition 3. *If entailment is $\mathfrak{C}$-complete for $\mathfrak{C}$ at least PSPACE, then IS-JUST is $\mathfrak{C}$-complete as well.*

The consequence of this proposition is that there is no need to analyse the complexity of IS-JUST for expressive ontology languages with complex entailment relations. For example, in any propositionally closed description logic, where entailments w.r.t. a so-called TBox are EXPTIME-complete, IS-JUST is EXPTIME-complete as well. However, the black-box method for deciding this problem still leaves a gap when the complexity of entailment $\mathfrak{C}$ is below PSPACE; in those cases, $\mathfrak{C}$ is usually smaller than $P^{\mathfrak{C}}$ and hence $\mathfrak{C}$-hardness vs. in $P^{\mathfrak{C}}$ are not tight bounds. The only exception is if $\mathfrak{C} = P$, where we again have that $P^{\mathfrak{C}} = P$.

To understand the complexity of finding justifications, several ontology languages with lower-complexity entailments—mainly polynomial—have been studied. Unfortunately, the picture that arises from these studies is more complex than what is observed in Proposition 3. For example, there are ontology languages, such as the language of propositional formulas in CNF, whose entailment problem is NP-complete, but IS-JUST is D^P-complete [47].[1]

Before considering the computation of all justifications, note that there are many important variants of IS-JUST which may be considered. As we will see later, it is sometimes relevant to order the justifications according to some preference (e.g., size) and IS-JUST becomes the problem of deciding whether a sub-ontology is the *most preferred* justification in terms of this ordering. This, of course, requires additional tests, and the complexity may change accordingly. For a study on how the complexity is affected by the preference relation and the ontology language see [47]. In that work, several ontology languages whose entailments can be decided in polynomial time were analysed as a means to understand where the complexity of axiom-pinpoining tasks arises.

---

[1] D^P is the class of problems which can be solved by one NP and one coNP test. It is believed to be strictly contained in $\Delta_2^P = P^{NP}$.

3.2 All Justifications

If we want to solve ALL-JUST, we can once again consider the black-box algorithm described in the previous section. For each of the input sets $\mathcal{M}_1, \ldots, \mathcal{M}_n$, we verify in $P^{\mathfrak{C}}$ that they are justifications, and afterwards we need to verify that no other set is a justification. The latter task can be performed through a PSPACE enumeration of all possible sub-ontologies, checking for each of them that they are not a justification. Hence, overall, we can solve ALL-JUST with a PSPACE$^{\mathfrak{C}}$ algorithm. As in Proposition 3, the PSPACE base method can be absorbed into the oracle $\mathfrak{C}$ if the latter is at least PSPACE itself.

Proposition 4. *If entailment is $\mathfrak{C}$-complete for $\mathfrak{C}$ at least PSPACE, then ALL-JUST is $\mathfrak{C}$-complete as well.*

While this proposition is very similar to the case for one justification, the PSPACE base at the algorithm makes the gap between the lower and upper bounds, for ontology languages having simpler entailment tests, larger. Indeed, while for deciding whether a set is a justification we incur in a jump at most of one level in the polynomial hierarchy (if $\mathfrak{C}$ belongs to it), for ALL-JUST the increment goes all the way to the limit of this hierarchy at once. The gap can be reduced by considering the following idea: if an input is not an instance of ALL-JUST, we can verify it by guessing (in NP) a new set $\mathcal{N}$ which does not contain any of the $\mathcal{M}_i$s and verifying (in $\mathfrak{C}$) that $\mathcal{N} \models c$. Thus, ALL-JUST is in coNP$^{\mathfrak{C}}$.

The landscape of complexities also gets more complex in this case. There exist ontology languages with polynomial entailment problems for which ALL-JUST is polynomial, coNP-complete, or hard for an intermediate class, respectively. There is also a language with NP-complete entailment problem for which the exact complexity of ALL-JUST is unknown.

So far, the discussion has focused on deciding ALL-JUST, but in fact we are more interested in being able to enumerate all the justifications (rather than deciding whether a set of sub-ontologies is indeed the class of all of them). The first thing to notice when dealing with the enumeration of justifications is that it is impossible spend less than exponential time on this task. Indeed, even for the very simple ontology language that we are using as an example (that is, for reachability in graphs), it is easy to build an example of a consequence that has exponentially many justifications w.r.t. an ontology [12]. For example, there are 2^n justifications for (v, x) w.r.t. the graph from Figure 2. As before, this means that a full enumeration requires at least exponential time (but only polynomial space if found justifications are not preserved in memory). Thus, for expressive ontology languages, the black-box algorithm for computing all justifications is optimal in terms of complexity.

