Explaining Axiom Pinpointing

Rafael Peñaloza[0000−0001−9236−9873]

University of Milano-Bicocca, Milano, Italy
rafael.penaloza@unimib.it

Abstract. Axiom pinpointing refers to the task of highlighting (or pinpointing) the axioms in an ontology that are responsible for a given consequence to follow. This is a fundamental task for understanding and debugging very large ontologies. Although the name axiom pinpointing was only coined in 2003, the problem itself has a much older history, even if considering only description logic ontologies. In this work, we try to explain axiom pinpointing: what it is; how it works; how it is solved; and what it is useful for. To answer this questions, we take a historic look at the field, focusing mainly on description logics, and the specific contributions stemming from one researcher, who started it all in more than one sense.

1 Introduction

One important aspect behind any (artifically) intelligent application is the availability of knowledge about the domain, which can be accessed and used effectively. In logic-based knowledge representation, this domain knowledge is expressed through a collection of logical constraints (or *axioms*) that limit how the different terms under consideration are interpreted, and related to each other. To abstract from the specific logical language used to express these constraints, we call any such representation an *ontology*.

Description logics (DLs) are a family of knowledge representation formalisms that have been successfully applied to represent the knowledge of several application domains. The family contains several different logical languages that are distinguished by their expressivity and the computational complexity of reasoning over them. They range from the extremely inexpressive DL-Lite [24] at the basis of ontology-based data access [72], to the expressive $\mathcal{SROIQ}(\mathcal{D})$ [36] underlying the standard ontology language for the semantic web OWL2 [50]. In between these two, many other DLs exist. Another prominent example is the light-weight DL $\mathcal{EL}$ [1], which allows for polynomial time reasoning [11], and is used for many ontologies within the bio-medical domain. In particular, we can mention Snomed CT, an ontology providing a unified nomenclature for clinical terms in medicine composed of approximately half a million axioms [59].

It is hardly surprising that ontology engineering—the task of building an ontology—is a costly and error-prone task. As the size of an ontology increases, it becomes harder to have a global perspective of its constraints and their relationships. For this reason, it becomes more likely to be surprised by some

of the consequences that can be derived from them. A classical example arose with SNOMED, from which at some point it was possible to derive that every amputation of a finger was in fact an amputation of a hand. Finding the six axioms—out of approximately 500,000—that cause this error without the help of an automated method would have been almost impossible.

Here is where axiom pinpointing comes into play. Essentially, axiom pinpointing refers to the task of highlighting (or pinpointing) the specific axioms that are responsible for a consequence to follow from an ontology. Considering that the underlying ontology language is monotonic, this task corresponds to finding classes of minimal subontologies still entailing the consequence. The term itself was coined by Schlobach and Cornet in 2003, in a work that triggered several follow-up approaches dealing with logics of varying expressivity. However, the underlying method was proposed almost a decade earlier by Baader and Hollunder in the context of default reasoning.[1]

By 2006, the field of axiom pinpointing was starting to mature to a point that begged for general solutions, beyond the specific attempts known at the time. With the advice of Franz, I embarked then on a trip to produce these general solutions. That trip was supposed to end in 2009 with the defense of my dissertation, and search for new research topics. However, the repercussions of the work continue today.

In this paper, we attempt to present a historical perspective on axiom pinpointing, explaining the existing methods, its applications, and newer extensions that have been developed throughout the years. We note that although axiom pinpointing has been studied—with different names—for other knowledge representation formalisms, and by several people, our focus here is constrained to DLs and specifically to the contributions that Franz made to the field either directly, or indirectly through the people working in his group. This work is intended as a basic, non-technical introduction to the field, but relevant references are provided for the reader interested in a deeper understanding.

2 Description Logics in a Nutshell

We briefly introduce the notions on description logics (DLs) [2, 6] that will be useful for understanding the rest of this work. Although there exist more complex and expressive DLs, here we will focus on the basic $\mathcal{ALC}$ [65], which is the smallest propositionally-closed DL, and the light-weight $\mathcal{EL}$ [1].

DLs are knowledge representation formalisms that are specially targeted to encode the terminological knowledge of an application domain. Their main ingredients are *individuals*, *concepts* (that is, classes of individuals), and *roles* providing relationships between individuals. Formally, they are nullary, unary, and binary predicates from first-order logic, respectively. The knowledge of the

[1] I am an informal (some will say impolite) Mexican, who insists on using the term *Franz* when referring to Franz Baader. I will do this often from now on. Please bear with me.

domain is represented via a set of *axioms* that restrict the way in which those ingredients can be interpreted.

Let N_I, N_C, and N_R be three mutually disjoint sets of *individual*, *concept*, and *role names*, respectively. $\mathcal{ALC}$ *concepts* are built following the syntactic rule $C ::= A \mid \neg C \mid C \sqcap C \mid \exists r.C$, where $A \in N_C$ and $r \in N_R$. A *general concept inclusion* (GCI) is an expression of the form $C \sqsubseteq D$, where C and D are concepts. An *assertion* is an expression of the form $C(a)$ (called a *concept assertion*) or $r(a, b)$ (*role assertion*), where $a, b \in N_I$, C is a concept, and $r \in N_R$. Historically, DLs separate the knowledge in *terminological* and *assertional knowledge*. The former describes general relations between the terms, encoded via a *TBox*, which is a finite set of GCIs. The latter refers to the knowledge about the individuals, which is encoded in a finite set of assertions called an *ABox*. Here, we use the general term *axiom* to refer to both GCIs and assertions. An *ontology* is a finite set of axioms; that is, the union of an ABox and a TBox.

The semantics of $\mathcal{ALC}$ is defined in terms of *interpretations*. These are tuples of the form $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set called the *domain*, and $\cdot^{\mathcal{I}}$ is the *interpretation function* that maps every individual name $a \in N_I$ to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, every concept name $A \in N_C$ to a set $A \subseteq \Delta^{\mathcal{I}}$, and every role name $r \in N_R$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. This interpretation function is extended to arbitrary concepts in the usual manner; that is, $(\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$, $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$, and $(\exists r.C)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid \exists y \in C^{\mathcal{I}}.(x, y) \in r^{\mathcal{I}}\}$. The interpretation $\mathcal{I}$ *satisfies* the GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; the assertion $C(a)$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$; and the assertion $r(a, b)$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$. $\mathcal{I}$ is a *model* of the ontology $\mathcal{O}$, denoted by $\mathcal{I} \models \mathcal{O}$, iff it satisfies all the axioms in $\mathcal{O}$.

It is often useful to consider abbreviations of complex concepts. We define $\bot := A \sqcap \neg A$ for an arbitary $A \in N_C$; $\top := \neg\bot$; $C \sqcup D := \neg(\neg C \sqcap \neg D)$; and $\forall r.C := \neg\exists r.\neg C$. In particular, in Section 4.1 dealing with these abbreviations is fundamental for the algorithm.

Once that the knowledge of a domain has been encoded in an ontology, we are interested in *reasoning*; that is, making inferences about this knowledge, which are implicit in the ontology. The most basic reasoning task is to decide *consistency*; i.e., whether a given ontology has at least one model. Other important reasoning problems are: *subsumption* (does $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ hold in every model $\mathcal{I}$ of $\mathcal{O}$?); *instance checking* (does $a^{\mathcal{I}} \in A^{\mathcal{I}}$ hold in every model $\mathcal{I}$ of $\mathcal{O}$?); and *classification* (finding all the subsumption relations between concept names appearing in $\mathcal{O}$). It has been shown that consistency, subsumption, and instance checking w.r.t. $\mathcal{ALC}$ ontologies is ExpTime-complete [31, 63]. Since the number of concept names appearing in $\mathcal{O}$ is bounded by the size of $\mathcal{O}$, it also follows that the ontology can be classified in exponential time.

