Using Sums-of-Products for Non-standard Reasoning

Rafael Peñaloza

Theoretical Computer Science
TU Dresden, Germany
penaloza@tcs.inf.tu-dresden.de

Abstract. An important portion of the current research in Description Logics is devoted to the expansion of the reasoning services and the developement of algorithms that can adequatedly perform so-called non-standard reasoning. Applications of non-standard reasoning services cover a wide selection of areas such as access control, agent negotiation, or uncertainty reasoning, to name just a few. In this paper we show that some of these non-standard inferences can be seen as the computation of a sum of products, where "sum" and "product" are the two operators of a bimonoid. We then show how the main ideas of automata-based axiom-pinpointing, combined with weighted model counting, yield a generic method for computing sums-of-products over arbitrary bimonoids.

1 Introduction

Description Logics (DL) [1] is a family of logic-based knowledge representation formalisms, which are employed in various application domains, like natural language processing, configuration, databases, and bio-medical ontologies. One of its most notable successes so far is the adoption of the DL-based language OWL [2] as the standard ontology language for the semantic web. For years, the main interest in the area revolved around the tradeoff between expressivity and the complexity of reasoning. Highly optimized DL reasoning systems have been developed [3–8], which can perform standard reasoning (i. e. deciding satisfiability, or subsumption between concepts) within short time bounds, even for realistic applications, where representation requires a very large number of axioms. Although these systems are still being optimized and improved, researchers are slowly turning their attention to the definition and solution of new reasoning problems. Some of these problems, like axiom-pinpointing [9, 10], refer to the extraction of more information from an unmodified knowledge base; in the case of axiom pinpointing, the goal is to detect the reason why a consequence follows. Other problems are defined by extending the expressivity of the knowledge base, not by adding new constructors, but rather by giving extended semantics to the axioms. As an example, consider the blend of uncertainty reasoning and DL [11, 12], where axioms are apended with a degree of uncertainty.

In this paper we show that some of these new inference problems can be seen as instances of the more general SumProd problem, which consists on computing sums of products of values attached to axioms, based on the sub-ontologies

Syntax	Semantics
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$
$A \doteq C$	$A^{\mathcal{I}} = C^{\mathcal{I}}$
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

Table 1. Syntax and semantics of $\mathcal{ALC}$.

from which a consequence follows. In the following section we introduce some basic notions of DL and general inference relations. We then define the SumProd problem and four of its instances that have been recently studied independently. Finally, we use the ideas of automata-based axiom pinpointing to show a reduction from the SumProd problem to weighted model counting, for which very efficient implementations exist [13]. Due to lack of space, we leave some of the proofs out of this paper.

2 Description Logics and Inference Relations

The common feature of all description logics is the use of *concepts*, that intuitively describe properties of the individuals of the domain, and *roles* that express relations between pairs of individuals. Complex *concept terms* are inductively defined with the help of a set of *constructors*, starting from a set $\mathsf{N_C}$ of *concept names* and a set $\mathsf{N_R}$ of *role names*. What distinguishes one DL from another is the set of constructors used to build concept terms. The most basic constructors are the Boolean ones: *conjunction* $\sqcap$*, disjunction* $\sqcup$, and *negation* $\neg$, and the *existential-* ($\exists$) and *value-restrictions* ($\forall$), whose syntax is shown in the first column of Table 1. The DL that uses only this constructors is called $\mathcal{ALC}$ [14].

We consider two kinds of axioms: *concept definitions* of the form $A \doteq C$, with $A \in \mathsf{N_C}$ and C a concept term, and *general concept inclusions* (GCIs) $C \sqsubseteq D$, where C, D are concept terms. An *acyclic TBox* is a finite set of concept definitions such that every concept name occurs at most once as a left-hand side, and there is no cyclic dependency between the definitions. A *general TBox* is an acyclic TBox extended with a finite set of GCIs. We will refer in general to a *TBox* whenever it is not relevant whether it is an acyclic or a general TBox.

