Semiring Provenance in Expressive Description Logics

Rafael Peñaloza

University of Milano-Bicocca, Italy

Abstract

Provenance refers to the task of identifying and tracing the axiomatic origins of a consequence from a knowledge base. Semiring provenance uses two operators to represent (i) axioms that together yield the consequence and (ii) the different possible derivations. To-date, most provenance approaches are limited to inexpressive logics mainly due to the difficulty in dealing with complex constructors like negations. We surpass this issue through a general notion of provenance for interpretation-based semantics. We then show how weighted automata can be exploited to compute this provenance, hence providing an effective method for computing the provenance of $\mathcal{ALC}$ consequences.

Keywords

semiring provenance, ALC, weighted automata, non-standard reasoning

1. Introduction

The question of tracing the provenance of consequences has gained quite some interest over the last years, being translated to and used for dealing with many different knowledge representation languages. In its basic form, provenance refers to the task of tracing the "origin" or "sources" of a given consequence, among the pieces of knowledge available—that is, the axioms of a knowledge base—and, perhaps, rules for manipulating them.

Semiring provenance refers to a general framework that uses two operators—the "addition" and "product" of the semiring—to represent the full provenance of a consequence. In a nutshell, the product is used to express which axioms can combine to produce the consequence, while the addition represents different possible derivations of the same result. Although the name originates from the database community [1, 2], the same problem has been studied under different names in other areas, and has been shown to generalise non-standard semantics and reasoning problems [3, 4, 5, 6, 7, 8, 9].

Existing definitions of semiring provenance over different languages have a common feature: they rely (sometimes implicitly) on a notion of "proof." The idea is that the provenance information does not only indicate *which* pieces of knowledge are responsible for a consequence, but also *how* they interact for the derivation. The disadvantage of such a view is that it becomes difficult to apply in knowledge representation languages for which a natural definition of a proof is unavailable, or difficult to handle within the semiring operations. One example of

DL 2023: 36th International Workshop on Description Logics, September 2–4, 2023, Rhodes, Greece
rafael.penaloza@unimib.it (R. Peñaloza)
https://rpenalozan.github.io/ (R. Peñaloza)
0000-0002-2693-5790 (R. Peñaloza)

such a language is the description logic $\mathcal{ALC}$ [10], for which the consequence-based reasoning method [11] makes use of *negations*, which are not easy to manage in the context of provenance.

When an $\mathcal{ALC}$ KB entails a consequence, it is often important to understand which of the constraints (axioms) are violated. Here, provenance comes into play. And yet, it is unclear how to define the provenance in this situation due, as mentioned already, to the lack of a notion of proof which can be manipulated in this language. To alleviate this issue, we introduce a general notion of provenance which depends, instead, on an interpretation-based semantics. Intuitively, the provenance of a consequence is based on the combinations of axioms that exclude each possible interpretation not satisfying the consequence. Through this idea, we provide a sound notion of provenance which is suitable for $\mathcal{ALC}$, but also for any KR language with interpretation-based semantics.

The next question is whether this provenance can be effectively computed. We provide a positive answer to this question whenever the underlying KR language allows for an automata-based decision procedure, under minor assumptions. In brief, we check that a decision procedure based on an *axiomatic automaton*—which encodes the automata for all the sub-KBs of a given KB—can be modified into a weighted automaton (over a semiring) whose behaviour corresponds precisely to the provenance of the consequence. We also show how to compute this behaviour, for arbitrary weighted tree automata. In addition, we show that the behaviour computation requires only polynomial time in the number of states, although potentially exponential time on the number of transitions. Yet, for some specific cases (as when the semiring is a lattice) the latter exponential time upper bound can be reduced. Since there exist automata-based procedures for reasoning in $\mathcal{ALC}$ and other DLs [12, 13], our framework yields an effective procedure for computing the provenance in these languages. The resulting automaton has exponentially many states on the size of the KB. As a consequence, provenance is computable in exponential time for these cases, matching the complexity of reasoning.

It is important to consider that, although our motivation arises from trying to understand the consequences of a KB, the formalism presented in this paper is very general. Our definition of provenance is applicable to any knowledge representation language with interpretation-based semantics; and the automata-based provenance computation algorithm works for any consequence that can be decided through an axiomatic automaton. The main limitation is that our proofs of correctness require the underlying semiring to be distributive, idempotent, and commutative. Yet, as we argue at the end of the paper, relaxing any of these conditions is likely to have important negative consequences on the complexity of the problem.

2. Preliminaries

We assume basic knowledge of DLs [14] and in particular of $\mathcal{ALC}$. We provide the background knowledge on semirings and weighted automata needed for understanding the rest of this work.

2.1. Semirings

A *semiring* is an algebraic structure $\mathbb{S} = (S, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ where $\oplus$ and $\otimes$ are two associative binary operators over S, called the *addition* and *product*, respectively, such that $\oplus$ is commutative

and has neutral element $\mathbf{0}$; $\mathbf{1}$ is the neutral element of $\otimes$; and $\otimes$ distributes over $\oplus$ on both sides.[1] As usual, $\otimes$ has precedence over $\oplus$.

Such a semiring is *commutative* if $\otimes$ is commutative, $\oplus$-idempotent if $s \oplus s = s$ for all $s \in S$, and $\otimes$-idempotent if $s \otimes s = s$ for all $s \in S$. $\mathbb{S}$ is *idempotent* if it is both, $\oplus$- and $\otimes$-idempotent.

In the context of provenance, for reasons that will become clear later, one often considers commutative idempotent semirings. A typical example of such a semiring is the structure $\mathbb{L} = (L, \vee, \wedge, \mathbf{0}, \mathbf{1})$, where L is a bounded distributive lattice, $\vee$ and $\wedge$ are the *join* and *meet* operators of L, and $\mathbf{0}$ and $\mathbf{1}$ are its least and greatest elements, respectively. A semiring or lattice $\mathbb{S}$ is *distributive* if both operations distribute over each other. Distributive lattices have the additional property of being *annihilating*: for any two elements $s, t \in S$, it holds that $s \otimes (s \oplus t) = s = s \oplus (s \otimes t)$. Importantly, other distributive idempotent semirings exist, and other types of semiring may be of interest in different contexts [15, 16, 17].

In this paper, we follow a common approach from provenance and consider distributive idempotent and commutative semirings only. A useful observation is that if $(S, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ is a distributive idempotent commutative semiring, then so is its *dual* semiring $\overline{\mathbb{S}} := (S, \otimes, \oplus, \mathbf{1}, \mathbf{0})$. That is, exchanging the product and addition operations yields a new semiring. In the following sections, we will resort to this dual semiring for some constructions and computations.