When dealing with enumeration problems, one can consider alternative complexity classes which take into account also the *total* number of answers, and the time needed to obtain new answers [28]. Alternatively, one may try to count the number of justifications available [60]. Also in this case, the enumeration complexity varies with the specific ontology language, and whether a specific
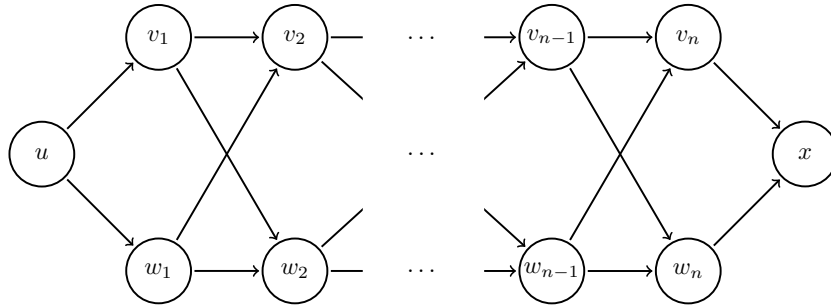
Fig. 2. A graph which produces exponentially many justifications.

ordering is requested. For ontology languages related to directed graphs and hypergraphs, the justifications can be enumerated with *polynomial delay*; that is, allowing polynomial time (on the size of the ontology) between two successive answers if no order is required. This enumeration still requires total exponential time, although polynomial in the number of justifications, if an order is required. For the light-weight description logic $\mathcal{EL}$, on the other hand, there is no algorithm that can enumerate all justification in total polynomial time. This means that there must exist consequences with polynomially many justifications, for which the enumeration problem requires super-polynomial time. For counting, in all the ontology languages with polynomial-time entailments studied so far (including our example language), the complexity is #P-complete while for propositional logic, where entailment is NP-complete, counting the number of justifications becomes #NP-complete.

4 Glass-box Algorithms

We have seen that black-box algorithms are complexity-optimal for expressive ontology languages, including the most common description logics—specifically, for any DL with value restrictions—assuming that an optimal implementation of the standard reasoning method exists. In addition, these algorithms are easy to implement and, as they only require repeated calls to an unmodified reasoner, can keep up to date with the newest optimisations and improvements. For that reason, they have become the approach of choice in those settings; for example, the explanation service in Protégé [32] is implemented as a black-box. Nonetheless, it is important to study more targeted approaches, which have the potential of behaving better in practice. This is particularly true for ontology languages whose complexity of reasoning is strictly below PSPACE.

The so-called *glass-box* approaches to axiom pinpointing are based on a modification of the original reasoning algorithms to be able to identify the justifications. In a nutshell, any reasoning algorithm must at some point consider the axioms in the ontology to decide whether the consequence follows. The idea of a glass-box algorithm is to trace which axioms were used during this process, yield-

ing candidates for justifications, or the whole class of justifications, depending on how the tracing mechanism is implemented.

In DLs, the first proposals for a glass-box algorithm for pinpointing were based on a modification of the tableaux-based reasoning method for reasoning in $\mathcal{ALC}$. This modification was originally designed for defeasible reasoning [5]. The idea was slowly improved and generalised to allow for more expressive languages [34, 44, 51] until a general approach for transforming tableaux-based reasoning methods into pinpointing algorithms was developed in [11]. The main drawback observed in these glass-box proposals is that, to guarantee that all the relevant axioms have been traced, important optimisations from the original decision algorithm have to be disabled. For example, for standard reasoning it suffices to halt the execution of the tableaux when the desired consequence is obtained, but for pinpointing one must continue until *all* possible derivations have been explored. As a consequence, these pinpointing extensions do not behave well in practice. An additional disadvantage highlighted in [11] is that the pinpointing extension of a terminating tableau algorithm is not guaranteed to terminate in general. On the other hand, some basic properties which are satisfied by all DL tableau algorithms suffice for guaranteeing termination. For a description of these properties, and how they apply to DLs, see [11].

Following a different idea, [10] introduced a pinpointing approach based on weighted automata. Briefly, the approach takes an existing automata-based reasoning method, and transforms it into a pinpointing approach by adding weights, belonging to the free distributive lattice, to all the transitions of this automaton. The *behaviour* of the weighted automaton obtained this way is a compact representation of all the justifications of the consequence. The main advantage of this approach is that it is optimal in terms of worst-case complexity. However, as in standard automata-based reasoning, the *best-case* complexity also matches the worst case, making it impractical for real applications, at least for standard ontology languages.[2]

To-date, the most successful approach which could be called glass-box is based on a translation of the execution of a consequence-based algorithm into a propositional formula. In essence, one tries to reduce an axiom pinpointing problem to pinpointing in a well-known ontology language for which efficient implementations exist. The original idea was introduced in [52, 53] for the lightweight DL $\mathcal{EL}$, but can be easily generalised to other consequence-based algorithms. Very briefly, consequence-based algorithms work by applying rules over the explicitly represented knowledge (originally, the ontology) to make some of its implicit consequences explicit. The method from [52] introduces a new propositional variable for each consequence, and a Horn clause simulating each possible rule application. Moreover, it adds the representative variable of each axiom of the original ontology.