The light-weight DL $\mathcal{EL}$ is the sublogic of $\mathcal{ALC}$ that disallows the negation constructor $\neg$, but includes the special concept $\top$ that specifies a tautology; that is $\top^{\mathcal{I}} := \Delta^{\mathcal{I}}$ for all interpretations $\mathcal{I}$ (compare to the abbreviation defined before). All other definitions are analogous as for $\mathcal{ALC}$. Interestingly, every $\mathcal{EL}$ ontology is consistent, and hence the consistency problem becomes trivial; more-

over, subsumption and instance checking are decidable in polynomial time via a so-called completion algorithm that in fact classifies the whole ontology.

3 What is Axiom Pinpointing?

Although the term "Axiom Pinpointing" was originally coined in the context of description logics, and the main focus in this work is on its development in DLs as well, closely related problems have been also studied—with different names—in other areas such as databases [47], propositional satisfiability [44], or constraint satisfaction problems [48]. Most of the basic ideas can, in fact, be traced to Raymond Reiter [61]. For that reason, we introduce the problem in the most general terms possible, trying to preserve readability.

We consider an abstract *ontology language*, which is composed of a class $\mathcal{A}$ of (well-formed) *axioms*, and a *consequence relation* $\models\colon 2^{\mathcal{A}} \to \mathcal{A}$.[2] An *ontology* is a finite set $\mathcal{O} \subseteq \mathcal{A}$ of axioms. If $\mathcal{O} \models c$, where $c \in \mathcal{A}$, we say that $\mathcal{O}$ *entails* c, or that c is a *consequence* of $\mathcal{O}$. For reasons that will become clear later, we focus solely on *monotone* ontology languages, which are such that if $\mathcal{O} \subseteq \mathcal{O}'$ and $\mathcal{O} \models c$, then also $\mathcal{O}' \models c$.

Notice that these definitions follow the typical terminology from description logics as seen in the previous section, but they are not restricted exclusively to DLs. For example, the set of propositional clauses $\bigvee_{i=1}^{n} \ell_i$, where each ℓ_i, $1 \leq i \leq n$ is a literal, forms an ontology language under the standard entailment relation between formulas. In this case, an ontology is a formula in conjunctive normal form (CNF), and one common entailment of interest is whether such an ontology entails the empty clause $\bot := \bigvee_{\ell \in \emptyset} \ell$; that is, whether a formula is unsatisfiable. Clearly, this ontology language is monotone as well.

Historically, most of the work on ontology languages focuses on *reasoning*; that is, on studying and solving the problem of deciding whether $\mathcal{O} \models c$ holds; e.g. deciding subsumption or instance checking in $\mathcal{ALC}$. For the most prominent ontology languages, such as DLs, the computational complexity of this problem is perfectly understood, and efficient methods have been already developed and implemented. The number of available tools for solving these reasoning tasks is too large to enumerate them. However, knowing that a consequence follows from an ontology is only part of the story. Once that this fact has been established, we are left with the issue of answering *why* the consequence holds.

Why do we want to answer why? Well, mainly because the consequence relation is often far from trivial, specially when it depends on the inter-relation of several potentially complex axioms. Hence, it might not be obvious that a given axiom follows from an ontology. In particular, when the ontology is large, surprising or erroneous consequences are bound to appear. Explaining them is important to confirm their correctness or, alternatively, understand the causes of error when they are incorrect. Other applications that are based on this general setting are described in Section 7.

[2] We use the infix notation for the consequence relation

In *axiom pinpointing*, we explain consequences by computing so-called *justifications*.[3] Formally, a justification for the entailment $\mathcal{O} \models c$, where $\mathcal{O}$ is an ontology and c is an axioms, is a minimal (w.r.t. set inclusion) subontology $\mathcal{M} \subseteq \mathcal{O}$ that still entails the consequence; more precisely, $\mathcal{M} \models c$ and for every $\mathcal{M}' \subset \mathcal{M}$, $\mathcal{M}' \not\models c$. We emphasise that the minimality is considered here w.r.t. set inclusion. Notice that justifications are not unique. In fact, a single consequence relation $\mathcal{O} \models c$ may allow for exponentially many justifications, measured on the number of axioms in $\mathcal{O}$ [16]. Depending on the situation at hand, one may want to compute one, several, or all these justifications. Axiom pinpointing approaches can be broadly grouped into three categories: *black-box* methods that use unmodified reasoners as an oracle, *glass-box* methods that adapt the reasoning procedure to trace the axioms used, and *gray-box* approaches, which combine the benefits of the other two.

As mentioned already, the basic idea behind the glass-box approach is to modify the reasoning algorithm to keep track of the axioms used throughout the reasoning process, in order to identify the elements of the justification. As we will see later, this process is effective for computing all justifications, but incurs an additional cost, either in terms of complexity, or in the need of a post-processing step. Alternatively, if one is only interested in one justification, the cost of tracing is essentially insignificant, but the result is only an approximation of a justification: it may contain superfluous axioms. Thus, it is often combined with a black-box minimisation step that guarantees that the resulting set is indeed a justification. In the following sections we describe these approaches in greater detail.

4 Finding All Justifications

The black-box method for finding all justifications was introduced and studied in detail by the groups in Maryland and Manchester [37, 39, 54]. Very briefly, this method uses a sub-procedure that can compute one justification at a time (we will see how to achieve this in the following section) and systematically removes and restores axioms from the ontology, following Reiter's Hitting Set enumeration method, to find new justifications.

4.1 Tableaux-based Axiom Pinpointing

The history of glass-box methods for axiom pinpointing in DLs started more than 20 years ago, when Baader and Hollunder [5] extended the standard tableau-based algorithm for testing the consistency of an $\mathcal{ALC}$ ABox by a labelling technique that traced the application of the tableau rules, and ultimately the axioms responsible.

[3] I rather prefer the name *MinA* coined by Franz Baader as we started our work on this topic. However, despite my best efforts, *justification* has become the *de facto* standard name in DLs. Even I must admit that it is catchier.

In a nutshell, the standard tableaux-based algorithm for deciding consistency tries to build a forest-shaped model of the input ABox, where each node represents an individual of the domain, by decomposing the complex concepts to which each individual is required to belong to smaller pieces, until either an obvious contradiction is observed, or a model is obtained. For instance, the algorithm will decompose the assertion Tall $\sqcap$ LongHaired(franz) into the two simpler assertions Tall(franz) and LongHaired(franz). Similarly, existential restrictions are solved by introducing new individuals in a tree-like fashion; that is, for decomposing the assertion $\exists$hasStudent.$\neg$LongHaired(franz), the algorithm introduces a new (anonymous) individual x and the assertions hasStudent(franz, x) and $\neg$LongHaired(x). To keep all the decomposition steps positive, the algorithm transforms the concepts to negation normal form (NNF), where negations can only occur in front of concept names. Thus, the algorithm needs to handle explicit disjunctions as well. To decompose an assertion like Serious$\sqcup$Angry(franz) , where we do not know whether Franz is serious or angry, the ABox is duplicated, and one of the alternatives is added to each of the copies to analyse.