From now on, unless otherwise specified, $\mathbb{S}$ refers to an arbitrary–but fixed–distributive idempotent commutative semiring over the set S, $\mathbb{S} = (S, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, and s to an element of S.

2.2. Weighted Tree Automata

Weighted tree automata generalise tree automata by allowing weights—from a given semiring—on their transitions [18, 19]. We consider only weighted automata on *infinite* trees [17, 20]. Given a number $k \in \mathbb{N}$, we denote as K the set $\{1, \dots, k\}$, and K^* is the set of all finite words over K. We see K^* as an infinite tree, where each word $w \in K^*$ is a node interpreted in the standard manner, with ε being the *root* and wi the i-th successor of w. For a set L, an *L-labelled tree* is a function $\mathsf{lab} : K^* \to L$ which assigns an element of L to each node of the tree. To make the reference to a tree explicit, we often name the function lab as T. For a node $u \in K^*$, $\overrightarrow{\mathsf{lab}(u)}$ denotes the tuple $(\mathsf{lab}(u), \mathsf{lab}(u1), \dots, \mathsf{lab}(uk))$.

Definition 1 (weighted automata)**.** *Let $\mathbb{S}$ be a semiring and $k \in \mathbb{N}$. A* weighted looping tree automaton *(WTA) over $\mathbb{S}$ for arity k is a tuple $\mathcal{A} = (Q, \mathsf{in}, \mathsf{wt})$ where Q is a finite set of* states*; $\mathsf{in} : Q \to S$ is the* initial weight *function; and $\mathsf{wt} : Q^{k+1} \to S$ is the* transition weight *function.*

A run *of the WTA $\mathcal{A}$ is a Q-labelled tree $R : K^* \to Q$. The class of all runs of $\mathcal{A}$ is denoted by $\mathsf{run}_{\mathcal{A}}$. The* weight *of the run R is $\mathsf{wt}(R) := \bigotimes_{u \in K^*} \mathsf{wt}(\overrightarrow{R(u)})$. The* behaviour *of the WTA $\mathcal{A}$ is*

$$\|\mathcal{A}\| := \bigoplus_{R \in \mathsf{run}_{\mathcal{A}}} \mathsf{in}(R(\varepsilon)) \otimes \mathsf{wt}(R).$$

Weighted tree automata generalise unweighted tree automata [21]. A *looping tree automaton* (LTA) is a tuple $\mathcal{A} = (Q, I, \Delta)$ where Q is the set of states, $I \subseteq Q$ is the set of *initial states*, and $\Delta \subseteq Q^{k+1}$ is the *transition relation*—recall that k is the tree arity. Consider now the Boolean

[1] Typically, but not always, $\mathbf{0}$ is required to annihilate over $\otimes$.

semiring $\mathbb{B} := (\{\mathbf{0}, \mathbf{1}\}, \vee, \wedge, \mathbf{0}, \mathbf{1})$, where $\vee$ and $\wedge$ are the standard Boolean disjunction and conjunction over the truth values $\mathbf{0}$ (false) and $\mathbf{1}$ (true). The LTA (Q, I, Δ) can be seen as the weighted automaton $(Q, \mathsf{in}, \mathsf{wt})$ over $\mathbb{B}$ where in and wt are the characteristic functions of I and Δ, respectively. Under this view, *accepted* runs are those with weight $\mathbf{1}$, and the automaton is *non-empty* (accepts at least one run) iff its behaviour is $\mathbf{1}$—and is *empty* if its behaviour is $\mathbf{0}$. Note that since we only consider distributive, idempotent, and commutative semirings, the behaviour of a WTA is well-defined.

Despite being a special case, for consistency and readability we use the notation (Q, I, Δ) when dealing with LTA and $(Q, \mathsf{in}, \mathsf{wt})$ when dealing with WTA. Both types of automata will be denoted with $\mathcal{A}$, allowing the context to clarify if it is a TA or a WTA.

3. Interpretation-based Provenance

The basic idea behind provenance is to keep track of the different combinations of axioms in a knowledge base which yield a consequence of interest. In logic-based KR languages, this is usually instantiated as a mirror of how (derived) consequences can be combined to yield new (previously implicit) consequences. The issue with this approach is that provenance semantics must be defined anew for each logical formalism, and it is not obvious how to define the provenance in settings where consequences are not cumulative in this sense. For instance, it is not obvious how to define provenance for $\mathcal{ALC}$ consequences in this manner. We introduce a general notion of provenance based on (tree-shaped) interpretation semantics.

Definition 2 (KR language)**.** *A* KR language *is a triple* $(\mathfrak{T}, \mathfrak{I}, \models)$ *where* $\mathfrak{T}$ *is a potentially infinite set of* axioms*,* $\mathfrak{I}$ *is a class of labelled trees called* interpretations*, and* $\models \subseteq \mathfrak{I} \times \mathfrak{T}$ *is the binary* satisfiability relation*. A* knowledge base *(KB) is a finite set of axioms* $\mathcal{T} \subseteq \mathfrak{T}$*. The interpretation* $I \in \mathfrak{I}$ *is a* model *of the KB* $\mathcal{T}$ *iff* $I \models \alpha$ *for all* $\alpha \in \mathcal{T}$*. An axiom* $\alpha \in \mathfrak{T}$ *is a* consequence *of a KB* $\mathcal{T}$ *(denoted* $\mathcal{T} \models \alpha$*) iff* $I \models \alpha$ *holds for every model* I *of* $\mathcal{T}$*.*

Under this definition, the consequence relation between KBs and axioms is *monotonic*: adding more axioms to a KB can never remove a consequence. DLs are examples of KR languages $(\Phi, \mathfrak{I}, \models)$, where Φ is the set of all GCIs, $\mathfrak{I}$ is the class of all DL interpretations, and $I \models C \sqsubseteq D$ iff $C^I \subseteq D^I$ as usual. In this case, a KB $\mathcal{T}$ (called often TBox in DL) is inconsistent iff $\mathcal{T} \models \top \sqsubseteq \bot$. We fix an arbitrary KR language $(\mathfrak{T}, \mathfrak{I}, \models)$, and assume that every axiom is annotated with an element of the semiring $\mathbb{S}$. The provenance of a consequence traces the combinations of axioms that yield it, as formalised next.

Definition 3 (provenance)**.** *Let* $\mathbb{S}$ *be a semiring and* $(\mathfrak{T}, \mathfrak{I}, \models)$ *a KR language. An* annotated KB *is a pair* $(\mathcal{T}, \mathsf{lab})$ *where* $\mathcal{T} \subseteq \mathfrak{T}$ *and* $\mathsf{lab} : \mathcal{T} \to S$*. The* provenance *of* $\alpha \in \mathfrak{T}$ *w.r.t. the annotated KB* $(\mathcal{T}, \mathsf{lab})$ *is* $\mathsf{Prov}_{\mathcal{T}}(\alpha) := \bigotimes_{I \in \mathfrak{I}, I \not\models \alpha} \bigoplus_{\beta \in \mathcal{T}, I \not\models \beta} \mathsf{lab}(\beta)$.