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

One of the main decision problems in DL is concept subsumption:[1]

Definition 1. *Let C, D be two concepts and $\mathcal{T}$ a TBox. We say that C is* subsumed *by D w.r.t. $\mathcal{T}$ (denoted as $C \sqsubseteq_{\mathcal{T}} D$) if, for every model $\mathcal{I}$ of $\mathcal{T}$, it holds that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. We say that C is* satisfiable *w.r.t. $\mathcal{T}$ ($\mathcal{T} \models C$) if $C \not\sqsubseteq_{\mathcal{T}} \bot$, where $\bot$ represents any contradictory concept.*

Following [15, 16], we will introduce the SumProd problem not for a specific logic and inference problem, but rather in a more general setting. The type of inference problems that we will consider is deciding whether a so-called inference relation holds. To obtain an intuitive understanding of the following definition, just assume that consequences are $\mathcal{ALC}$ concept terms, admissible sets of axioms are $\mathcal{ALC}$ TBoxes, and the inference relation is unsatisfiablility.

Definition 2. *Let $\mathfrak{I}$ and $\mathfrak{T}$ be (possibly infinite) sets of* consequences *and* axioms*, respectively, and let $\mathscr{P}_{admis}(\mathfrak{T}) \subseteq \mathscr{P}_{fin}(\mathfrak{T})$ be a set of finite subsets of $\mathfrak{T}$ such that $\mathcal{T} \in \mathscr{P}_{admis}(\mathfrak{T})$ implies $\mathcal{T}' \in \mathscr{P}_{admis}(\mathfrak{T})$ for all $\mathcal{T}' \subseteq \mathcal{T}$.*

A relation $\vdash$ between $\mathscr{P}_{admis}(\mathfrak{T})$ and $\mathfrak{I}$ is an inference relation *if for every $\mathcal{T} \in \mathscr{P}_{admis}(\mathfrak{T}), \alpha \in \mathfrak{T}, \mathcal{T} \vdash \alpha$ implies $\mathcal{T}' \vdash \alpha$ for all $\mathcal{T}' \in \mathscr{P}_{admis}(\mathfrak{T})$ with $\mathcal{T}' \supseteq \mathcal{T}$.*

The reason why we have introduced the set $\mathscr{P}_{admis}(\mathfrak{T})$ of admissible subsets of $\mathfrak{T}$ (rather than taking all finite subsets of $\mathfrak{T}$) is to allow us to impose additional restrictions on the sets of axioms that must be considered. For instance, acyclic TBoxes are not arbitrary finite sets of concept definitions: in addition, we require that there is no cyclic dependency between axioms, and that every concept name appears at most once as a left-hand side. Clearly, these restrictions satisfy our requirement for admissible sets of axioms. For the rest of this work, we will often call an admissible set of axioms an *ontology*.

The problem of *un*satisfiability of $\mathcal{ALC}$ concepts w.r.t. TBoxes is an inference relation. More formally, let $\mathfrak{I}$ be all $\mathcal{ALC}$ concepts, $\mathfrak{T}$ all GCIs and concept definitions, and $\mathscr{P}_{admis}(\mathfrak{T})$ all TBoxes. The following is an inference relation:

$$\vdash = \{(\mathcal{T}, C) \mid C \text{ is unsatisfiable w.r.t. } \mathcal{T}\}.$$

3 The SumProd Problem

For the SumProd problem we consider that every axiom in an ontology is annotated with a value. These values can be extended to sets of axioms by computing the *product* of the values of axioms in the set. The SumProd problem consists then on computing the *sum* of the values of all subontologies from which a consequence follows. The specific instances of this problem are characterised by the choice of operators for the *sum* and the *product*. To stay as general as possible, we simply assume that there is a *bimonoid* $(M, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, where $\oplus$ is the "sum", with neutral element $\mathbf{0}$, and $\otimes$ is the "product", whose neutral element is $\mathbf{1}$.

[1] For the rest of this paper, we will often refer to *concept terms* simply as *concepts*.

Definition 3. *Let* $(M, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ *be a bimonoid,* $\mathcal{T}$ *an ontology,* α *a consequence with* $\mathcal{T} \vdash \alpha$*, and* $\mathsf{lab}_M : \mathcal{T} \to M$*. The* SumProd problem *is the task of computing*

$$\mathsf{SP}(\mathcal{T}, \alpha, \mathsf{lab}_M) := \bigoplus_{\mathcal{S} \subseteq \mathcal{T}, \mathcal{S} \vdash \alpha} \bigotimes_{t \in \mathcal{S}} \mathsf{lab}_M(t).$$

We now present some instances of this problem that have received some attention from research communities in recent years.