Since every ABox has a finite forest-shaped model, it can be shown that this process terminates after finitely many decomposition steps. Ultimately, the tableaux algorithm produces a set of ABoxes $\mathfrak{A}$ that represent the possible alternatives for bulding a model of the input ABox $\mathcal{A}$. An obvious contradiction (called a *clash*) is observed if the decomposed ABox contains two assertions $A(a), \neg A(a)$. It follows that $\mathcal{A}$ is consistent iff there is an ABox in $\mathfrak{A}$ without any clash; in fact, such clash-free ABox is a representation of a model of $\mathcal{A}$.

The tracing extension proposed in [5] labels each assertion obtained through the decomposition process with a value (formally a propositional variable) that represents the original axioms responsible for it. For example, suppose that the input ABox $\mathcal{A}$ contains the assertion $\exists$hasStudent.$\neg$LongHaired(franz). The algorithm first provides a unique name for this assertions, say $\mathsf{a_1}$. Then, when the decomposition process generates the two assertions hasStudent(franz, x) and $\neg$LongHaired(x), it marks both of them with the label $\mathsf{a_1}$ to express that they were caused by that original assertion. If an assertion is caused by more than one original axiom, its label is modified to be the disjunction of the variables associated with those axioms. A clash is caused by two contradictory assertions, each of which is labelled by a disjunction of variables. The conjunction of these labels describes the combinations of axioms required to produce this clash. Each ABox in $\mathfrak{A}$ may have more than one clash, but only one of them is required for inconsistency. Hence, one can define the *pinpointing formula*[4]

$$\varphi_{\mathcal{A}} := \bigwedge_{\mathcal{B}\in\mathfrak{A}} \bigvee_{A(a),\neg A(a)\in\mathcal{B}} \mathsf{lab}(A(a)) \wedge \mathsf{lab}(\neg A(a)).$$

This formula expresses all the combinations of axioms that lead to inconsistency, in the following way. If $\mathcal{V}$ is a valuation that satisfies $\phi_{\mathcal{A}}$, then the set of axioms

[4] In the original paper [5], this was called a *clash formula*, since it explains the clashes obtained by the algorithm. The name was later changed to pinpointing formula to reflect its more general puropose.

$\{\alpha \in \mathcal{A} \mid \mathsf{lab}(\alpha) \in \mathcal{V}\}$ is an inconsistent sub-ABox of $\mathcal{A}$. In particular, minimal valuations satisfying $\phi_{\mathcal{A}}$ define minimal inconsistent sub-ABoxes. Hence, the pinpointing formula can be seen as a (compact) representation of all the justifications for ABox inconsistency.

This original approach considered only ABoxes, but did not use any terminological knowledge in the form of GCIs. Later on, Schlobach and Cornet [64] extended the tracing method to explain inconsistency and concept unsatisfiability with respect to so-called unfoldable $\mathcal{ALC}$ terminologies, which allow for only a limited use of GCIs as concept definitions. This extension required adapting an additional tableau rule, necessary for handling the concept definitions, but followed the main steps from [5], based on the finite tree model property. This paper, which coined the term *axiom pinpointing*, started a series of extensions including additional constructors or different kinds of axioms [39, 49]. The correctness of these extensions was shown on an individual basis at each paper, despite all of them being based on the same principles: take the original tableau-based algorithm for the logic under consideration, and include a tracing mechanism—based on the execution of the tableau rules—that associates each newly derived assertion with the sets of axioms responsible for its occurrence. It was only when the notion of blocking was considered for handling arbitrary GCIs [41] that new ideas had to be developed to avoid stopping the process too early. As we will see, observing that the classical notion of blocking did not suffice for axiom pinpointing was the first hint of the problems that would later arise for this glass-box idea.

So, what do you do when you observe several instances of a process, and understand the principles behind it? You generalise them, of course! It was at that time that we started our attempts to develop a general notion of pinpointing extensions of tableau algorithms. This required two steps: giving a precise definition of what an abstract tableau algorithm (what we called a *general tableau* at the time) is, and describing how to implement the tracing mechanism on top of it, while guaranteeing correctness from an axiom pinpointing perspective. Fortunately, we could build on previous work by Franz for both steps. Indeed, a general notion of tableau had been defined a few years earlier to study the connections between tableau and automata reasoning methods [3]. Moreover, as mentioned before, the tracing mechanism and its correctness for axiom pinpointing was originally presented in [5]. Putting together both ingredients, after some necessary modifications, serious considerations on the notion and effect of blocking, and extension of the correctness proofs, led to the first general glass-box approach for axiom pinpointing in DLs and other logics [12].

An obvious drawback of these pinpointing extensions, which was almost immediately observed, is that the tracing mechanism requires the main optimisations of the tableau method to be disallowed. More precisely, to ensure efficiency, tableau algorithms stop exploring an ABox in the set of alternatives once a clash has been found. While this is correct for deciding a consequence (e.g., consistency), stopping at this point is bound to ignore some of the potential causes of the consequence, leading to an incomplete pinpointing method. In reality,

this inefficiency hid a larger problem that took some time to be understood and solved effectively.

After Baader and Hollunder proved the correctness of their approach in full detail, the following work took a rather abstract and simplified view on termination of the pinpointing extensions. In essence, most of the work starting from [64] argued that termination of the pinpointing method was a direct consequence of termination of the original tableau algorithm. This argument was convincing enough to make us believe that it should hold in general, but we needed a formal proof of this fact. After struggling for a long time, we ended up finding out that it is not true: there are terminating tableaux whose pinpointing extension does not terminate [12].[5] Fortunately for all the work coming before this counterexample was discovered, we were later able to identify a class of tableau methods where termination is guaranteed [15]. This class strictly contained all the previously studied algorithms in DL. Hence, for them we were still able to guarantee termination.

For the light-weight DL $\mathcal{EL}$, polynomial-time reasoning is achieved through a *completion* algorithm, which is an instance of what later became known as *consequence-based* methods [66,67]. As tableau methods, consequence-based approaches apply extension rules to try to prove an entailment from an ontology. The difference is that, while tableau algorithms attempt to construct a model of a special kind, consequence-based methods try to enumerate the consequences of the ontology that are relevant for the reasoning problem considered. Despite these differences, some consequence-based algorithms—including the completion method for $\mathcal{EL}$—can be seen as simple tableau approaches that are guaranteed to terminate. Thus, we can obtain a pinpointing formula for explaining consequences of $\mathcal{EL}$ ontologies [16,17]. This instance is a perfect example of the cost of adding the tracing mechanism to a tableau algorithm: while the original completion algorithm is guaranteed to terminate in polynomial time [10], its pinpointing extension may require exponentially many rule applications, and hence can only terminate in exponential time.

4.2 Automata-based Axiom Pinpointing

In addition to tableau-based (and consequence-based) algorithms, automata-based techniques are often used to reason in DLs. Automata-based algorithms are mostly considered in theoretical settings to prove complexity results, but to the best of our knowledge, only one experimental automata-based reasoner exists [23] (and stopped being developed long ago). The main reason for this is that, even though automata's worst-case behaviour is often better than for tableau-based algorithms, their *best-case* behaviour matches the worst case, and for practical ontologies tableau algorithms tend to be more efficient due to their goal-directed nature. However, as mentioned several times already, the tracing mechanism

[5] I still remember when I managed to construct the first counterexample just before the deadline for submitting the paper. Imagine a scared first-year PhD student interrupting his supervisor's holidays to tell him the bad news.

implemented for pinpointing reduces the efficiency of tableau methods. It is hence worth analysing the possibility of modifying automata-based algorithms to compute a pinpointing formula as well.