We analyse this definition in more detail. Traditionally, provenance is defined as an addition of products of labels referring to combinations of axioms that *yield* a consequence. We take a dual approach inspired by an approach from process analysis [22]. For α to be a consequence, any interpretation I that *does not* satisfy α must be excluded from the set of models; this is expressed by the outer product. The "addition" $\bigoplus_{\beta \in \mathcal{T}, I \not\models \beta} \mathsf{lab}(\beta)$ intuitively expresses that at least one axiom violated by I is needed to exclude I. This is clarified by the following example.

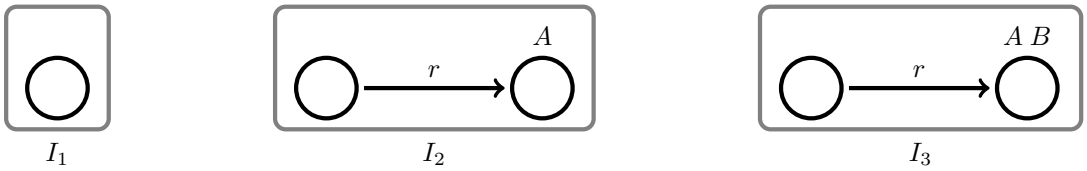

Figure 1: Representatives for interpretations violating axioms in $\mathcal{T}$ from Example 4.

Example 4. *Consider the annotated $\mathcal{ALC}$ KB*

$$\mathcal{T} := \{\top \sqsubseteq \exists r.A,\ \top \sqsubseteq \forall r.B,\ A \sqcap B \sqsubseteq \bot,\ A \sqsubseteq \neg A\},$$

with the axioms labelled as $s_1, \ldots, s_4$, respectively. $\mathcal{T}$ is inconsistent; that is, $\mathcal{T} \models \top \sqsubseteq \bot$. To compute the provenance for inconsistency, we have to multiply over all interpretations (since none satisfies $\top \sqsubseteq \bot$) the addition of the labels of the axioms they violate. Since we consider only idempotent semirings, it suffices to partition $\mathfrak{I}$ by the axioms they violate. Representatives of these classes are depicted in Figure 1. The interpretation I_1 violates s_1 only. I_2 violates s_2 and s_4 yielding $s_2 \oplus s_4$; and I_3 similarly yields $s_3 \oplus s_4$. $\mathsf{Prov}_{\mathcal{T}}(\top \sqsubseteq \bot)$ is the product of those additions: $s_1 \otimes (s_2 \oplus s_4) \otimes (s_3 \oplus s_4)$.

If the semiring $\mathbb{S}$ is annihilating, distributing this expression yields $(s_1 \otimes s_4) \oplus (s_1 \otimes s_2 \otimes s_3)$. The attentive reader will notice that the sets of axioms corresponding to each of these products (that is, $\{\top \sqsubseteq \exists r.A, A \sqsubseteq \neg A\}$, and $\{\top \sqsubseteq \exists r.A, \top \sqsubseteq \forall r.B, A \sqcap B \sqsubseteq \bot, \}$) characterise the minimal inconsistent sub-KBs; also known as justifications *in the DL literature [23, 24].*

As this example suggests, our definition of provenance traces the axioms that are responsible for a consequence. Consider the notion of a *justification* of a consequence w.r.t. a KB as a (subset-) minimal sub-KB that entails the consequence [24, 25]. If the underlying semiring is annihilating, the provenance of a consequence can be obtained by operating over the labels of the axioms appearing in the justifications [26]. Thus, as expected, our notion of provenance generalises that of justification-based explanations; also known as *axiom pinpointing* [6].

Theorem 5. *Let $\mathcal{T}$ be a KB and α a consequence, and $\mathsf{Just}_{\mathcal{T}}(\alpha)$ the class of all justifications of α w.r.t. $\mathcal{T}$. If the semiring $\mathbb{S}$ is annihilating, then $\mathsf{Prov}_{\mathcal{T}}(\alpha) = \bigoplus_{\mathcal{J} \in \mathsf{Just}_{\mathcal{T}}(\alpha)} \bigotimes_{\beta \in \mathcal{J}} \mathsf{lab}(\beta)$.*

From a different point of view, this definition also generalises reasoning problems based on Gödel (fuzzy) logic or $\min$-based possibilistic logic [27, 28]. Despite the simplicity of Example 4, it is in general not obvious how to compute the provenance of a consequence. In fact, a direct application of Definition 3 is made impossible from the fact that the class $\mathfrak{I}$ of interpretations is potentially infinite. In Example 4 we had to resort to a partition of $\mathfrak{I}$ into equivalence classes defined by the axioms each interpretation violates. In more general settings, or as the number of axioms grows, it is not obvious how to apply this approach. In Section 5 we see that for the special case of $\mathcal{ALC}$, creating and handling this partition is always possible.

Next we develop a method for computing the provenance of consequences for arbitrary KR languages having automata-based decision processes, with only some minor limitations.

4. Automata-based Provenance

Recall that it is possible, for a given $\mathcal{ALC}$ TBox $\mathcal{T}$, to construct an automaton whose runs characterise the models of $\mathcal{T}$. Consistency can thus be verified through an emptiness test of the constructed automaton: does it accept at least one successful run? The automaton is empty iff $\mathcal{T}$ is inconsistent. Similarly, our definition of provenance requires us to operate, akin to the behaviour of weighted automata, over all possible interpretations. This suggests that it should be possible to modify the automata-based decision procedure into the problem of computing the behaviour of an adequate weighted automaton.

Before formally describing the construction of the weighted automaton, two things should be taken into account. First, there is a duality between the emptiness of an automaton, and the consequence of interest. If the consequence (in the case of $\mathcal{ALC}$ *in*consistency) holds, then its automaton is *empty*; that is, its behaviour is $\mathbf{0}$—which is usually interpreted as *not* having a property. Dualising the Boolean semiring and using $\overline{\mathbb{B}}$ instead yields the "intuitive" answer of $\mathbf{1}$ when the consequence follows. Building on this intuition, the weighted automaton uses the dual structure $\overline{\mathbb{S}}$ instead of the original one $\mathbb{S}$.