---

[2] For some ontology languages, like the variant of linear temporal logic used in [10], this approach could be practical. However, at the moment there is no empirical evidence to support this claim.

As a very simple example, consider our graph ontology language, where the entailment relation is reachability. In this case, a consequence-based algorithm would only have the rule $\{(X,Y),(Y,Z)\} \rightarrow (X,Z)$ expressing that if we have the (explicit) knowledge that the ontology entails (X,Y) and (Y,Z), then we can derive that it also entails (X,Z). From this rule, and the graph in Figure 1, we obtain a set of Horn clauses which contains, among others,

$$x_{(u,v)} \wedge x_{(v,w)} \rightarrow x_{(u,w)}, \quad x_{(u,v)} \wedge x_{(v,x)} \rightarrow x_{(u,x)}, \quad x_{(u,v)} \wedge x_{(v,y)} \rightarrow x_{(u,y)},$$
$$x_{(u,x)} \wedge x_{(x,w)} \rightarrow x_{(u,w)}, \quad x_{(u,x)} \wedge x_{(x,y)} \rightarrow x_{(u,y)}, \quad x_{(u,y)} \wedge x_{(y,u)} \rightarrow x_{(u,u)}.$$

The ontology is then represented through the variables

$$x_{(u,v)}, \quad x_{(u,w)}, \quad x_{(v,w)}, \quad x_{(v,x)}, \quad x_{(v,y)}, \quad x_{(x,w)}, \quad x_{(x,y)}, \quad x_{(y,u)}.$$

The conjunction of all these elements yields a Horn formula F that entails the variable representing each relevant consequence. Hence, for example $F \models x_{(u,w)}$.

In order to find the justifications for a consequence c w.r.t. the ontology $\mathcal{O}$, one needs only to enumerate the justifications for the consequence variable x_c w.r.t. the set of Horn clauses F, with the difference that the Horn clauses simulating the execution of the rules are always present; that is, a justification may only remove from the formula variables representing the original ontology.

In essence, what this translation does is to reduce axiom pinpointing in an arbitrary ontology language which accepts consequence-based reasoning into axiom pinpointing in propositional logic (or, indeed, in Horn logic). The advantage is that we can then focus on the development of highly-efficient pinpointing tools for this very specific ontology language, which can be further optimised taking the shape of the obtained problem into account. Indeed, this idea has given rise to many axiom pinpointing tools for $\mathcal{EL}^+$; an extension of $\mathcal{EL}$ for which reasoning remains tractable [3]. Out of these systems [2, 38, 39, 53], the most efficient, and only one currently capable of computing all the justifications for all the 5415670 atomic entailments from the very large ontology SNOMED—is PULi [30].

The Grey-Box Approach

Before looking into different applications and variations of axiom pinpointing, we would be amiss to ignore a popular approach for finding *one* justification, which combines the glass-box tracing approach with the black-box method from Algorithm 1; hence, it is usually called *grey-box*. The main idea, as with the glass-box approach, is to trace all the axioms used to prove the entailment of a consequence, through the application of an algorithm; for example, in a tableau-based or consequence-based algorithm. However, in contrast to the case for finding all justifications, we stop once the consequence is derived. This yields a set of axioms that is not guaranteed to be a justification, but from which the consequence still follows. That is, it might still require some pruning of superfluous axioms to become a justification. On the other hand, this limited tracing only imposes a minimal overhead to the original decision algorithm, and does not affect the

existing optimisations. Moreover, the resulting set tends to be much smaller than the original ontology, and very close to a justification. Once this smaller ontology is found, it can be minimised by a call to the black-box method, yielding a justification. Note that this second step needs to check potentially much less axioms than a direct use of Algorithm 1 on the original ontology.

This grey-box approach was used originally by the CEL system [9] to find justifications efficiently for $\mathcal{EL}$, where the goal was not to find them all, but only a relevant subset. A similar idea is followed in the database community to trace the facts which provide an answer to a query. Although, to the best of our knowledge, there is no implementation of the grey-box approach for more complex ontology languages, or based on other kinds of algorithms, it should be relatively straightforward to modify a tableaux-based tool for this purpose.[3]