3.1 Pinpointing Formula

Suppose that we have a consequence α that follows from an ontology $\mathcal{T}$. The pinpointing formula is a monotone Boolean formula that describes all the subsets of $\mathcal{T}$ from wich a consequence α still follows. More formally, let $\mathsf{lab}_{\mathbb{B}}$ be a mapping that assigns to each axiom t in $\mathcal{T}$ a unique propositional variable. A monotone Boolean formula ϕ is called a *pinpointing formula* for $\mathcal{T}, \alpha$ if for every $\mathcal{S} \subseteq \mathcal{T}$ it holds: $\mathcal{S} \vdash \alpha$ iff $\bigwedge_{t \in \mathcal{S}} \mathsf{lab}_{\mathbb{B}}(t) \models \phi$.[2]

It is easy to see that if we consider as bimonoid the lattice of monotone Boolean formulae over the image of $\mathsf{lab}_{\mathbb{B}}$ (modulo equivalence) $(\mathbb{B}^+, \vee, \wedge, \bot, \top)$, then computing a pinpointing formula is an instance of the SumProd problem; i. e. $\mathsf{SP}(\mathcal{T}, \alpha, \mathsf{lab}_{\mathbb{B}})$ is a pinpointing formula for $\mathcal{T}, \alpha$. We will later show that the pinpointing formula is in fact a general solution to the SumProd problem.

3.2 Access Control

In access control we assume that there is a finite lattice $(L, \leqslant)$ that represents the levels of security in an application. Given an ontology $\mathcal{T}$, each axiom $t \in \mathcal{T}$ is assigned an element $\mathsf{lab}_L(t)$ of L. Basically, $\mathsf{lab}_L(t_1) < \mathsf{lab}_L(t_2)$ means that axiom t_2 is more public than t_1 (which is more private). Additionally, there are some users that are assigned an access level in L; that is, there is a mapping acc from the set of all users to L. The access level of a user u defines a subset of axioms that are visible to this user: $\mathcal{T}_u := \{t \in \mathcal{T} \mid \mathsf{acc}(u) \leqslant \mathsf{lab}_L(t)\}$.

Let α be a consequence such that $\mathcal{T} \vdash \alpha$. We are interested in finding a so-called boundary. An element $\mu \in L$ is called a *boundary* for $\mathcal{T}, \alpha$ under lab_L if for every user u it follows that $\mathcal{T}_u \vdash \alpha$ iff $\mathsf{acc}(u) \leqslant \mu$.

It was shown in [17] that $\mathsf{lub}_{\mathcal{S} \subseteq \mathcal{T}, \mathcal{S} \vdash \alpha} \mathsf{glb}_{t \in \mathcal{S}} \mathsf{lab}_L(t)$ is a boundary.[3] Hence, if we consider the lattice L with its lub and glb operators as a bimonoid, we obtain that the computation of a boundary is an instance of the SumProd problem. That is, $\mathsf{SP}(\mathcal{T}, \alpha, \mathsf{lab}_L)$ is a boundary for $\mathcal{T}, \alpha$ under lab_L.

3.3 Utility From Preference Formulae

We now leave behind applications where a lattice is used and allow for more general cases of bimonoids. One problem that has started to raise interest is how

[2] A monotone Boolean formula is a propositional formula that contains no negation.
[3] lub and glb denote the least upper bound and the greatest lower bound, respectively.

to compute the utility of a preference set in a negotiation process. We first define the problem of finding the minimal utility value in DL [18, 19], and then show that this problem is an instance of the SumProd problem.