Second, the automata-based decision procedure for a consequence cannot be based on an arbitrary construction; rather, it should grasp the effect of each axiom from the KB over the derivation of the consequence. Otherwise, it becomes impossible to trace the provenance of the consequence. To deal with this issue, we recall the notion of axiomatic automata from [29]. Briefly, we consider automata-based decision procedures which take a KB $\mathcal{T}$ and an axiom α, and construct a LTA $\mathcal{A}_{\mathcal{T},\alpha}$ such that $\mathcal{T} \models \alpha$ iff $\mathcal{A}_{\mathcal{T},\alpha}$ is empty. To grasp the influence of the axioms in the KB over the consequence, we consider only automata that "contain"—in a sense that will become clear briefly—all the automata for each sub-KB $\mathcal{T}' \subseteq \mathcal{T}$. The idea is that the automaton $\mathcal{A}_{\mathcal{T}',\alpha}$ for each $\mathcal{T}' \subseteq \mathcal{T}$ can be obtained by adding states and transitions to $\mathcal{A}_{\mathcal{T},\alpha}$. This is achieved through restricting functions.

Definition 6 (restricting functions)**.** *Let $\mathcal{A} = (Q, I, \Delta)$ be a LTA of arity k, and $\mathcal{T}$ a KB. A* transition restricting function *is a function $\Delta\mathsf{res} : \mathcal{T} \to \mathscr{P}(Q^{k+1})$; an* initial restriction function *is a function $I\mathsf{res} : \mathcal{T} \to \mathscr{P}(Q)$. They are extended to sets of axioms in $\mathcal{T}' \subseteq \mathcal{T}$ by defining*

$$\Delta\mathsf{res}(\mathcal{T}') := \bigcap_{\alpha\in\mathcal{T}'} \Delta\mathsf{res}(\alpha), \qquad I\mathsf{res}(\mathcal{T}') := \bigcap_{\alpha\in\mathcal{T}'} I\mathsf{res}(\alpha).$$

*Given $\mathcal{T}' \subseteq \mathcal{T}$, the $\mathcal{T}'$-*restricted subautomaton *of $\mathcal{A}$ w.r.t. $\Delta\mathsf{res}$ and $I\mathsf{res}$ is the LTA*

$$\mathcal{A}_{|\mathcal{T}'} := (Q, I \cap I\mathsf{res}(\mathcal{T}'), \Delta \cap \Delta\mathsf{res}(\mathcal{T}')).$$

The restricting functions express which of the initial states and transitions are allowed for a given axiom $\alpha \in \mathcal{T}$. Hence, if a whole set of axioms is considered, only the elements of $\mathcal{A}$ that are allowed by all of them (the intersection) can be included. By including more axioms in a sub-KB, we disallow more initial states and transitions, which makes it more likely to obtain an empty automaton. This is coherent with the monotonicity of consequences. This also simulates the notion of entailment from Definition 2. Each run of the automaton acts as a representative of a class of interpretations, and the restricting functions as a means to express which interpretations are satisfied by which axioms.

Definition 7 (axiomatic automata). *Let $\mathcal{A} = (Q, I, \Delta)$ be an LTA of arity k, $\mathcal{T}$ a KB, and $\Delta\mathsf{res} : \mathcal{T} \to \mathscr{P}(Q^{k+1})$, $I\mathsf{res} : \mathcal{T} \to \mathscr{P}(Q)$ restriction functions. The triple $(\mathcal{A}, \Delta\mathsf{res}, I\mathsf{res})$ is called an* axiomatic automaton *for $\mathcal{T}$. Let α be an axiom such that $\mathcal{T} \models \alpha$. The axiomatic automaton $(\mathcal{A}, \Delta\mathsf{res}, I\mathsf{res})$ is* correct *for $\mathcal{T} \models \alpha$ (w.r.t. the underlying KR language) iff for every $\mathcal{T}' \subseteq \mathcal{T}$ it holds that $\mathcal{T}' \models \alpha$ iff $\mathcal{A}_{|\mathcal{T}'}$ is empty.*

Under this definition, we can check which combinations of axioms entail a consequence α simply by checking emptiness of the restricted automata as long as the axiomatic automaton is correct for this consequence relation. The next step is to transform a correct axiomatic automaton for $\mathcal{T} \models \alpha$ into a process that computes the provenance of the consequence α w.r.t. the labels of the axioms in $\mathcal{T}$. The approach constructs a weighted automaton whose behaviour corresponds precisely to the desired provenance. The main idea is very natural: we combine the information from the axiomatic automaton with the notion of axioms that are violated in an interpretation to keep track of the provenance operations. In this sense, the weights of the automaton correspond to the annotations of the axioms that they violate; these are expressed through two violating functions, constructed over the restricting functions. Recall that we consider a fixed but arbitrary distributive semiring $\mathbb{S} = (S, \oplus, \otimes, \mathbf{0}, \mathbf{1})$. Moreover, as usual, we define $\bigoplus_{s \in \emptyset} s := \mathbf{0}$; i.e., an addition over an empty set is the neutral element $\mathbf{0}$.

Definition 8 (provenance automata). *Let $(\mathcal{A}, \Delta\mathsf{res}, I\mathsf{res})$ be an axiomatic automaton for $\mathcal{T} \models \alpha$ with $\mathcal{A} = (Q, I, \Delta)$ of arity k. The* violating functions $\Delta\mathsf{vio} : Q^{k+1} \to S$ *and* $I\mathsf{vio} : Q \to S$ *are:*

$$\Delta\mathsf{vio}(q_0, q_1, \ldots, q_k) := \bigoplus_{\{\beta \in \mathcal{T} \mid (q_0, \ldots, q_k) \notin \Delta\mathsf{res}(\beta)\}} \mathsf{lab}(\beta), \quad I\mathsf{vio}(q) := \bigoplus_{\{\beta \in \mathcal{T} \mid q \notin I\mathsf{res}(\beta)\}} \mathsf{lab}(\beta).$$

The provenance automaton *induced by $(\mathcal{A}, \Delta\mathsf{res}, I\mathsf{res})$ is the WTA over $\overline{\mathbb{S}} = (S, \otimes, \oplus, \mathbf{1}, \mathbf{0})$ $\mathcal{A}^{\mathsf{Prov}} := (Q, \mathsf{in}, \mathsf{wt})$ where:*

$$\mathsf{in}(q) := \begin{cases} I\mathsf{vio}(q) & q \in I, \\ \mathbf{1} & \textit{otherwise;} \end{cases} \quad \mathsf{wt}(q_0, \ldots, q_k) := \begin{cases} \Delta\mathsf{vio}(q_0, \ldots, q_k) & (q_0, \ldots, q_k) \in \Delta, \\ \mathbf{1} & \textit{otherwise.} \end{cases}$$

If a run R of $\mathcal{A}$ is accepted by $\mathcal{A}$ then its weight is $\mathsf{wt}(R) = \bigoplus_{u \in R} \Delta\mathsf{vio}(\overrightarrow{R(u)})$;[2] otherwise, $\mathsf{wt}(R) = \mathbf{1}$. From the definition of $\Delta\mathsf{vio}$, this expression resembles the internal addition in the definition of provenance (Definition 3). This is not coincidental. The behaviour of the automaton $\mathcal{A}^{\mathsf{Prov}}$, which applies $\otimes$ over the weights of all runs yields the provenance as expected.