An important drawback of the grey-box algorithm as described above is that the sub-ontology obtained by the tracing step is only guaranteed to contain *one* justification. This means that if one is interested in potentially finding more justifications, then the whole process needs to be started anew for each successive solution. To alleviate this problem, some work has focused on computing *justification-preserving modules*; that is, sub-ontologies which are still guaranteed to contain all the justifications for a given consequence. Ideally, these modules should be fast to compute and still be small enough to guarantee that all justifications can be extracted from them following, e.g. the black-box approach, efficiently. It has been observed that different modularization methods yield adequate solutions in this direction. Usually, these modules are based on syntactic or semantic relationships between the axioms of the ontology [20, 57, 58]. More recently, a modularization method based on a modification of a consequence-based reasoning algorithm was proposed first for formulas in propositional logic, and later extended to description logics [33, 40, 42, 46]. These approaches modify the tracing technique of the glass-box method. As usual, they keep track of the axioms that are being used during the execution of the algorithm for deriving new knowledge, but instead of distinguishing between different derivation paths, they are all stored in one single set. This simple modification avoids the problems with termination of the original glass-box approach, but provides more information than just following one derivation as in the description of the grey-box approach above. In empirical analyses, this approach has shown to be very helpful for solving axiom-pinpointing related tasks.

5 Applications and Variations

The task of axiom pinpointing as defined in Section 2 has been extended to cover a large class of variations, and can be used to solve different reasoning tasks.

The simplest generalisation which we can consider is to allow some parts of the ontology to be fixed; that is, consider a justification to be a minimal subontology $\mathcal{M}$ that, when added to a fixed ontology $\mathcal{O}_{\mathsf{f}}$, entails the consequence.

---

[3] Although a grey-box algorithm was described for the DL reasoner Pellet [56], to our knowledge the pinpointing tools in this reasoner are fully black-box.

This application makes sense in the context of debugging, when we trust some axioms, and we do not want to look into them when trying to understand or correct an erroneous consequence. We have also observed its use in the reduction from consequence-based algorithms to axiom pinpointing in SAT in Section 4. Further generalising this idea, one can consider the atomic elements to be not axioms, but rather sets of axioms. In the literature, these have been called group-MUSes [36] or *contexts* [7]. Note that these contexts may appear implicitly in several applications. For example, in consequence-based algorithms [54, 55], a common pre-processing step is to modify the ontology into a given normal form. In those cases, several separated axioms may have been produced by a single original statement, and thus should all be considered together. Another setting where considering sets of axioms makes sense is to find classes of modules of different kinds where a consequence can be derived. In this case, we can group axioms according to the atomic decomposition of the ontology [61].

Returning justifications at the granularity of (sets of) axioms from the original ontology is not always satisfactory. These may hide some relevant information or, conversely, be too complex to be understood or managed adequately by domain experts. Hence some research—specially in DLs—has considered building different representations. One direction produces so-called *laconic* and *precise* justifications [25] which, from a very abstract point of view, provide only the specific pieces of the axioms which are really responsible for the consequence, removing any superfluous information. The other direction combines several axioms into *lemmas*, which remove excessive detail and become easier to read [26]. More recently, single justifications of partial entailments have been used to provide understandable explanations for a derivation [1].

In a similar direction, more recent work has focused on the goal of minimally modifying ontologies through weakening [8, 59]. Very briefly, after a justification has been found, its axioms are modified to weaker ones in order to get rid of the consequence. Hence, this process goes one step beyond pinpointing by further identifying the strongest possible weakenings which yield the desired result. Note that axiom pinpointing is a special step of this idea, where the only possible weakening an axiom is to replace it by a tautology.

Regarding the computation of all justifications, it is sometimes convenient to try to enumerate them in a specific order. For example, one may want to observe the smallest (w.r.t. cardinality) justifications first, or alternatively have a pre-specified order for accessing them. Obviously, once that we can compute all the justifications, it is also possible to order them before showing them to the user, but this brings a large overhead in general. To understand the issue better, the complexity of enumerating justifications w.r.t. some natural orderings has also been studied, with the unsurprising result that it depends not only on the ontology language, but also on the chosen order [47].

Axiom pinpointing is not only useful for understanding and potentially correcting consequences from an ontology, but it has also found applications aiding different kinds of supplemental and non-standard reasoning tasks. Perhaps the most studied and applied to date is related to the representation and handling

of uncertain knowledge. When dealing with probabilities, several semantics and applications use axiom pinpointing, albeit often implicitly. In probabilistic logic programming [48], the probability of an inference is given by the probabilities of combinations of facts that entail it (together with a fixed program) which correspond to the justifications of an ontology. This idea was generalised to description logics under the so-called Disponte semantics [50]. In these approaches, it is assumed that all the uncertain elements are probabilistically independent. To lift this assumption, newer approaches include a Bayesian network which expresses the joint probability distribution of the axioms (or more in general, contexts) of the ontology [16,19,21]. For all these approaches, axiom pinpointing provides a helpful intermediate step which can be exploited also for approximating solutions. For example, finding the justification with the highest probability can often yield a good approximation for the full probability. In terms of possibility theory, when using the standard min-max semantics, it is well known that it suffices to find one justification with the highest possibility degree [13].