Definition 4. *Let $\mathcal{T}$ be a DL ontology. A* preference *is a pair (P, v) where P is a DL concept such that $\mathcal{T} \models P$ and $v \in \mathbb{R}^+$.*

Intuitively, a preference (P, v) tells us how much *value* we assign to the satisfaction of the concept P. If we have a set of preferences $\mathcal{P}$, and are presented with a concept C (called a *proposal*) we would like to be able to know how good this proposal is related to $\mathcal{P}$, in the sense of knowing the total value of the preferences in $\mathcal{P}$ that are compatible with C. Taking the conservative approach, we want to know the *minimal utility value.*

Definition 5. *Let $\mathcal{T}$ be an ontology, C a concept such that $\mathcal{T} \models C$ and $\mathcal{P}$ a set of preferences. The* minimal utility value *for C w.r.t. $\mathcal{P}$ is given by:*

$$\mathsf{MUV}(\mathcal{T}, C, \mathcal{P}) := \min_{\mathcal{I} \models C, \mathcal{I} \models \mathcal{T}} \sum_{(P,v) \in \mathcal{P}, \mathcal{I} \models P} v.$$

Basically, the minimal utility value expresses the least value that we are expected to obtain whenever the proposal C is satisfied. In a negotiation process, we would be confronted with several proposals. We can then compare how worth each of them is w.r.t. our preference set and accept that with the highest MUV.

We now show that the problem of finding the minimal utility value is in fact an instance of SumProd. We consider the bimonoid $(\mathbb{R}^+ \cup \{0, \infty\}, \min, +, \infty, 0)$, which is a semiring, and construct a new ontology $\mathcal{T}' := \mathcal{T} \cup \{\top \sqsubseteq \neg P \mid (P, v) \in \mathcal{P}\}$.[4] The labeling $\mathsf{lab}_{\mathbb{R}} : \mathcal{T}' \to \mathbb{R}^+ \cup \{0, \infty\}$ is defined as follows:

$$\mathsf{lab}_{\mathbb{R}}(t) := \begin{cases} 0 & \text{if } t \in \mathcal{T}, \\ v & \text{if } (P, v) \in \mathcal{P}, t = \top \sqsubseteq \neg P. \end{cases}$$

Finally, given a proposal C, we consider the consequence $\alpha := C \sqsubseteq \bot$.

Theorem 1. *Let $\mathcal{T}$ be an ontology, C a concept such that $\mathcal{T} \models C$ and $\mathcal{P}$ a set of preferences. If $\mathcal{T}, \alpha$ and $\mathsf{lab}_{\mathbb{R}}$ are constructed as above, then, under the bimonoid $(\mathbb{R}^+ \cup \{0, \infty\}, \min, +, \infty, 0)$,*

$$\mathsf{SP}(\mathcal{T}', \alpha, \mathsf{lab}_{\mathbb{R}}) = \mathsf{MUV}(\mathcal{T}, C, \mathcal{P}).$$

3.4 Best Entailment Degree

Another problem that is gaining the interest of the community is the combination of DLs with reasoning under uncertainty and, in particular, with the use of fuzzy operators: *t-norm* $\boxtimes$, *t-conorm* $\boxplus$, *negation* $\boxminus$, and *implication* $\Rightarrow$. The exact semantics of fuzzy DLs depends on the specific family of fuzzy operators chosen.

[4] $\top$ represents any tautological concept.

The most important of these families are the *Zadeh* [20], the *Łukasiewicz*, the *Product* and the *Gödel* [21] families.

In fuzzy DLs, every axiom in an ontology has an associated *degree of truth*, denoted as a pair $\langle t, n\rangle$, where t is a DL axiom and $n \in [0,1]$. Intuitively, such a pair denotes that axiom t is true with a degree of at least n. The semantics of fuzzy DLs is defined by means of *fuzzy interpretations*. A fuzzy interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set $\Delta^{\mathcal{I}}$, called the *domain*, and a *fuzzy interpretation function* that assigns to each concept name $A \in \mathsf{N_C}$ a function $A^{\mathcal{I}} : \Delta^{\mathcal{I}} \to [0,1]$ and to each role name $r \in \mathsf{N_R}$ a function $r^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to [0,1]$. This function is extended to concept terms as follows: $(C_1 \sqcap C_2)^{\mathcal{I}}(x) = C_1(x)^{\mathcal{I}} \boxtimes C_2(x)^{\mathcal{I}}$; $(C_1 \sqcup C_2)^{\mathcal{I}}(x) = C_1(x)^{\mathcal{I}} \boxplus C_2(x)^{\mathcal{I}}$; $(\neg C)^{\mathcal{I}}(x) = \boxminus C^{\mathcal{I}}(x)$; $(\forall r.C)^{\mathcal{I}}(x) = \inf_{y \in \Delta^{\mathcal{I}}} r^{\mathcal{I}}(x,y) \Rightarrow C^{\mathcal{I}}(y)$; and $(\exists r.C)^{\mathcal{I}}(x) = \sup_{y \in \Delta^{\mathcal{I}}} r^{\mathcal{I}}(x,y) \boxtimes C^{\mathcal{I}}(y)$.