Theorem 9. *If $(\mathcal{A}, \Delta\mathsf{res}, I\mathsf{res})$ is correct for $\mathcal{T} \models \alpha$, then $\|\mathcal{A}^{\mathsf{Prov}}\| = \mathsf{Prov}_{\mathcal{T}}(\alpha)$.*

This result tells us that the behaviour of the WTA $\mathcal{A}^{\mathsf{Prov}}$ yields the provenance of a consequence; yet, we still do not know how to compute this behaviour. To our knowledge, the only effective algorithms for computing the behaviour of weighted automata appear in [29, 30]. Abstracting from the acceptance conditions that they handle, the proofs of their correctness rely on specific properties of distributive *lattices* and are not directly applicable to our case where distributive semirings are considered. We provide an algorithm that builds on the ideas of the standard emptiness test approach for unweighted automata [31].

[2] Recall that we are using the dual semiring $\overline{\mathbb{S}}$, where the semiring "product" is $\oplus$.

Behaviour Computation

Let $\mathcal{A}$ be a WTA over the semiring $\mathbb{S}$. To compute the behaviour of $\mathcal{A}$, we need to compute the weight of all successful runs of $\mathcal{A}$. From the distributivity of $\mathbb{S}$, it follows that

$$\|\mathcal{A}\| = \bigoplus_{R \in \mathsf{run}_{\mathcal{A}}} \mathsf{in}(R(\varepsilon)) \otimes \mathsf{wt}(R) = \bigoplus_{q \in Q} \mathsf{in}(q) \otimes \bigoplus_{R \in \mathsf{run}_{\mathcal{A}}, R(\varepsilon)=q} \mathsf{wt}(R)$$

Based on this insight, we construct a function $\mathsf{swt} : Q \to S$ computing, for each state $q \in Q$, the addition of the weights of all runs starting with q; that is, $\mathsf{swt}(q) = \bigoplus_{R \in \mathsf{run}_{\mathcal{A}}, R(\varepsilon)=q} \mathsf{wt}(R)$. This function is constructed recursively, by considering runs of increasing depth. We start with the function swt_0 where $\mathsf{swt}_0(q) := \mathbf{0}$ if for all $q_1, \ldots, q_k \in Q$, $\mathsf{wt}(q, q_1, \ldots, q_k) = \mathbf{0}$ and $\mathsf{swt}_0(q) = \mathbf{1}$ otherwise. It identifies the weights of runs of depth 0; that is, those that only contain the root node, and allow for at least one transition. Then we iteratively construct for each $n \in \mathbb{N}$ the functions:

$$\mathsf{swt}_{n+1}(q) = \mathsf{swt}_n(q) \oplus \bigoplus_{q_1,\ldots,q_k \in Q} \left(\mathsf{wt}(q, q_1, \ldots, q_k) \otimes \bigotimes_{i=1}^{k} \mathsf{swt}_n(q_i) \right).$$

It is easy to verify, based on distributivity, that $\mathsf{swt}_n(q)$ yields the sum of all the runs of depth *at most* n starting with the state q. Let swt be the limit of the functions swt_n as n grows to infinity. Then we get the following result.

Theorem 10. *Let $\mathcal{A}$ be a WTA and* swt *the function defined above.* $\|\mathcal{A}\| = \bigoplus_{q \in Q} \mathsf{in}(q) \otimes \mathsf{swt}(q)$.

Let $S(\mathcal{A}) := \{s \in S \mid \exists q_0, \ldots, q_k \in Q . \mathsf{wt}(q_0, \ldots, q_k) = s\}$ be the set of all weights appearing in $\mathcal{A}$. All values appearing in the functions swt_n belong to the semiring $S_{\mathcal{A}}$ generated by $S(\mathcal{A})$; that is, they are constructed through arbitrary additions and products of elements of $S(\mathcal{A})$.[3] Due to distributivity, every such element can be rewritten in *monomial form*; that is, it can be expressed as an addition of monomials, where a monomial is a product of elements of $S(\mathcal{A})$. Idempotency allows us to restrict this even further, by requiring that each monomial appears at most once in the addition, and each element of $S(\mathcal{A})$ appears at most once in a monomial. Note that since $\mathbb{S}$ can be any arbitrary distributive idempotent semiring, and the weights of the transitions can be arbitrary elements of S, the monomial form of an element may not be unique. For our purposes, it suffices to know that there always exists one.

Define a partial ordering over the elements of $S_{\mathcal{A}}$ as follows. Let $s, t \in S_{\mathcal{A}}$; s is *contained* in t (denoted $s \subseteq t$) if there are monomial forms of s and t such that every monomial from s appears also in t. One can see that for every $n \in \mathbb{N}$ and every $q \in Q$ it follows that $\mathsf{swt}_n(q) \subseteq \mathsf{swt}_{n+1}(q)$; that is, the construction of swt is monotonically increasing and swt is its least fixpoint [32]. This fixpoint is bounded from above by the *maximal element* $\hat{s} := \bigoplus_{T \subseteq S(\mathcal{A})} \bigotimes_{s \in T} s$ constructed as the sum of all possible monomials over $S(\mathcal{A})$.

By the previous arguments, if there is an $n \in \mathbb{N}$ such that $\mathsf{swt}_n = \mathsf{swt}_{n+1}$, then $\mathsf{swt}_n = \mathsf{swt}$. To understand how expensive it is to compute swt is suffices to bound the number of iterations that yield different functions. Since $\mathsf{swt}_n(q) \subseteq \mathsf{swt}_{n+1}(q)$, if $\mathsf{swt}_n \neq \mathsf{swt}_{n+1}$ there exists some

[3] In particular, $\mathbf{1}$ is the monomial formed by empty products, and $\mathbf{0}$ is the object with an empty addition of monomials.

state $q \in Q$ and a monomial m appearing in $\mathsf{swt}_{n+1}(q)$ but not in $\mathsf{swt}_n(q)$. That is, every strictly increasing iteration must add at least one monomial to some state. The number of monomials is bounded by $2^{|S(\mathcal{A})|}$; thus after at most $|Q| \cdot 2^{|S(\mathcal{A})|}$ iterations the fixpoint is found. Assuming that semiring operations take constant time,[4] each iteration requires as many operations as there are transitions in $\mathcal{A}$; i.e., $|Q^{k+1}|$ of them for each state. Thus, the time for each iteration is bounded polynomially by the number of states of $\mathcal{A}$ yielding the following result.