Another application is in the context of access control, where one wants to provide only partial views to an ontology to different users [6]. Here, axiom pinpointing is not only useful to find out the access levels (that is, the contexts) where a consequence is derivable, but also to suggest changes in the access level of axioms in order to hide or open implicit consequences to some users [31].

There is a current interest in being able to reason in the presence of inconsistencies or errors in an ontology [14, 15, 35, 37]. In this case, we are interested in finding so-called *repairs*, which correspond to the dual notion of a justification; that is, maximal subontologies which do not entail the consequence under consideration. Axiom pinpointing comes into place in this case in two different ways. On the one hand, repairs can be computed from justifications through a hitting set computation. On the other hand, most of the techniques developed for axiom pinpointing (and in particular, the black-box methods) can be easily adapted to compute repairs directly, usually leading to the same complexity bounds [45]. Still, keeping this dual view is often helpful for finding new research problems.

To finish this brief overview on applications, we mention *provenance* [18, 22–24], which has a similar motivation, but different development, to axiom pinpointing. Originally developed for databases, and later extended to DLs, the goal of provenance is to track the origins for an answer to a query, in terms of the facts used to derive it. The main difference with axiom pinpointing is that with provenance one is also interested in finding how often an axiom (or group of axioms) is used in the derivation, and that minimality is replaced by a weaker notion, where all axioms need to be relevant, but they may be superfluous.

6 Conclusions

We have presented a general overview on axiom pinpointing: what it is about, how it is usually tackled, and its main applications and variations. We insist here that, although most of the terminology used was originally introduced in the context of description logics, we have tried to keep the presentation as general

as possible to include the notions developed in many other areas of knowledge representation and reasoning. This general presentation is a first step towards a unified view of the problem of finding the axiomatic causes for consequences in different languages, which should ultimately lead towards an exchange of techniques, problems, and tools between areas.

Clearly, the work on axiom pinpointing does not finish with this content. There exist many applications and variations to the problem which have not been mentioned, or commented only briefly, and which would require much more space to explore in detail. Even for the ones which were more thoroughly commented, there are usually many open problems which still require some additional work. Moreover, even at its basic state, axiom pinpointing is not fully resolved. For example, in terms of its practical application, the development of efficient tools for more expressive languages—or, even better, for general use—is still missing. A new line of research was also introduced very recently with the notion of strong inconsistency [17], which allows extending some of the notions of axiom pinpointing to non-monotonic logics as well.

There are many specialised techniques developed in areas other than description logics (e.g., databases, CSP, and SAT). Some of these techniques are general enough to allow their applicability across other areas with minor modifications. Alternatively, it may be possible to reduce problems between areas as done in the reduction to propositional logic described here. This allows to focus on one specific problem, and using advanced engineering techniques for its effective solution.

References

1. Alrabbaa, C., Baader, F., Borgwardt, S., Koopmann, P., Kovtunova, A.: Finding small proofs for description logic entailments: Theory and practice. In: Albert, E., Kovács, L. (eds.) Proc. of LPAR 2020. EPiC Series in Computing, vol. 73, pp. 32–67. EasyChair (2020), https://easychair.org/publications/paper/qgX6
2. 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: Proceedings of the 19th International Conference on Theory and Applications of Satisfiability Testing (SAT 2016). LNCS, vol. 9710, pp. 521–530. Springer (2016). https://doi.org/10.1007/978-3-319-40970-2_32
3. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Kaelbling, L., Saffiotti, A. (eds.) Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05). pp. 364–369. Professional Book Center (2005)
4. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, second edn. (2007)
5. Baader, F., Hollunder, B.: Embedding defaults into terminological knowledge representation formalisms. J. of Automated Reasoning **14**(1), 149–180 (1995). https://doi.org/10.1007/BF00883932
6. Baader, F., Knechtel, M., Peñaloza, R.: A generic approach for large-scale ontological reasoning in the presence of access restrictions to the ontology's axioms. In: Proc. 8th Int. Semantic Web Conf. (ISWC'09). LNCS, vol. 5823, pp. 49–64. Springer (2009)

7. Baader, F., Knechtel, M., Peñaloza, R.: Context-dependent views to axioms and consequences of semantic web ontologies. Journal of Web Semantics **12**, 22–40 (2012). https://doi.org/10.1016/j.websem.2011.11.006

8. Baader, F., Kriegel, F., Nuradiansyah, A., Peñaloza, R.: Making repairs in description logics more gentle. In: Proc. of 16th International Conference on Principles of Knowledge Representation and Reasoning (KR 2018). pp. 319–328. AAAI Press (2018), https://aaai.org/ocs/index.php/KR/KR18/paper/view/18056