Automata-based reasoning methods for DLs exploit the fact that these logics often allow for well-structured models. For example, as we have seen before, every $\mathcal{ALC}$ satisfiable concept has a tree-shaped model. In this case, we can build an automaton that accepts the tree-shaped models of a given concept C w.r.t. an ontology. It then follows that C is unsatisfiable iff the language accepted by this automaton is empty. At a very high level, the automata-based algorithm for $\mathcal{ALC}$ tries to build a tree-shaped model by labelling the nodes of an infinite tree (which will form the domain of the interpretation) with a set of concepts to which they most belong. These sets of concepts, which form a type, correspond to the states of the automaton. The automaton should label the root node with a set containing the input concept C; the children of each node are then labelled in a way that satisfies the existential and value restrictions appearing in the parent node. Moreover, each type should be consistent with the constraints specified in the TBox. More precisely, if the TBox contains an axiom $C \sqsubseteq D$, then every type that contains C must also contain D. The automaton *accepts* the infinite tree iff such a labelling is possible. Importantly, deciding whether such a labelling exists is polynomial on the number of states of the automaton [29]. It is also important to notice that the automaton accepts *infinite* trees; hence, it does not require any special blocking technique to search for periodicity of a model.

To transform this reasoning algorithm into a pinpointing method, we modify the underlying automata model into a *weighted* automaton [33]. Very briefly, weighted automata generalise the classical notion of automata to provide an *initial weight* to each state (as a generalisation of the set of initial states), and a *weight* to each transition generalising the transition relation. Weighted automata require an algebraic structure called a *semiring* that has a *domain* S and two binary relations $\oplus$ (addition) and $\otimes$ (product), where $\otimes$ distributes over $\oplus$ and a few additional properties hold. Rather than dividing runs into successful and unsuccessful, weighted automata give a weight to every run, which is computed as the product of the weights of all the transitions used, and the weight of the initial state. The *behaviour* of the automaton is a function that maps every input tree T into a semiring value computed as the addition of the weights of all the runs this automaton over T.

For axiom pinpointing, we realise that the class of all propositional formulas over a finite alphabet of variables with the logical disjunction $\vee$ and conjunction $\wedge$ forms a distributive lattice, which is a specific kind of semiring. Thus, automata-based axiom pinpointing uses formulas as weights, and the two logical operators mentioned, to construct the pinpointing formula of a consequence. Recall that the states of the automaton used to decide concept satisfiability enforce that all the GCIs in the TBox are satisfied. Instead, we now allow these types to violate some of these constraints (e.g., even if the TBox contains the axiom $C \sqsubseteq D$, we allow a state that contains C but not D). The weights are used to keep track of exactly which axioms are being violated while labelling the input

tree. Thus, the behaviour of this automaton tells us which axioms need to be violated to obtain a model; i.e., a pinpointing formula for unsatisfiability. This was precisely the approach we followed in [13, 14].

At this point, we only need to find out how to compute the behaviour of a weighted automaton. Interestingly, despite extensive work made on different weighted automata models, by the time we were considering this issue there was no behaviour computation algorithm, and the computational complexity of this problem was not very well understood. So we had to develop our own techniques.[6] In the end, we developed a technique that extended the ideas behind the emptiness test for unweighted automata [4, 71] to track the weights; an alternative approach was independently developed around the same time [32]. Our approach, which only works on distributed lattices [42], runs in polynomial time on the size of the automaton, which matches the complexity of deciding emptiness of unweighted automata.

Thus, we showed that automata-based methods can also provide tight complexity bounds for axiom pinpointing, without worrying about issues like termination. There is, however, a small caveat. In order to guarantee a polynomial-time behaviour computation, the algorithm does not really compute the pinpointing formula, but a compact representation of it built through structure sharing. If the automaton is exponential on the size of the ontology, as is the case for $\mathcal{ALC}$, this representation could be expanded into a formula without any cost in terms of computational complexity. However, for less expressive logics such as $\mathcal{EL}$, this expansion may yield an exponential blow-up. Still, this blow-up may in fact be unavoidable in some of these logics. For example, it has been shown that there exist $\mathcal{EL}$ ontologies and consequences whose smallest pinpointing formula is of super-polynomial length [55]. In particular, this pinpointing formula cannot be generated in polynomial time.

For axiom pinpointing, the advantage of using automata-based methods over the tableau-based approach is that the problem with termination does not show up. Moreover, given that the computational resources needed to compute the behaviour are polynomially bounded on the size of the automaton, automata yield tight complexity bounds for this problem as well. However, it preserves the disadvantage observed for classical reasoning; namely, that its best-case complexity matches the worst-case one. Indeed, the first step of this approach corresponds to the construction of the automaton from the input ontology.

Overall, we have seen two methods for computing a pinpointing formula, and by extension all justifications, for a consequence from an ontology. In both cases, the methods present some issues that make them impractical. In fact, this is not very surprising given the now known complexity results for problems related to axiom pinpointing. As mentioned already, a single consequence may have exponentially many justifications, and hence it is impossible to enumerate them all

[6] As a historical remark, an important reason why I ended up working with Franz was because I fell in love with automata theory while I was doing my masters in Dresden. Being the Chair for Automata Theory, it only made sense to ask him for a topic. Little did I know at the time where this would take me.

in polynomial time. What is more interesting, though, is that even if a consequence has polynomially many justifications, there is no guarantee that they can be all found in polynomial time. In particular, any enumeration algorithm will necessarily lead to a longer waiting time between any two successive solutions, and even counting the number of justifications is a hard counting problem [57]. Given these hardness results, it makes sense to focus on the issue of computing only one justification. This is the topic of the next section.

5 Finding One Justification

The black-box approach for finding one justification relies on a straight-forward deletion procedure. The idea is to remove superfluous axioms from the ontology one at a time through a sequence of entailment tests. Starting from the full input ontology $\mathcal{O}$, the method iteratively removes one axiom at a time and checks whether the consequence c follows from the remaining ontology. If it does, the axiom is permanently removed; otherwise, the axiom is re-inserted in the ontology and the process continues. At the end of this process, when every axiom has been tested for deletion, the remaining ontology is guaranteed to be a justification for c [17,38]. Clearly, this process incurs in a linear overhead over reasoning: a standard reasoner is called as many times as there are axioms in the ontology. In terms of complexity, this means that as long as reasoning is at least polynomial, finding one justification is not noticeably harder than just deciding a consequence.

In practice, it has been observed empirically that justifications tend to be small, containing only a few axioms. If the original ontology is large, as in the case of SNOMED CT, the linear overhead may actually make the process unfeasible. Indeed, even if reasoning takes only a millisecond, repeating the process half a million times would require well over 5 minutes. Thus different optimisations have been proposed to try to prune several axioms at a time [37].

Trying to obtain a glass-box approach for computing one justification, one can adapt the tracing technique extending tableau-based algorithms to focus on *one* cause of derivation only. Recall that the idea behind the tracing technique is to keep track of the axioms used when deriving any new information throughout the execution of the tableau algorithm. To find only a justification, it thus makes sense to apply the same tracing technique, but ignore alternative derivations of the same fact. In other words, rather than preserving a (monotone) Boolean formula describing all the derivation of a given assertion, one just stores a conjunction of propositional variables obtained by the first derivation of this assertion. The method hence preserves one derivation of each assertion. When a clash is found, we can immediately find out the axioms known to cause this clash by conjoining the labels of the two assertions forming it. Likewise, to explain the presence of a clash in all the ABoxes generated by the execution of the algorithm, we simply conjoin the labels the clashes of each of them. Overall, this conjunction yields a set of axioms that is guaranteed to entail the consequence under consideration; e.g., inconsistency of the input ABox.