Theorem 11. *The behaviour of a WTA $\mathcal{A} = (Q, \mathsf{in}, \mathsf{wt})$ over a distributive, commutative, idempotent semiring $\mathbb{S}$ can be computed in time polynomial in $|Q|$ but exponential on the cardinality of the range of* wt.

As usual, this upper bound is based on the worst-case scenario and can be reduced in special cases. For instance, if $\mathbb{S}$ is a distributive lattice, the annihilation property limits the class of monomials of interest to (perhaps surprisingly) a polynomial tree-depth, yielding a polynomial upper bound over $|Q|$ which does not depend explicitly on the range of wt [26, 29].

5. Provenance in $\mathcal{ALC}$

We now pivot to the special case of $\mathcal{ALC}$ and show how the automata-based approach can be used to trace the provenance of inconsistency of a TBox in this setting. The automata-based decision procedure for $\mathcal{ALC}$ TBox inconsistency tries to construct a special kind of tree-shaped model called a Hintikka tree [12]. In the following, we assume any concept to be (implicitly) rewritten into negation normal form (NNF).

Given a TBox $\mathcal{T}$, $\mathsf{sub}(\mathcal{T})$ denotes the set that contains $\neg C \sqcup D$ for each $C \sqsubseteq D \in \mathcal{T}$, and is closed under subconcepts. A set $H \subseteq \mathsf{sub}(\mathcal{T})$ is a *Hintikka set* iff (i) $C \sqcap D \in H$ implies $\{C, D\} \subseteq H$; (ii) $C \sqcup D \in H$ implies $\{C, D\} \cap H \neq \emptyset$; and (iii) for every $A \in N_C$ $\{A, \neg A\} \not\subseteq H$. In words, a Hintikka set is a logically consistent set of concepts, which preserves the semantics of $\sqcap$ and $\sqcup$. Such a set is *compatible* with the GCI $C \sqsubseteq D$ iff either $H = \emptyset$ or $\neg C \sqcup D \in H$.

Hintikka sets label the nodes of the tree, and existential restrictions are satisfied by their successors. Let k be the number of existential restrictions in $\mathsf{sub}(\mathcal{T})$. To know which successor refers to which existential restriction we fix an arbitrary bijection $\varphi : \{\exists r.C \mid \exists r.C \in \mathsf{sub}(\mathcal{T})\} \to K$. Some successors will be labeled with the dummy Hintikka set $\emptyset$ to keep the constant arity k. The tuple of Hintikka sets $(H_0, H_1, \ldots, H_k)$ satisfies the *Hintikka condition* iff for every $\exists r.C \in \mathsf{sub}(\mathcal{T})$, (i) if $\exists r.C \in H_0$, then $\{C\} \cup \{D \mid \forall r.D \in H_0\} \subseteq H_{\varphi(\exists r.C)}$; and (ii) if $\exists r.C \notin H_0$, then $H_{\varphi(\exists r.C)} = \emptyset$. It is *compatible* with the GCI α iff $H_0, \ldots, H_k$ are all compatible with α.

Definition 12 ($\mathcal{ALC}$ automaton)**.** *Let $\mathcal{T}$ be a TBox and k the number of existential restrictions in* $\mathsf{sub}(\mathcal{T})$*. The axiomatic automaton for $\mathcal{T}$ is the triple $(\mathcal{A}_\mathcal{T}, \Delta\mathsf{res}_\mathcal{T}, I\mathsf{res}_\mathcal{T})$ where $\mathcal{A} = (Q, \Delta, Q)$ with Q the set of all Hintikka sets and Δ the set of tuples satisfying the Hintikka condition; $\Delta\mathsf{res}_\mathcal{T}$ maps each $\alpha \in \mathcal{T}$ to the set of tuples compatible with α; and $I\mathsf{res}_\mathcal{T}(\alpha) = Q$ for all $\alpha \in \mathcal{T}$.*

[4] Depending on the structure of $\mathbb{S}$, this assumption may require a more precise specification. For instance [29] introduce structure sharing for efficient calculations.

For the case of $\mathcal{ALC}$, the fundamental element is $\Delta\mathsf{res}_{\mathcal{T}}$, which guarantees that all nodes in the tree satisfy the GCIs of the sub-TBox of interest. Or, from a dual view, can be used to see which axioms are violated in a tree-shaped interpretation.

Theorem 13. *The axiomatic automaton* $(\mathcal{A}_{\mathcal{T}}, \Delta\mathsf{res}_{\mathcal{T}}, I\mathsf{res}_{\mathcal{T}})$ *is correct for* $\mathcal{T}$ *w.r.t. inconsistency.*

By Theorem 9, the behaviour of $\mathcal{A}_{\mathcal{T}}^{\mathsf{Prov}}$ yields the provenance for inconsistency of the TBox $\mathcal{T}$. Thus this provenance can be effectively computed.

As usual for $\mathcal{ALC}$ and other DLs, the approach for dealing with inconsistency can be applied, with minor variations, to other reasoning tasks. For instance, if one is interested in verifying whether a concept C is *unsatisfiable* w.r.t. $\mathcal{T}$—that is, whether every model I of $\mathcal{T}$ is such that $C^I = \emptyset$—it suffices to modify the set of initial states of the automaton $\mathcal{A}$ (Definition 12) to contain only those Hintikka sets that include the concept C. Then, the provenance for the entailment of $C \sqsubseteq D$ w.r.t. $\mathcal{T}$ becomes the provenance for unsatisfiability of $C \sqcap \neg D$.

6. Variants and Extensions

Consider once again Example 4. Under the assumption of annihilation, we first computed the provenance to be equivalent to $s_1 \otimes (s_2 \oplus s_4) \otimes (s_3 \oplus s_4)$. It turns out that each of these additions characterises a so-called *diagnosis*: a minimal set of axioms that, when removed from the KB, cancel the consequence [33]. For instance, we notice that if we remove the first axiom, the resulting KB $\{\top \sqsubseteq \forall r.B, A \sqcap B \sqsubseteq \bot, A \sqsubseteq \neg A\}$ is consistent. This is consistent with the well-known duality between justifications and repairs, observed in different domains [34, 35, 36, 37]. Thus, if rather than *explaining* a consequence through its provenance one was interested in *removing* it, one could use a similar technique to compute this "correction provenance." The only necessary change would be to consider the original semiring $\mathbb{S}$ as underlying the weights of the provenance automaton, rather than the dual $\overline{\mathbb{S}}$ as done for provenance.