9. Baader, F., Lutz, C., Suntisrivaraporn, B.: CEL—a polynomial-time reasoner for life science ontologies. In: Furbach, U., Shankar, N. (eds.) Proc. 3rd Int. Joint Conf. on Automated Reasoning (IJCAR'06). LNCS, vol. 4130, pp. 287–291. Springer (2006)

10. Baader, F., Peñaloza, R.: Automata-based axiom pinpointing. J. of Automated Reasoning **45**(2), 91–129 (2010). https://doi.org/10.1007/s10817-010-9181-2

11. Baader, F., Peñaloza, R.: Axiom pinpointing in general tableaux. Journal of Logic and Computation **20**(1), 5–34 (2010). https://doi.org/10.1093/logcom/exn058

12. Baader, F., Peñaloza, R., Suntisrivaraporn, B.: Pinpointing in the description logic EL+. In: Proceedings of the 30th Annual German Conference on AI (KI'07). LNCS, vol. 4667, pp. 52–67. Springer (2007). https://doi.org/10.1007/978-3-540-74565-5_7

13. Benferhat, S., Bouraoui, Z.: Min-based possibilistic dl-lite. J. Log. Comput. **27**(1), 261–297 (2017). https://doi.org/10.1093/logcom/exv014

14. Bertossi, L.E., Hunter, A., Schaub, T. (eds.): Inconsistency Tolerance, LNCS, vol. 3300. Springer (2005). https://doi.org/10.1007/b104925

15. Bienvenu, M., Bourgaux, C.: Inconsistency-tolerant querying of description logic knowledge bases. In: 12th International Reasoning Web Summer School, Tutorial Lectures. LNCS, vol. 9885, pp. 156–202. Springer (2016). https://doi.org/10.1007/978-3-319-49493-7_5

16. Botha, L., Meyer, T., Peñaloza, R.: The bayesian description logic BALC. In: Proceedings of the 16th European Conference on Logics in Artificial Intelligence (JELIA'19) (2019), to appear

17. Brewka, G., Thimm, M., Ulbricht, M.: Strong inconsistency. Artificial Intelligence **267**, 78 – 117 (2019). https://doi.org/https://doi.org/10.1016/j.artint.2018.11.002

18. Calvanese, D., Lanti, D., Ozaki, A., Peñaloza, R., Xiao, G.: Enriching ontology-based data access with provenance. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019. pp. 1616–1623. ijcai.org (2019). https://doi.org/10.24963/ijcai.2019/224

19. Ceylan, İ.İ., Peñaloza, R.: The bayesian ontology language BEL. J. of Automated Reasoning **58**(1), 67–95 (2017). https://doi.org/10.1007/s10817-016-9386-0

20. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Just the right amount: extracting modules from ontologies. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007. pp. 717–726. ACM (2007). https://doi.org/10.1145/1242572.1242669

21. d'Amato, C., Fanizzi, N., Lukasiewicz, T.: Tractable reasoning with bayesian description logics. In: Proceedings of the Second International Conference on Scalable Uncertainty Management (SUM 2008). LNCS, vol. 5291, pp. 146–159. Springer (2008). https://doi.org/10.1007/978-3-540-87993-0_13

22. Dividino, R., Sizov, S., Staab, S., Schueler, B.: Querying for provenance, trust, uncertainty and other meta knowledge in {RDF}. Web Semantics: Science, Services and Agents on the World Wide Web **7**(3), 204 – 219 (2009), the Web of Data

23. Geerts, F., Unger, T., Karvounarakis, G., Fundulaki, I., Christophides, V.: Algebraic structures for capturing the provenance of SPARQL queries. J. ACM **63**(1), 7 (2016)

24. Green, T.J., Karvounarakis, G., Tannen, V.: Provenance semirings. In: Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China. pp. 31–40 (2007)

25. Horridge, M., Parsia, B., Sattler, U.: Laconic and precise justifications in OWL. In: Sheth, A., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) Proc. 7th Int. Semantic Web Conf. (ISWC'08). LNCS, vol. 5318, pp. 323–338. Springer (2008)

26. Horridge, M., Parsia, B., Sattler, U.: Lemmas for justifications in OWL. In: Grau, B.C., Horrocks, I., Motik, B., Sattler, U. (eds.) Proceedings of the 22nd International Workshop on Description Logics (DL 2009). CEUR Workshop Proceedings, vol. 477. CEUR-WS.org (2009), http://ceur-ws.org/Vol-477/paper_24.pdf

27. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06). pp. 57–67. AAAI Press (2006)

28. Johnson, D.S., Papadimitriou, C.H., Yannakakis, M.: On generating all maximal independent sets. Inf. Process. Lett. **27**(3), 119–123 (1988). https://doi.org/10.1016/0020-0190(88)90065-8, https://doi.org/10.1016/0020-0190(88)90065-8