Interestingly, the problems surrounding the glass-box approach for computing *all* justifications do not play a role in this case. Indeed, the execution of the original tableau-based algorithm does not need to be modified, but only some additional memory is required to preserve the conjunction of labels at each generated assertion. In particular, the fundamental optimisations of the classical tableau methods are preserved: the expansion of an ABox may be stopped once a clash is found in it, without the need to find other possible clashes. Hence, this method runs within the same resource bounds as the original tableau method. Moreover, termination of the method is not affected by this tracing technique, and there is no need to adapt any blocking procedure used, since standard blocking remains correct for this setting. Unfortunately, as was almost immediately noted, the set of axioms generated by this modified algorithm is not necessarily a justification, since it might not be minimal [17]. In fact, it is not difficult to build an example where this approach produces a result with superfluous axioms, potentially caused by the order of rule applications or other dependencies between the assertions generated.

To obtain a justification, this glass-box approach can be combined with the black-box method described before. After finding an approximate justification through the glass-box approach, it can be minimised by deleting the superfluous axioms via the black-box method. Empirical evaluations of this *gray-box* approach show that it behaves well in practice, even for very large ontologies like SNOMED. Indeed, an intensive analysis started by Boontawee Suntisrivaraporn[7] [68] and concluded by Kazakov and Skocovskỳ [40] shows that consequences in this ontology tend to have very small justifications, mostly containing 10 axioms or less. Moreover, the average number of axioms that appear in at least one justification is less than 40 [56]. More interestingly, although the tracing algorithm has no guarantee of finding a justification, it very often does; and even when it does not, it usually gives only one or two superfluous axioms [68]. Thus, the minimisation step might not be needed in some applications.

To date, there is no automata-based method targeted to compute only one justification. Although it is possible to imagine a way to adapt the idea used in tableaux (i.e., rather than preserving formulas, weights are associated with conjunctions of propositional variables), this would require larger changes in terms of the chosen semiring and the use of the operations, and would not yield any real benefit in terms of complexity. Indeed, the automaton would be of the same size, and computing the behaviour requires the same resources, while still not guaranteeing minimality of the set of axioms computed. Hence, automata-based methods are not really meaningful in the context of computing only one justification, and we will not cover them further.

6 Extensions

The original question of axiom pinpointing, as described at the beginning of this chapter, deals exclusively with *whole* axioms. This is a reasonable approach

[7] Also known, and herewith referred as *Meng*.

when trying to debug errors in a hand-written ontology, as one can assume that ontology engineers follow modelling guidelines, which limit the variability in the expression of specific piece of knowledge. However, this also makes the results dependent on the representation chosen. For example, $\{\text{Tall} \sqcap \text{Professor}(\text{franz})\}$ is logically equivalent to $\{\text{Tall}(\text{franz}), \text{Professor}(\text{franz})\}$, but in axiom pinpointing both ontologies are treated differently: the first contains only one axiom, and is thus the only possible justification, while the second has a more fine-grained view where each of the individual axioms (or both together) may serve as a justification. We notice that the choice of the shape of the axioms is not banal. It may affect the underlying modelling language—and by extension the reasoning method chosen—among other things. Consider, for example, the difference between the $\mathcal{ALC}$ GCI $\top \sqsubseteq \neg\text{LongHaired} \sqcup \text{Professor}$ and the logically equivalent $\mathcal{EL}$ GCI $\text{LongHaired} \sqsubseteq \text{Professor}$. More importantly, it may also affect the complexity of axiom pinpointing itself. As shown in [57], allowing conjunctions as in the axiom $\text{Tall} \sqcap \text{Professor}(\text{franz})$ can increase the complexity of axiom pinpointing related tasks, like counting or enumerating justifications.

Depending on the application, the domain, the user, and the shape of the ontology, it may be desirable to produce a finer or a coarser view to a justification. For instance, if the goal is to *repair* an error in an ontology, it makes sense to try to view the ontology in as much detail as possible. Indeed, knowing that a long and (syntactically) complicated axiom is causing an error is less helpful for correcting it than observing more precise pieces, which highlight where the errors occur. As a simple example, knowing that the axiom $\text{Tall} \sqcap \text{Professor} \sqcap \neg\text{LongHaired}(\text{franz})$ causes an error is less informative than knowing that $\neg\text{LongHaired}(\text{franz})$ is causing it. On the other hand, if the goal is to *understand* why a consequence follows from an ontology, then a coarser view, where some of the irrelevant details are hidden from the user, may be more informative.

Variations of axiom pinpointing targeting finer or coarser justifications have been proposed throughout the years. Specifically for description logics, coarser justifications—called *lemmata*—were proposed in [35]. The idea in this case is to combine several axioms within a justification into one (simpler) axiom that follows from them and explains their relationship to the remaining ones in the justification. A simple example of this approach is a chain of atomic subsumptions $A_0 \sqsubseteq A_1, A_1 \sqsubseteq A_2, \ldots A_{n-1} \sqsubseteq A_n$ summarised into the lemma $A_0 \sqsubseteq A_n$. If a user is interested in observing the details of the lemma, then it can be expanded to its original form. Of course, the challenge is to summarise more complex combinations of axioms going beyond simple sequences of implications. To understand how this is done, it is worth looking at the original work in detail.

The approach for providing finer justifications originally took the form of so-called precise and laconic justifications [34]. In a nutshell, the idea of these justifications is to *cut* axioms into smaller, but still meaningful, pieces in a way that only the relevant pieces are presented to the user. In our previous example, instead of the (original) axiom $\text{Tall} \sqcap \text{Professor} \sqcap \neg\text{LongHaired}(\text{franz})$, the piece $\neg\text{LongHaired}(\text{franz})$ could be used as a laconic justification. Note that in

practice, the pieces of axioms derived for these laconic justifications are generalisations of the original axioms, with the additional property that they tend to be shorter and easier to read. One can think of taking this idea a step further, and trying to explain a consequence through the most general variants of the axioms possible. We will explore a closely related task in the next section.

Another important extension of the original approach to axiom pinpointing refers to the way axioms are related to each other. Recall that a justification is a minimal set of axioms (from the original ontology) that entails a given consequence. From this point of view, all axioms are independent in the sense that their presence or absence in the ontology does not depend on the existence of any other axiom. However, it is not difficult to find cases where this independence is not necessary. Without going far, we can consider the completion algorithm for $\mathcal{EL}$, which requires that the input ontology is written in a special normal form. Before calling the algorithm proper, the axioms in the ontology are transformed to this form; for example, an axiom $A \sqsubseteq B \sqcap C$ is replaced by the two axioms $A \sqsubseteq B$, $A \sqsubseteq C$. Note that this transformation does not affect the logical properties of the ontology, but as mentioned before, it may have an effect on axiom pinpointing. In this case, one should notice that, as the two latter axioms originate from the same input axiom, they are also bound to appear together. That is, whenever a justification from the normalised ontology contains $A \sqsubseteq B$, then it must also contain $A \sqsubseteq C$, and *vice-versa*. Another example is when the axioms are available to some users only, through an access control mechanism [7, 8]. In this case, all axioms at the same access level should be available simultaneously, along with those at more public levels.