One assumption that we make in the definition of general KR languages (see Definition 2) is that any combination of axioms can form a KB—indeed, a KB is defined as a set of axioms. In some cases, it makes sense to restrict the class of KBs to so-called *admissible* sets of axioms, where if a set $\mathcal{T}$ is admissible, then any subset of $\mathcal{T}$ is also admissible [38]. For instance in expressive description logics, some axioms can only be used if the symbols that appear in them do not appear in other axioms [39]. Importantly, even if we restrict the KBs of KR languages to such admissible sets of axioms, the definition of an axiomatic automaton and the construction of the provenance automaton as presented above still work. Hence, our approach is general enough to handle also the settings discussed in [38].

7. Conclusions

We have considered a semantic notion of provenance, which can be readily applied to arbitrary knowledge representation languages that allow for an interpretation-based consequence relation. In contrast to other existing notions of provenance [40, 41, 42], our notion does not depend on an executional consequence relationship between the axioms in a knowledge base, but rather on how they combine to exclude interpretations negating the consequence of interest.

The advantage of our definition is that it allows us to handle provenance in languages where consequence-based reasoning methods are not available, or require awkward constructions.

At first sight, our definition of provenance may not be very intuitive. It is based on finding the combinations of axioms that guarantee consequences to be entailed, taking into consideration the interpretation-based semantics. As a means to justify our approach, we show that it naturally generalises one of the best known special cases of provenance; namely, that of finding all subset-minimal sub-KBs that entail the consequence. Moreover, it generalises the cases of weighted reasoning where weights come from a distributive lattice [43]. It is worth mentioning that, while we use axiom pinpointing as a motivation and an example for the formalism, the approach is applicable to provenance over any distributive, idempotent, and commutative semiring.

From a practical point of view, we showed how to transform any axiomatic automaton which is correct for deciding a consequence, into a weighted automaton whose behaviour corresponds precisely to the provenance of the consequence. We also provide an effective algorithm for computing this behaviour by recursively considering runs of increasing length. Computing the provenance is not more expensive than standard reasoning (based on automata emptiness), except for a potential exponential overhead on the number of different transition weights.

To showcase the importance and use of our definitions, we instantiated the framework on $\mathcal{ALC}$. Although there exist consequence propagation methods for this logic [11], its "natural" reasoning method is based on the construction of (tree) models. Applying our construction, we showed that computing the provenance in this logic requires exponential time, which matches the complexity of deciding inconsistency.

One may consider that limiting the framework to distributive, idempotent, commutative semirings is a very strong restriction. Yet, these assumptions are justified in the context of our framework. Idempotency and commutativity are relatively common requirements in the context of provenance for expressive languages; see e.g., [41]. Indeed, there is some evidence that a lack of idempotency yields to a high complexity, and perhaps even undecidability of provenance-related problems. If the operators are not commutative, then one can construct increasingly long monomials, in detriment to the resource upper bounds derived. Distributivity, on the other hand, is useful for the automata behaviour computation. Indeed, it is known that even for the case of lattice-valued WTA, the polynomial-time upper bounds depend strongly on distributivity [44]. Moreover, the construction of the provenance automaton depends on the dualisation of the semiring, which is only guaranteed for the class of algebras considered here. This last issue can be solved by considering diagnose-based provenance, instead of explanations. On the other hand, some applications of provenance do consider less restrictive scenarios looking at e.g., the Viterbi semiring whose product is not idempotent or the tropical semiring with non-idempotent addition.

Some automata-based reasoning methods are not based on the interpretation semantics, but follow a more consequence-based approach of building "proofs" for the derivation of a consequence. Examples of this view are propositional resolution [45] and the completion-based approach for $\mathcal{EL}$ [46]. In future work we will study the connection between these automata-based approaches, and verify whether provenance can be computed efficiently over them.

Acknowledgments

This work was partially supported by the MUR for the Department of Excellence DISCo at the University of Milano-Bicocca and under the PRIN project PINPOINT Prot. 2020FNEB27, CUP H45E21000210001.

References

[1] T. J. Green, G. Karvounarakis, V. Tannen, Provenance semirings, in: Proc. of PODS 2007, ACM, 2007, pp. 31–40. doi:10.1145/1265530.1265535.

[2] T. J. Green, V. Tannen, The semiring framework for database provenance, in: Proc. of PODS 2017, ACM, 2017, pp. 93–99. doi:10.1145/3034786.3056125.

[3] L. F. Sikos, D. Philp, Provenance-aware knowledge representation: A survey of data models and contextualized knowledge graphs, Data Science and Engineering 5 (2020) 293–316. doi:10.1007/s41019-020-00118-0.

[4] R. Dividino, S. Sizov, S. Staab, B. Schueler, Querying for provenance, trust, uncertainty and other meta knowledge in RDF, Journal of Web Semantics 7 (2009) 204–219.

[5] F. Baader, B. Hollunder, Embedding defaults into terminological knowledge representation formalisms, J. Automated Reas. 14 (1995) 149–180. doi:10.1007/BF00883932.

[6] S. Schlobach, R. Cornet, Non-standard reasoning services for the debugging of description logic terminologies, in: Proc. of IJCAI 2003, Morgan Kaufmann, 2003, pp. 355–362. URL: http://ijcai.org/Proceedings/03/Papers/053.pdf.

[7] A. Kalyanpur, B. Parsia, M. Horridge, E. Sirin, Finding all justifications of OWL DL entailments, in: Proc. of ISWC 2007, volume 4825 of *Lecture Notes in Computer Science*, Springer, 2007, pp. 267–280. doi:10.1007/978-3-540-76298-0_20.

[8] H. Kleine Büning, O. Kullmann, Minimal unsatisfiability and autarkies, in: Handbook of Satisfiability - Second Edition, volume 336 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2021, pp. 571–633. doi:10.3233/FAIA200997.

[9] O. Kullmann, I. Lynce, J. Marques-Silva, Categorisation of clauses in conjunctive normal forms: Minimally unsatisfiable sub-clause-sets and the lean kernel, in: Proc. of SAT 2006, volume 4121 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 22–35. doi:10.1007/11814948_4.

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

[11] F. Simancik, Y. Kazakov, I. Horrocks, Consequence-based reasoning beyond Horn ontologies, in: T. Walsh (Ed.), Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI/AAAI, 2011, pp. 1093–1098. doi:10.5591/978-1-57735-516-8/IJCAI11-187.

[12] F. Baader, J. Hladik, R. Peñaloza, Automata can show PSPACE results for description logics, Information and Computation 206 (2008) 1045–1056. doi:10.1016/j.ic.2008.03.006.

[13] D. Calvanese, D. Carbotta, M. Ortiz, A practical automata-based technique for reasoning in expressive description logics, 2011, pp. 798–804. doi:10.5591/978-1-57735-516-8/IJCAI11-140.