An interpretation $\mathcal{I}$ is a *model* of a fuzzy ontology $\mathcal{T}$ if for every $\langle C \sqsubseteq D, n\rangle \in \mathcal{T}$ it holds that $\inf_{x \in \Delta^{\mathcal{I}}}(C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x)) \geqslant n$. A fuzzy GCI $\langle C \sqsubseteq D, n\rangle$ is a *consequence* of a fuzzy ontology $\mathcal{T}$, denoted $\mathcal{T} \vdash \langle C \sqsubseteq D, n\rangle$, if every model of $\mathcal{T}$ is also a model of $\langle C \sqsubseteq D, n\rangle$.[5]

Definition 6. *Let $\mathcal{T}$ be a fuzzy ontology and t a (crisp) GCI. The* best entailment degree *of t w.r.t. $\mathcal{T}$ is*

$$\mathsf{BED}(\mathcal{T}, t) := \sup_{\mathcal{T} \vdash \langle t, n\rangle} n.$$

Briefly, the best entailment degree expresses the best bound that can be given on the fuzzy value at which t follows from the ontology $\mathcal{T}$. This problem is in fact an instance of the SumProd problem over the bimonoid $([0,1], \max, \boxtimes, 0, 1)$, where $\boxtimes$ is the t-norm being used and the function $\mathsf{lab}_{[0,1]}$ maps every GCI in $\mathcal{T}$ to its associated degree of truth. The following theorem holds for the Łukasiewicz, Product, and Gödel families of fuzzy operators, but not for the Zadeh family.

Theorem 2. *Let $\mathcal{T}$ be an ontology, $\mathsf{lab}_{[0,1]} : \mathcal{T} \to [0,1]$ the function assigning, to every axiom in $\mathcal{T}$, its associated degree of truth, and t a (crisp) GCI. Then, under the bimonoid $([0,1], \max, \boxtimes, 0, 1)$,*

$$\mathsf{SP}(\mathcal{T}, t, \mathsf{lab}_{[0,1]}) = \mathsf{BED}(\mathcal{T}, t).$$

4 Solving the SumProd Problem

It was shown in [22] that the pinpointing formula is the most general solution of the SumProd problem over distributive lattices: given an arbitrary distributive lattice M, the (unique) homomorphism from $\mathbb{B}^+$ to M can be used to compute $\mathsf{SP}(\mathcal{T}, \alpha, \mathsf{lab}_M)$ from the pinpointing formula for $\mathcal{T}, \alpha$. In fact, the pinpointing formula can be used to solve the SumProd problem over any bimonoid, through *weighted model counting*.

[5] For simplicity, we are restricting ourselves to the case where both, the axioms in the ontology and the consequences, are concept inclusions. For settings dealing with a wider variety of axioms and consequences, see, e.g. [11].

Definition 7. *Let $(M, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ be a bimonoid, V a set of propositional variables, ψ a Boolean formula over V and wt a function that maps every literal corresponding to a variable in V to an element of M.* Weighted model counting *corresponds to the task of computing*

$$\mathsf{WMC}(\psi, \mathsf{wt}) := \bigoplus_{\mathcal{V} \models \psi} \bigotimes_{\ell \in \mathcal{V}} \mathsf{wt}(\ell).$$

Let $\mathcal{T}$ be an ontology and α a consequence such that $\mathcal{T} \vdash \alpha$. If ϕ is a pinpointing formula for $\mathcal{T}, \alpha$, then by definition there is a bijective function $\mathsf{lab}_{\mathbb{B}}$ between $\mathcal{T}$ and a superset of the propositional variables appearing in ϕ. Let now $(M, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ be a bimonoid and $\mathsf{lab}_M : \mathcal{T} \to M$. We construct the function wt as follows: for every positive literal p, we set $\mathsf{wt}(p) = \mathsf{lab}_M(\mathsf{lab}_{\mathbb{B}}^{-1}(p))$, and for every negative literal $\neg p$, we set $\mathsf{wt}(\neg p) = \mathbf{1}$.