One way to deal with this dependency between axioms is by means of *contexts*. From a very simplistic point of view, a context is merely a sub-ontology containing inter-related axioms. Technically, this inter-relation is expressed by a label associated with each axiom. In a nutshell, axioms that share the same label should always appear together. More complex relationships can then be expressed by a variation of the labelling language; e.g., by using propositional formulas, one can say that an axiom is available when another one is not.

In this context-based scenario, axiom pinpointing corresponds to finding the contexts from which a consequence can be derived, rather than just finding the specific axioms within those contexts. Since several axioms may belong to the same context, this provides a coarser explanation of the causes of the entailment. Interestingly, all the methods described in Sections 4 and 5 can be adapted to this variant of context-based reasoning, by using context labels—rather than the representation of an axiom—within the tracing process, and by including or removing whole contexts together in the black-box approach.

7 Applications

Now that we know what is axiom pinpointing, some of its variants, and how to solve these issues, we will see what they are useful for. Of course, we have already

hinted to various applications, which have motivated the previous descriptions, but now we try to cover them in larger detail.

The first obvious application is about correcting errors in an ontology. An early motivation for axiom pinpointing—definitely one that caught the interest of Franz—arose from working on the very large ontology SNOMED CT. At some point, it was observed that this ontology entailed the consequence AmputationOfFinger $\sqsubseteq$ AmputationOfHand; i.e., according to SNOMED, every amputation of a finger was also an amputation of a hand (and indeed, also of an arm, suggesting that there was something wrong with part-whole representations). This is a clearly erroneous conclusion that may have extreme consequences if the ontology is used to reason about real-world events.[8] So it was important to find the cause of the error, and correct it adequately. Using the gray-box approach for finding one justification, combined with the hitting-set tree method for finding successive ones, Franz and Meng managed to identify the *six* specific axioms that caused this error [18]. Perhaps more interestingly, they showed that there was a systematic error in the modelling approach used for the construction of the ontology, which was leading to these erroneous consequences. Hence, they proposed to use a slightly more expressive logic than $\mathcal{EL}$, in order to provide a more direct and intuitive mechanism for modelling relations between parts [69]. Since then, this error was erradicated from the ontology.

Staying in the context of correcting the errors in an ontology, recent efforts consider the approach that originated with laconic justifications, but rather than trying to find a justification over generalised axioms from the ontology, they generalise one further step to automatically remove the consequence. This line of research started from the idea of finding a consensus between mutually inconsistent ontologies by different agents [58]. It has then been continued by the research group in Bolzano [70] and, through a differnt motivation based on privacy, by Franz and his group [9].

Understanding and correcting the causes for an unwanted consequence to follow is a hard and time-consuming task that often requires the involvement of experts to decide which of the potentially exponentially many options to choose. In addition, updates to ontologies are often planned according to a stable calendar; for example, new versions of SNOMED are published twice per year. Hence, one should expect to wait some time before a known error is corrected in an ontology. In the meantime, one should still be able to use the ontology deriving meaningful consequences that avoid the potentially erroneous parts. This is the basic idea underlying *inconsistency tolerant* [19, 43], and more generally *error tolerant reasoning* [45]. Essentially, suppose that one knows that an ontology entails an erroneous consequence; the goal is to derive other consequences that would still hold in the absence of this error, and hence would still make sense after the ontology is repaired. To this end, three main semantics have been

[8] Imagine someone making an insurance claim after having a finger amputated. If the insurer makes this kind of error, they might end of paying a larger lump for an amputated arm.

defined, based on the notion of a *repair* [43]:[9] *brave* semantics consider consequences that follow from at least one repair; *cautious* semantics require that the conclusion is entailed by all repairs; and the *intersection* semantics, originally proposed for efficiency reasons, uses as a correct ontology the intersection of all the repairs [20]. In DLs, this aspect was originally motivated by the analogous notion in databases. From a similar motivation, currently the idea of tracing the provenance of a consequence in an ontology is gaining interest in the DL world. The difference with axiom pinpointing is that for provenance, the minimality assumption is relaxed; in fact, one is rather interested in finding all the possible ways in which a consequence can be derived [52].

Other important applications use axiom pinpointing as a background step for doing other complex inferences that depend on the combinations of axioms that entail some consequence. These applications are usually, although not always, based on the context-based generalisation described in the previous section. The first one, which we have already mentioned, is about access control. In this scenario, axioms have an access degree that limits the class of users that are able to retrieve them. These access degrees are extended also to the implicit consequences of the ontology in the obvious way: a user can observe a consequence iff they have access to a set of axioms that entail this consequence. Hence, for a given consequence, the problem is now to identify the access levels that can observe it, so that it remains hidden from all others. This becomes a problem of axiom pinpointing at the context level, where each access degree defines one context, and one is not interested in the specific axioms entailing the consequence, but rather their contexts. In [8] this problem is solved through a purely black-box approach using expressive DLs as underlying ontology language.

In order to handle uncertainty about the knowledge in an ontology, probabilistic extensions of DLs have been proposed [46]. A relevant example for this chapter are the DLs with so-called *distribution semantics* originally proposed for probabilistic logic programming [60, 62]. In this semantics, every axiom is associated with a probability of being true, and all axioms are assumed to be probabilistically independent. Then, the probability of a consequence is derived from the probabilities of all the combinations of axioms that entail the consequence. The most recent implementation of a reasoner for these logics uses the tableau-based glass-box method to compute a pinpointing formula, which is later fed to a propositional probabilistic reasoner [73]. While this approach seems to work well in empirical evaluations, the assumption of probabilistic independence is too strong to model realistic situations. For that reason, a more general formalism based on contexts was proposed. The idea of these newer probabilistic logics is to model certain knowledge that holds in uncertain contexts. That is, axioms are interrelated via context labels as described before, but the contexts are associated with a probability distribution, which in this case is expressed via a Bayesian network. Hence, these logics are often known as Bayesian logics. The

[9] Formally, a repair is a maximal subontology that does not entail the consequence. This is the dual notion of a justification, which is also studied in variations of axiom pinpointing in different fields.

first Bayesian DL was studied, as a variant of another probabilistic extension of DL-Lite [30], as an extension of $\mathcal{EL}$ [26], but it was immediately clear that the underlying ideas could be extended to other ontology languages as well [28]. The reasoning methods proposed for Bayesian $\mathcal{EL}$ included a black-box approach, and reductions to pure Bayesian networks [27] and to probabilistic logic programming with distribution semantics [25]. Later on, a glass-box approach based on a modification of the tracing algorithm for $\mathcal{ALC}$ was considered in [21, 22].

To conclude this section, and without going into excessive detail, we note that axiom pinpointing is also an effective sub-procedure for dealing with other extensions of logical reasoning. Examples of this are reasoning about preferences, possibilistic reasoning, and belief revision. More generally, whenever a reasoning problem can be expressed as a sum of products of weights from a semiring, where weights are associated with axioms, the weights of axioms in an ontology entailing a consequence are multiplied, and the results of different derivations are added, axiom pinpointing is a perfect companion to any reasoning method.

8 Conclusions

We have attempted to explain what is axiom pinpointing in the context of Description Logics, as studied by Franz Baader, and his academic successors. As mentioned throughout this work, the idea of axiom pinpointing is not restricted to the DL community, and pops out whenever people study some kind of monotonic entailment relation in detail. Unfortunately, each area chose a different name for this task, thus causing confusions and hindering communication about techniques, successes, and failures.