[14] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation, and Applications, second ed., Cambridge University Press, 2007.
[15] A. Kaya, M. Satyanarayana, Semirings satisfying properties of distributive type, Proceedings of the American Mathematical Society 82 (1981) 341–346.
[16] F. Pastijn, Y. Guo, The lattice of idempotent distributive semiring varieties, Science in China Series A: Mathematics 42 (1999) 785–804. doi:10.1007/BF02884266.
[17] Z. Fülöp, H. Vogler, Weighted tree automata – may it be a little more?, 2022. doi:10.48550/ARXIV.2212.05529.
[18] H. Seidl, Finite tree automata with cost functions, in: J. C. Raoult (Ed.), CAAP '92, Springer, Berlin, Heidelberg, 1992, pp. 279–299.
[19] M. Droste, W. Kuich, H. Vogler, Handbook of Weighted Automata, EATCS, 1st ed., Springer, 2009.
[20] M. Droste, G. Rahonis, Weighted automata and weighted logics on infinite words, in: Proc. of DLT 2006, volume 4036 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 49–58. doi:10.1007/11779148_6.
[21] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, C. Löding, S. Tison, M. Tommasi, Tree automata techniques and applications, 2008.
[22] D. Deutch, Y. Moskovitch, V. Tannen, A provenance framework for data-dependent process analysis, Proc. VLDB Endow. 7 (2014) 457–468. doi:10.14778/2732279.2732283.
[23] R. Peñaloza, Explaining axiom pinpointing, in: C. Lutz, U. Sattler, C. Tinelli, A. Turhan, F. Wolter (Eds.), Description Logic, Theory Combination, and All That - Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday, volume 11560 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 475–496. doi:10.1007/978-3-030-22102-7_22.
[24] M. Horridge, J. Bauer, B. Parsia, U. Sattler, Understanding entailments in OWL, in: Proceedings of the Fifth OWLED Workshop, volume 432 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2008. URL: http://ceur-ws.org/Vol-432/owled2008eu_submission_23.pdf.
[25] R. Peñaloza, Axiom pinpointing, in: Applications and Practices in Ontology Design, Extraction, and Reasoning, volume 49 of *Studies on the Semantic Web*, IOS Press, 2020, pp. 162–177. URL: https://ebooks.iospress.nl/volumearticle/56013. doi:10.3233/SSW200042.
[26] F. Baader, M. Knechtel, R. Peñaloza, Context-dependent views to axioms and consequences of semantic web ontologies, J. Web Sem. 12–13 (2012) 22–40. doi:10.1016/j.websem.2011.11.006.
[27] N. Preining, Gödel logics – a survey, in: Logic for Programming, Artificial Intelligence, and Reasoning, Springer, Berlin, Heidelberg, 2010, pp. 30–51.
[28] D. Dubois, H. Prade, Possibilistic logic - an overview, in: Computational Logic, volume 9 of *Handbook of the History of Logic*, Elsevier, 2014, pp. 283–342. doi:10.1016/B978-0-444-51624-4.50007-1.
[29] F. Baader, R. Peñaloza, Automata-based axiom pinpointing, J. Automated Reas. 45 (2010) 91–129. doi:10.1007/s10817-010-9181-2.
[30] M. Droste, W. Kuich, G. Rahonis, Multi-valued MSO logics over words and trees, Fundamenta Informaticae 84 (2008) 305–327.
[31] M. Y. Vardi, P. Wolper, Automata-theoretic techniques for modal logics of programs, Journal of Computer and System Sciences 32 (1986) 183–221. doi:10.1016/0022-0000(86)

90026-7.
[32] A. Tarski, A lattice-theoretical fixpoint theorem and its applications., Pacific Journal of Mathematics 4 (1955) 285–309.
[33] R. Reiter, A theory of diagnosis from first principles, Artificial Intelligence 32 (1987) 57–95. URL: https://www.sciencedirect.com/science/article/pii/0004370287900622. doi:https://doi.org/10.1016/0004-3702(87)90062-2.
[34] E. Birnbaum, E. L. Lozinskii, Consistent subsets of inconsistent systems: structure and behaviour, Journal of Experimental & Theoretical Artificial Intelligence 15 (2003) 25–46. doi:10.1080/0952813021000026795.
[35] M. H. Liffiton, K. A. Sakallah, Algorithms for computing minimal unsatisfiable subsets of constraints, Journal of Automated Reasoning 40 (2008) 1–33. doi:10.1007/s10817-007-9084-z.
[36] A. Ignatiev, N. Narodytska, N. Asher, J. Marques-Silva, On relating 'why?' and 'why not?' explanations, CoRR abs/2012.11067 (2020). URL: https://arxiv.org/abs/2012.11067. arXiv:2012.11067.
[37] F. Baader, R. Peñaloza, B. Suntisrivaraporn, Pinpointing in the description logic EL+, in: Proc. of KI 2007, volume 4667 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Osnabrück, Germany, 2007, pp. 52–67.
[38] F. Baader, R. Peñaloza, Axiom pinpointing in general tableaux, Journal of Logic and Computation 20 (2010) 5–34. doi:10.1093/logcom/exn058.
[39] I. Horrocks, O. Kutz, U. Sattler, The even more irresistible SROIQ, in: Proc. of KR 2006, AAAI Press, 2006, pp. 57–67. URL: http://www.aaai.org/Library/KR/2006/kr06-009.php.
[40] K. M. Dannert, E. Grädel, Provenance Analysis: A Perspective for Description Logics?, Springer International Publishing, Cham, 2019, pp. 266–285. doi:10.1007/978-3-030-22102-7_12.
[41] C. Bourgaux, A. Ozaki, R. Peñaloza, L. Predoiu, Provenance for the description logic ELHr, ijcai.org, 2020, pp. 1862–1869. doi:10.24963/ijcai.2020/258.
[42] M. Knorr, C. V. Damásio, R. Gonçalves, J. Leite, Towards provenance in heterogeneous knowledge bases, in: Logic Programming and Nonmonotonic Reasoning, Springer International Publishing, Cham, 2022, pp. 287–300.
[43] S. Borgwardt, R. Peñaloza, Consistency reasoning in lattice-based fuzzy description logics, International Journal of Approximate Reasoning 55 (2014) 1917–1938. doi:10.1016/j.ijar.2013.07.006.
[44] K. Lehmann, R. Peñaloza, The complexity of computing the behaviour of lattice automata on infinite trees, Theoretical Computer Science 534 (2014) 53–68. URL: http://www.sciencedirect.com/science/article/pii/S0304397514001625. doi:10.1016/j.tcs.2014.02.036.
[45] M. Davis, H. Putnam, A computing procedure for quantification theory, J. ACM 7 (1960) 201–215. doi:10.1145/321033.321034.
[46] F. Baader, S. Brandt, C. Lutz, Pushing the EL envelope, in: Proc. of IJCAI 2005, Professional Book Center, 2005, pp. 364–369. URL: http://ijcai.org/Proceedings/05/Papers/0372.pdf.