Theorem 3. *Let ϕ be a pinpointing formula for $\mathcal{T}, \alpha$ and wt built as above. Then $\mathsf{SP}(\mathcal{T}, \alpha, \mathsf{lab}_M) = \mathsf{WMC}(\phi, \mathsf{wt})$.*

Proof. Let $\mathcal{V}$ be a valuation and $p_1, \ldots, p_n$ the positive literals appearing in $\mathcal{V}$, and set $\mathcal{S} = \{\mathsf{lab}_{\mathbb{B}}^{-1}(p_i) \mid 1 \leqslant i \leqslant n\}$. As ϕ is a pinpointing formula, we have that $\mathcal{S} \vdash \alpha$ iff $\bigwedge_{i=1}^{n} p_i \models \phi$ iff $\mathcal{V} \models \phi$. Additionally, $\bigotimes_{t \in \mathcal{S}} \mathsf{lab}_M(t) = \bigotimes_{i=1}^{n} \mathsf{lab}_M(\mathsf{lab}_{\mathbb{B}}^{-1}(p_i)) = \bigotimes_{i=1}^{n} \mathsf{wt}(p_i) = \bigotimes_{\ell \in \mathcal{V}} \mathsf{wt}(\ell)$. Hence, we have that

$$\mathsf{SP}(\mathcal{T}, \alpha, \mathsf{lab}_M) = \bigoplus_{\mathcal{S} \vdash \alpha} \bigotimes_{t \in \mathcal{S}} \mathsf{lab}_M(t) = \bigoplus_{\mathcal{V} \models \phi} \bigotimes_{\ell \in \mathcal{V}} \mathsf{wt}(\ell) = \mathsf{WMC}(\phi, \mathsf{wt}).$$

□

This theorem shows that if one has a pinpointing formula for some ontology $\mathcal{T}$ and consequence α, then one can solve any instance of the SumProd problem related to $\mathcal{T}, \alpha$ through a call to a weighted model counter. It has been shown that the pinpointing formula can be computed by a modified version of the decision algorithm used to verify that $\mathcal{T} \vdash \alpha$. Recently, general approaches that modify tableaux- [15, 9] and automata-based [16, 10] decision procedures have been developed. However, the formulas obtained by these methods are in general form, with conjunctions and disjunctions nested within each other, while the efficiency of modern weighted model counters relies on having an input formula in CNF.

It is well-known that for every formula ψ it is possible to construct in polynomial time (on the length of ψ) a formula ψ' in CNF such that there is a bijection between the models of ψ and the models of ψ' [23]. The idea consists in introducing new variables that capture complex subformulae of ψ. By setting to $\mathbf{1}$ the weights of all newly added literals, we can ensure that both formulas are also equivalent w.r.t. weighted model counting. Although this does not affect the overall complexity of the method, it introduces an unnecessary step. In fact, it is possible to improve the structure-sharing idea used in [10] to directly obtain a formula that is equivalent (with respect to WMC) to the pinpointing formula.

We first recall the necessary notions for automata-based pinpointing, and then show how these ideas yield a formula in CNF that can be used to solve any instance of SumProd.

4.1 Axiomatic Automata

We consider Büchi automata over trees, whose input alphabet has only one element of arity k. A *Büchi automaton for arity k* is a tuple (Q, Δ, I, F), where Q is a finite set of *states*, $\Delta \subseteq Q^{k+1}$ is the set of transitions, and $I, F \subseteq Q$ are the set of *initial* and *final states*, respectively. A *weighted Büchi automaton* (WBA) over a lattice L is a tuple (Q, in, wt, F), where Q is a finite set of states, $in : Q \to L, wt : Q^{k+1} \to L$, are the *initial* and *transition distribution*, and $F \subseteq Q$.