After finishing my dissertation, I have been trying to collect names and examples of axiom pinpointing *in the wild*, and keep on finding them in many different places. I guess it is true that when all you have is a hammer, then everthing looks like a nail. But this work is not about me, but about Franz, who not only proposed this topic to me as I started working with him, but was also the first to propose a solution to a special case, well before the name *axiom pinpointing* was coined by Schlobach and Cornet. Since Franz has a vested interest in DLs, and my work together with him has mainly focused on this area as well, this chapter does not go in much detail about other logical languages. Still, it would be wrong to leave anyone with the impression that the work presented here is exclusive for DLs. Many of the ideas are applicable to, and have often been independently developed for, other ontology languages as well. Famous examples are databases, propositional satisfiability, and constraint satisfaction problems; but there are many more. Even some that I have not yet encountered.

Perhaps more importantly, axiom pinpointing is not dead. Throughout the years, I have tried to leave aside the topic a couple of times, thinking that there cannot possibly exist much more to explore. And each time it came knocking back to my door, under different disguises. It is a fortune to us all that Franz continues exploring these topics as well. I am looking forward to the next years of axiom pinpointing-related research, in DLs and in other fields. And to the

results that Franz, and those of us that grew under him, will still be able to contribute.

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Kaelbling, L., Saffiotti, A. (eds.) Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05). pp. 364–369. Professional Book Center (2005)
2. 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)
3. Baader, F., Hladik, J., Lutz, C., Wolter, F.: From tableaux to automata for description logics. Fundamenta Informaticae **57**(2-4), 247–279 (2003), http://content.iospress.com/articles/fundamenta-informaticae/fi57-2-4-08
4. Baader, F., Hladik, J., Peñaloza, R.: Automata can show PSpace results for description logics. Information and Computation **206**(9-10), 1045–1056 (2008). https://doi.org/10.1016/j.ic.2008.03.006
5. Baader, F., Hollunder, B.: Embedding defaults into terminological knowledge representation formalisms. Journal of Automated Reasoning **14**(1), 149–180 (1995). https://doi.org/10.1007/BF00883932
6. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An Introduction to Description Logic. Cambridge University Press (2017)
7. 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: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) Proc. 8th Int. Semantic Web Conf. (ISWC'09). LNCS, vol. 5823, pp. 49–64. Springer (2009)
8. 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
9. Baader, F., Kriegel, F., Nuradiansyah, A., Peñaloza, R.: Making repairs in description logics more gentle. In: Thielscher, M., Toni, F., Wolter, F. (eds.) Proceedings of the Sixteenth 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
10. 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). Lecture Notes in Computer Science, vol. 4130, pp. 287–291. Springer (2006)
11. Baader, F., Lutz, C., Suntisrivaraporn, B.: Efficient reasoning in $\mathcal{EL}^+$. In: Parsia et al. [53], http://ceur-ws.org/Vol-189/submission_8.pdf
12. Baader, F., Peñaloza, R.: Axiom pinpointing in general tableaux. In: Olivetti, N. (ed.) Proceedings of the 16th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'07). Lecture Notes in Computer Science, vol. 4548, pp. 11–27. Springer (2007). https://doi.org/10.1007/978-3-540-73099-6_4
13. Baader, F., Peñaloza, R.: Automata-based axiom pinpointing. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR 2008). Lecture Notes in Computer Sci-

ence, vol. 5195, pp. 226–241. Springer (2008). https://doi.org/10.1007/978-3-540-71070-7_19
14. Baader, F., Peñaloza, R.: Automata-based axiom pinpointing. Journal of Automated Reasoning **45**(2), 91–129 (2010). https://doi.org/10.1007/s10817-010-9181-2
15. 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
16. Baader, F., Peñaloza, R., Suntisrivaraporn, B.: Pinpointing in the description logic $\mathcal{EL}$. In: Calvanese, D., Franconi, E., Haarslev, V., Lembo, D., Motik, B., Turhan, A.Y., Tessaris, S. (eds.) Proceedings of the 2007 International Workshop on Description Logics (DL'07). CEUR Workshop Proceedings, vol. 250. CEUR-WS.org (2007)
17. Baader, F., Peñaloza, R., Suntisrivaraporn, B.: Pinpointing in the description logic $\mathcal{EL}^+$. In: Hertzberg, J., Beetz, M., Englert, R. (eds.) Proceedings of the 30th Annual German Conference on AI (KI'07). Lecture Notes in Computer Science, vol. 4667, pp. 52–67. Springer (2007). https://doi.org/10.1007/978-3-540-74565-5_7
18. Baader, F., Suntisrivaraporn, B.: Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}^+$. In: Cornet, R., Spackman, K.A. (eds.) Proceedings of the Third International Conference on Knowledge Representation in Medicine. CEUR Workshop Proceedings, vol. 410. CEUR-WS.org (2008), http://ceur-ws.org/Vol-410/Paper01.pdf
19. Bertossi, L.E., Hunter, A., Schaub, T. (eds.): Inconsistency Tolerance, Lecture Notes in Computer Science, vol. 3300. Springer (2005). https://doi.org/10.1007/b104925
20. Bienvenu, M., Rosati, R.: Tractable approximations of consistent query answering for robust ontology-based data access. In: Rossi, F. (ed.) Proc. 23rd Int. Joint Conf. on Artificial Intelligence (IJCAI'13). pp. 775–781. AAAI Press/IJCAI (2013), http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6904
21. Botha, L., Meyer, T., Peñaloza, R.: The Bayesian description logic $\mathcal{BALC}$. In: Ortiz and Schneider [51], http://ceur-ws.org/Vol-2211/paper-09.pdf
22. Botha, L., Meyer, T., Peñaloza, R.: The Bayesian description logic $\mathcal{BALC}$. In: Proceedings of the 16th European Conference on Logics in Artificial Intelligence (JELIA'19). Lecture Notes in Computer Science, vol. 11468. Springer (2019)
23. Calvanese, D., Carbotta, D., Ortiz, M.: A practical automata-based technique for reasoning in expressive description logics. In: Walsh, T. (ed.) Proc. 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11). pp. 798–804. AAAI Press/IJCAI (2011). https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-140
24. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. of Automated Reasoning **39**(3), 385–429 (2007)
25. Ceylan, İ.İ., Mendez, J., Peñaloza, R.: The Bayesian ontology reasoner is BORN! In: Dumontier, M., Glimm, B., Gonçalves, R.S., Horridge, M., Jiménez-Ruiz, E., Matentzoglu, N., Parsia, B., Stamou, G.B., Stoilos, G. (eds.) Informal Proceedings of the 4th International Workshop on OWL Reasoner Evaluation (ORE-2015). CEUR Workshop Proceedings, vol. 1387, pp. 8–14. CEUR-WS.org (2015), http://ceur-ws.org/Vol-1387/paper_5.pdf
26. Ceylan, İ.İ., Peñaloza, R.: The Bayesian description logic $\mathcal{BEL}$. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) Proceedings of the 7th International Joint Conference on Automated Reasoning (IJCAR 2014). Lecture Notes in Computer Science, vol. 8562, pp. 480–494. Springer (2014). https://doi.org/10.1007/978-3-319-08587-6_37