29. Kalyanpur, A., Parsia, B., Sirin, E., Hendler, J.A.: Debugging unsatisfiable classes in OWL ontologies. Journal of Web Semantics **3**(4), 268–293 (2005). https://doi.org/10.1016/j.websem.2005.09.005

30. Kazakov, Y., Skocovský, P.: Enumerating justifications using resolution. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) Proceedings of the 9th International Joint Conference on Automated Reasoning IJCAR 2018. Lecture Notes in Computer Science, vol. 10900, pp. 609–626. Springer (2018). https://doi.org/10.1007/978-3-319-94205-6_40, https://doi.org/10.1007/978-3-319-94205-6_40

31. Knechtel, M., Peñaloza, R.: A generic approach for correcting access restrictions to a consequence. In: Proceedings of the 7th Extended Semantic Web Conference, ESWC 2010, Part I. Lecture Notes in Computer Science, vol. 6088, pp. 167–182. Springer (2010). https://doi.org/10.1007/978-3-642-13486-9_12, https://doi.org/10.1007/978-3-642-13486-9_12

32. Knublauch, H., Fergerson, R.W., Noy, N.F., Musen, M.A.: The Protégé OWL Plugin: An open development environment for Semantic Web applications. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) Proc. 3rd Int. Semantic Web Conf. (ISWC'04). LNCS, vol. 3298, pp. 229–243. Springer (2004)

33. 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.) Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing - SAT 2006. Lecture Notes in Computer Science, vol. 4121, pp. 22–35. Springer (2006). https://doi.org/10.1007/11814948_4, https://doi.org/10.1007/11814948_4

34. Lee, K., Meyer, T.A., Pan, J.Z., Booth, R.: Computing maximally satisfiable terminologies for the description logic ALC with cyclic definitions. In: Parsia, B., Sattler, U., Toman, D. (eds.) Proceedings of the 2006 International Workshop on Description Logics (DL'06). CEUR Workshop Proceedings, vol. 189. CEUR-WS.org (2006), http://ceur-ws.org/Vol-189/submission_29.pdf

35. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Inconsistency-tolerant semantics for description logics. In: Hitzler, P., Lukasiewicz, T. (eds.) Proc. 4th Int. Conf. on Web Reasoning and Rule Systems (RR 2010). LNCS, vol. 6333, pp. 103–117. Springer (2010). https://doi.org/10.1007/978-3-642-15918-3_9

36. Liffiton, M.H., Sakallah, K.A.: Algorithms for computing minimal unsatisfiable subsets of constraints. J. of Automated Reasoning **40**(1), 1–33 (2008). https://doi.org/10.1007/s10817-007-9084-z

37. Ludwig, M., Peñaloza, R.: Error-tolerant reasoning in the description logic EL. In: Fermé, E., Leite, J. (eds.) Proceedings of the 14th European Conference on Logics in Artificial Intelligence (JELIA 2014). LNCS, vol. 8761, pp. 107–121. Springer (2014). https://doi.org/10.1007/978-3-319-11558-0_8

38. Manthey, N., Peñaloza, R., Rudolph, S.: Efficient axiom pinpointing in EL using SAT technology. In: Lenzerini, M., Peñaloza, R. (eds.) Proceedings of the 29th International Workshop on Description Logics. CEUR Workshop Proceedings, vol. 1577. CEUR-WS.org (2016)

39. Manthey, N., Peñaloza, R., Rudolph, S.: Satpin: Axiom pinpointing for lightweight description logics through incremental sat. Künstliche Intelligenz (2020). https://doi.org/https://doi.org/10.1007/s13218-020-00669-4

40. Marques-Silva, J., Ignatiev, A., Morgado, A., Manquinho, V.M., Lynce, I.: Efficient autarkies. In: Schaub, T., Friedrich, G., O'Sullivan, B. (eds.) Proceedings of the 21st European Conference on Artificial Intelligence(ECAI 2014). Frontiers in Artificial Intelligence and Applications, vol. 263, pp. 603–608. IOS Press (2014). https://doi.org/10.3233/978-1-61499-419-0-603, https://doi.org/10.3233/978-1-61499-419-0-603

41. Meliou, A., Gatterbauer, W., Halpern, J.Y., Koch, C., Moore, K.F., Suciu, D.: Causality in databases. IEEE Data Engineering Bulletin **33**(3), 59–67 (2010), http://sites.computer.org/debull/A10sept/suciu.pdf

42. Mencía, C., Kullmann, O., Ignatiev, A., Marques-Silva, J.: On computing the union of muses. In: Janota, M., Lynce, I. (eds.) Proceedings of the 22nd International Conference on Theory and Applications of Satisfiability Testing. Lecture Notes in Computer Science, vol. 11628, pp. 211–221. Springer (2019). https://doi.org/10.1007/978-3-030-24258-9_15, https://doi.org/10.1007/978-3-030-24258-9_15