The reasoning necessary for the computation of the pinpointing formula (and, in general, the SumProd problem) for $\mathcal{T}, \alpha$, needs to know for which subontologies $\mathcal{T}'$ of $\mathcal{T}$, $\mathcal{T}' \vdash \alpha$ holds. Thus, we assume that the automaton $\mathcal{A}_{\mathcal{T},\alpha}$ for deciding $\mathcal{T} \vdash \alpha$ also contains automata for all axiomatized inputs $\mathcal{T}', \alpha$ with $\mathcal{T}' \subseteq \mathcal{T}$,[6] which can be obtained by appropriately restricting the states and transitions of $\mathcal{A}_{\mathcal{T},\alpha}$. To be more precise, let $\mathcal{A} = (Q, \Delta, I, F)$ be a Büchi automaton for arity k, $\mathcal{T}$ an ontology and α a consequence. The functions $\Delta\mathsf{res} : \mathcal{T} \to \mathscr{P}(Q^{k+1})$ and $I\mathsf{res} : \mathcal{T} \to \mathscr{P}(Q)$ are respectively called a *transition restricting function* and an *initial restricting function*. The restricting functions $\Delta\mathsf{res}$ and $I\mathsf{res}$ are extended to sets of axioms $\mathcal{T}' \subseteq \mathcal{T}$ as follows:

$$\Delta\mathsf{res}(\mathcal{T}') := \bigcap_{t \in \mathcal{T}'} \Delta\mathsf{res}(t) \quad \text{and} \quad I\mathsf{res}(\mathcal{T}') := \bigcap_{t \in \mathcal{T}'} I\mathsf{res}(t).$$

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

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

Definition 8. *Let $\mathcal{A} = (Q, \Delta, I, F)$ be a Büchi automaton for arity k, $\mathcal{T}$ an ontology, α a consequence, and $\Delta\mathsf{res} : \mathcal{T} \to \mathscr{P}(Q^{k+1})$ and $I\mathsf{res} : \mathcal{T} \to \mathscr{P}(Q)$ a transition and an initial restricting function, respectively. We call $(\mathcal{A}, \Delta\mathsf{res}, I\mathsf{res})$ an* axiomatic automaton *for Γ.*

Given an inference relation $\vdash$, we say that $(\mathcal{A}, \Delta\mathsf{res}, I\mathsf{res})$ is correct *for $\mathcal{T}, \alpha$ w.r.t. $\vdash$ if the following holds for every $\mathcal{T}' \subseteq \mathcal{T}$: $\mathcal{T}' \vdash \alpha$ iff $\mathcal{A}_{|\mathcal{T}'}$ does not have a successful run r with $r(\varepsilon) \in I \cap I\mathsf{res}(\mathcal{T}')$.*

Given a correct axiomatic automaton for $\mathcal{T}, \alpha$, we can decide $\mathcal{T}' \vdash \alpha$ for $\mathcal{T}' \subseteq \mathcal{T}$ through an emptiness test on the automaton $\mathcal{A}_{|\mathcal{T}'}$. Any correct axiomatic automaton can be transformed into a *pinpointing automaton*: a weighted Büchi automaton whose behaviour is a pinpointing formula for the input.

Recall first that in the definition of pinpointing formula we consider a mapping $\mathsf{lab}_{\mathbb{B}}$ assigning a unique propositional variable to each $t \in \mathcal{T}$. The pinpointing automaton takes its weights from the $\mathcal{T}$-Boolean bimonoid $(\mathbb{B}(\mathcal{T}), \wedge, \vee, \top, \bot)$, where $\mathbb{B}(\mathcal{T})$ is the quotient set of all monotone Boolean formulae over $\mathsf{lab}_{\mathbb{B}}(\mathcal{T})$ by the propositional equivalence relation, i.e., two propositionally equivalent formulae correspond to the same element of $\mathbb{B}(\mathcal{T})$. It is easy to see that this bimonoid is in fact a finite distributive lattice, where the partial order is defined as $\phi \leqslant \psi$

[6] Recall that every subset of an admissible set of axioms is also admissible.

iff $\psi \to \phi$ is valid.[7] Note that this bimonoid is different from the one used in Section 3.1, in that the two operations are exchanged. This is done to follow the construction in [10] and be able to reuse their results.