27. Ceylan, İ.İ., Peñaloza, R.: Reasoning in the description logic $\mathcal{BEL}$ using Bayesian networks. In: Proceedings of the 2014 AAAI Workshop on Statistical Relational Artificial Intelligence. AAAI Workshops, vol. WS-14-13. AAAI (2014), http://www.aaai.org/ocs/index.php/WS/AAAIW14/paper/view/8765
28. Ceylan, İ.İ., Peñaloza, R.: The Bayesian ontology language $\mathcal{BEL}$. Journal of Automated Reasoning **58**(1), 67–95 (2017). https://doi.org/10.1007/s10817-016-9386-0
29. Comon, H., Dauchet, M., Gilleron, R., Löding, C., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata (2007), release October, 12th 2007
30. d'Amato, C., Fanizzi, N., Lukasiewicz, T.: Tractable reasoning with Bayesian description logics. In: Greco, S., Lukasiewicz, T. (eds.) Proceedings of the Second International Conference on Scalable Uncertainty Management (SUM 2008). Lecture Notes in Computer Science, vol. 5291, pp. 146–159. Springer (2008). https://doi.org/10.1007/978-3-540-87993-0_13
31. Donini, F.M., Massacci, F.: Exptime tableaux for $\mathcal{ALC}$. Artificial Intelligence **124**(1), 87–138 (2000). https://doi.org/10.1016/S0004-3702(00)00070-9
32. Droste, M., Kuich, W., Rahonis, G.: Multi-valued MSO logics overwords and trees. Fundamenta Informaticae **84**(3-4), 305–327 (2008), http://content.iospress.com/articles/fundamenta-informaticae/fi84-3-4-02
33. Droste, M., Kuich, W., Vogler, H.: Handbook of Weighted Automata. Springer, 1st edn. (2009)
34. 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)
35. 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
36. 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)
37. Kalyanpur, A.: Debugging and Repair of OWL Ontologies. Ph.D. thesis, University of Maryland College Park, USA (2006)
38. Kalyanpur, A., Parsia, B., Sirin, E., Grau, B.C.: Repairing unsatisfiable concepts in OWL ontologies. Lecture Notes in Computer Science, vol. 4011, pp. 170–184. Springer (2006). https://doi.org/10.1007/11762256_15
39. 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
40. 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
41. Lee, K., Meyer, T.A., Pan, J.Z., Booth, R.: Computing maximally satisfiable terminologies for the description logic $\mathcal{ALC}$ with cyclic definitions. In: Parsia et al. [53], http://ceur-ws.org/Vol-189/submission_29.pdf
42. Lehmann, K., Peñaloza, R.: The complexity of computing the behaviour of lattice automata on infinite trees. Theoretical Computer Science **534**, 53–68 (2014). https://doi.org/10.1016/j.tcs.2014.02.036

43. 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). Lecture Notes in Computer Science, vol. 6333, pp. 103–117. Springer (2010). https://doi.org/10.1007/978-3-642-15918-3_9
44. Liffiton, M.H., Sakallah, K.A.: Algorithms for computing minimal unsatisfiable subsets of constraints. Journal of Automated Reasoning **40**(1), 1–33 (2008). https://doi.org/10.1007/s10817-007-9084-z
45. Ludwig, M., Peñaloza, R.: Error-tolerant reasoning in the description logic $\mathcal{EL}$. Lecture Notes in Computer Science, vol. 8761, pp. 107–121. Springer (2014). https://doi.org/10.1007/978-3-319-11558-0_8
46. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the semantic web. Journal of Web Semantics **6**(4), 291–308 (2008)
47. 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
48. 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
49. Meyer, T.A., Lee, K., Booth, R., Pan, J.Z.: Finding maximally satisfiable terminologies for the description logic $\mathcal{ALC}$. 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
50. Motik, B., Patel-Schneider, P.F., Cuenca Grau, B. (eds.): OWL 2 Web Ontology Language: Direct Semantics. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-direct-semantics/
51. Ortiz, M., Schneider, T. (eds.): Proceedings of the 31st International Workshop on Description Logics (DL'18), CEUR Workshop Proceedings, vol. 2211. CEUR-WS.org (2018)
52. Ozaki, A., Peñaloza, R.: Provenance in ontology-based data access. In: Ortiz and Schneider [51], http://ceur-ws.org/Vol-2211/paper-28.pdf
53. 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)
54. Parsia, B., Sirin, E., Kalyanpur, A.: Debugging OWL ontologies. In: Ellis, A., Hagino, T. (eds.) Proceedings of the 14th international conference on World Wide Web (WWW 2005). pp. 633–640. ACM (2005). https://doi.org/10.1145/1060745.1060837
55. 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
56. 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). Lecture Notes in Computer Science, vol. 10249, pp. 518–533 (2017). https://doi.org/10.1007/978-3-319-58068-5_32
57. 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

58. Porello, D., Troquard, N., Confalonieri, R., Galliani, P., Kutz, O., Peñaloza, R.: Repairing socially aggregated ontologies using axiom weakening. In: An, B., Bazzan, A.L.C., Leite, J., Villata, S., van der Torre, L.W.N. (eds.) Proceedings of the 20th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA 2017). Lecture Notes in Computer Science, vol. 10621, pp. 441–449. Springer (2017). https://doi.org/10.1007/978-3-319-69131-2_26
59. Price, C., Spackman, K.: Snomed clinical terms. British Journal of Healthcare Computing and Information Management **17**(3), 27–31 (2000)
60. 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
61. 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
62. 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
63. Schild, K.: A correspondence theory for terminological logics: Preliminary report. In: Mylopoulos, J., Reiter, R. (eds.) Proc. 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91). pp. 466–471. Morgan Kaufmann (1991)
64. 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)
65. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. J. of Artificial Intelligence **48**, 1–26 (1991)
66. Simancik, F., Motik, B., Horrocks, I.: Consequence-based and fixed-parameter tractable reasoning in description logics. Artificial Intelligence **209**, 29–77 (2014)
67. 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
68. Suntisrivaraporn, B.: Polynomial time reasoning support for design and maintenance of large-scale biomedical ontologies. Ph.D. thesis, Technische Universität Dresden, Germany (2009), http://hsss.slub-dresden.de/deds-access/hsss.urlmapping.MappingServlet?id=1233830966436-5928
69. Suntisrivaraporn, B., Baader, F., Schulz, S., Spackman, K.A.: Replacing sep-triplets in SNOMED CT using tractable description logic operators. In: Bellazzi, R., Abu-Hanna, A., Hunter, J. (eds.) Proceedings of the 11th Conference on Artificial Intelligence in Medicine (AIME 2007). Lecture Notes in Computer Science, vol. 4594, pp. 287–291. Springer (2007). https://doi.org/10.1007/978-3-540-73599-1_38
70. 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
71. Vardi, M.Y., Wolper, P.: Automata-theoretic techniques for modal logics of programs. Journal of Computer and System Sciences **32**(2), 183–221 (1986). https://doi.org/10.1016/0022-0000(86)90026-7

72. Xiao, G., Calvanese, D., Kontchakov, R., Lembo, D., Poggi, A., Rosati, R., Zakharyaschev, M.: Ontology-based data access: A survey. pp. 5511–5519. ijcai.org (2018). https://doi.org/10.24963/ijcai.2018/777
73. Zese, R., Bellodi, E., Riguzzi, F., Cota, G., Lamma, E.: Tableau reasoning for description logics and its extension to probabilities. Annals of Mathematics and Artificial Intelligence **82**(1-3), 101–130 (2018). https://doi.org/10.1007/s10472-016-9529-3