43. Mencía, C., Marques-Silva, J.: Efficient relaxations of over-constrained csps. In: Proceedings of the 26th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2014). pp. 725–732. IEEE Computer Society (2014). https://doi.org/10.1109/ICTAI.2014.113

44. Meyer, T.A., Lee, K., Booth, R., Pan, J.Z.: Finding maximally satisfiable terminologies for the description logic ALC. In: Proceedings of The Twenty-First National Conference on Artificial Intelligence (AAAI'06). pp. 269–274. AAAI Press (2006), http://www.aaai.org/Library/AAAI/2006/aaai06-043.php

45. Peñaloza Nyssen, R.: Axiom pinpointing in description logics and beyond. Ph.D. thesis, Technische Universität Dresden, Germany (2009), http://nbn-resolving.de/urn:nbn:de:bsz:14-qucosa-24743

46. Peñaloza, R., Mencía, C., Ignatiev, A., Marques-Silva, J.: Lean kernels in description logics. In: Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., Hartig, O. (eds.) Proceeding of the 14th Semantic Web Conference (ESWC 2017). LNCS, vol. 10249, pp. 518–533 (2017). https://doi.org/10.1007/978-3-319-58068-5_32

47. Peñaloza, R., Sertkaya, B.: Understanding the complexity of axiom pinpointing in lightweight description logics. Artificial Intelligence **250**, 80–104 (2017). https://doi.org/10.1016/j.artint.2017.06.002

48. Raedt, L.D., Kimmig, A., Toivonen, H.: Problog: A probabilistic prolog and its application in link discovery. In: Veloso, M.M. (ed.) Proc. 20th Int.

Joint Conf. on Artificial Intelligence (IJCAI'07). pp. 2462–2467. IJCAI (2007), http://ijcai.org/Proceedings/07/Papers/396.pdf

49. Reiter, R.: A theory of diagnosis from first principles. Artificial Intelligence **32**(1), 57–95 (1987). https://doi.org/10.1016/0004-3702(87)90062-2

50. Riguzzi, F., Bellodi, E., Lamma, E., Zese, R.: Probabilistic description logics under the distribution semantics. Semantic Web **6**(5), 477–501 (2015). https://doi.org/10.3233/SW-140154

51. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03). pp. 355–360. Morgan Kaufmann Publishers Inc. (2003)

52. Sebastiani, R., Vescovi, M.: Axiom pinpointing in lightweight description logics via Horn-SAT encoding and conflict analysis. In: Proc. of 22nd International Conference on Automated Deduction (CADE-22). LNCS, vol. 5663, pp. 84–99. Springer (2009). https://doi.org/10.1007/978-3-642-02959-2_6, https://doi.org/10.1007/978-3-642-02959-2

53. Sebastiani, R., Vescovi, M.: Axiom pinpointing in large EL+ ontologies via SAT and SMT techniques. Disi technical report, University of Trento (2015)

54. Simancik, F., Motik, B., Horrocks, I.: Consequence-based and fixed-parameter tractable reasoning in description logics. Artificial Intelligence **209**, 29–77 (2014)

55. Simančík, F.: Consequence-based reasoning for ontology classification. Ph.D. thesis, University of Oxford (2013), https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.581368

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

57. Stuckenschmidt, H., Parent, C., Spaccapietra, S. (eds.): Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization, Lecture Notes in Computer Science, vol. 5445. Springer (2009). https://doi.org/10.1007/978-3-642-01907-4, https://doi.org/10.1007/978-3-642-01907-4

58. Suntisrivaraporn, B.: Module extraction and incremental classification: A pragmatic approach for ontologies. In: Proceedings of the 5th European Semantic Web Conference (ESWC 2008). LNCS, vol. 5021, pp. 230–244. Springer (2008). https://doi.org/10.1007/978-3-540-68234-9_19

59. Troquard, N., Confalonieri, R., Galliani, P., Peñaloza, R., Porello, D., Kutz, O.: Repairing ontologies via axiom weakening. In: McIlraith, S.A., Weinberger, K.Q. (eds.) Proceedings of The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI'18). pp. 1981–1988. AAAI Press (2018), https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17189

60. Valiant, L.G.: The complexity of computing the permanent. Theor. Comput. Sci. **8**, 189–201 (1979). https://doi.org/10.1016/0304-3975(79)90044-6, https://doi.org/10.1016/0304-3975(79)90044-6

61. Vescovo, C.D., Klinov, P., Parsia, B., Sattler, U., Schneider, T., Tsarkov, D.: Empirical study of logic-based modules: Cheap is cheerful. In: Proceedings of the 12th International Semantic Web Conference, Part I. LNCS, vol. 8218, pp. 84–100. Springer (2013). https://doi.org/10.1007/978-3-642-41335-3_6