Definition 9. *Let* $(\mathcal{A}, \Delta\mathsf{res}, I\mathsf{res})$ *be an axiomatic automaton for* $\mathcal{T}, \alpha$*, with* $\mathcal{A} = (Q, \Delta, I, F)$*. The* violating functions $\Delta\mathsf{vio} : Q^{k+1} \to \mathbb{B}(\mathcal{T})$ *and* $I\mathsf{vio} : Q \to \mathbb{B}(\mathcal{T})$ *are*

$$\Delta\mathsf{vio}(q_0, q_1, \ldots, q_k) := \bigvee_{\{t \in \mathcal{T} \mid (q_0, q_1, \ldots, q_k) \notin \Delta\mathsf{res}(t)\}} \mathsf{lab}(t);$$

$$I\mathsf{vio}(q) := \bigvee_{\{t \in \mathcal{T} \mid q \notin I\mathsf{res}(t)\}} \mathsf{lab}(t).$$

The pinpointing automaton *induced by* $(\mathcal{A}, \Delta\mathsf{res}, I\mathsf{res})$ *w.r.t.* $\mathcal{T}$ *is the WBA over* $\mathbb{B}^{\mathcal{T}}$ $(\mathcal{A}, \Delta\mathsf{res}, I\mathsf{res})^{\mathsf{pin}} = (Q, in, wt, F)$*, where*

$$in(q) := \begin{cases} I\mathsf{vio}(q) & \text{if } q \in I, \\ \top & \text{otherwise;} \end{cases}$$

$$wt(q_0, q_1, \ldots, q_k) := \begin{cases} \Delta\mathsf{vio}(q_0, q_1, \ldots, q_k) & \text{if } (q_0, q_1, \ldots, q_k) \in \Delta, \\ \top & \text{otherwise.} \end{cases}$$

As shown in [10], the behaviour of the pinpointing automaton yields the pinpointing formula. However, the iterative approach for computing the behaviour of a weighted automaton requires an alternation of the operators $\otimes$ and $\oplus$, and hence, when grounded to the bimonoid $\mathbb{B}(\mathcal{T})$, the formula obtained this way is not in CNF. Furthermore, in order to ensure a polynomially bounded execution time, it was necessary to resort to a compact encoding of the generated formula, using *structure sharing*. Translating this encoding into a CNF formula may result in an exponential blowup.

Fortunately, it is possible to modify the above mentioned iterative approach so that it explicitely exploits the idea of structure sharing by adding new variables during the construction of the formula. The result of this modification is an algorithm that outputs a formula ψ in CNF such that every valuation satisfying the pinpointing formula can be uniquely extended to a valuation satisfying ψ, and conversely, every valuation that satisfies ψ, satisfies also the pinpointing formula. We now show how these changes can be made.[8]

4.2 Computing a CNF Formula

We first briefly recall the iterative method for computing the behaviour of the pinpointing automaton and some of its properties. We later show how it can be used to compute the desired CNF formula.

[7] More precisely, $\mathbb{B}(\mathcal{T})$ is the free distributive lattice over the generators $\mathsf{lab}_{\mathbb{B}}(\mathcal{T})$.
[8] For the DL $\mathcal{EL}$, our approach reduces to the one in [24]

In the following we assume that we have a pinpointing automaton $\mathcal{A} = (Q, in, wt, F)$. The iterative method defines operators $\mathcal{O}_f, \mathcal{Q} : \mathbb{B}(\mathcal{T})^Q \to \mathbb{B}(\mathcal{T})^Q$, where $\mathbb{B}(\mathcal{T})^Q$ denotes the set of all mappings from Q to $\mathbb{B}(\mathcal{T})$, and $f \in \mathbb{B}(\mathcal{T})^Q$. The operator $\mathcal{O}_f$ is defined as follows for every $\sigma \in \mathbb{B}(\mathcal{T})^Q$:

$$\mathcal{O}_f(\sigma)(q) = \bigwedge_{(q,q_1,\dots,q_k)\in Q^{k+1}} \left(wt(q, q_1, \dots, q_k) \vee \bigvee_{j=1}^{k} \mathsf{step}_f(\sigma)(q_j) \right),$$

where

$$\mathsf{step}_f(\sigma)(q) = \begin{cases} f(q) & \text{if } q \in F \\ \sigma(q) & \text{otherwise.} \end{cases}$$

This operator is monotonic, and hence it makes sense to speak about its least fixpoint (lfp). The operator $\mathcal{Q}$ is based in this lfp: given $\sigma \in \mathbb{B}(\mathcal{T})^Q$,

$$\mathcal{Q}(\sigma) = \mathsf{lfp}(\mathcal{O}_\sigma).$$

The operator $\mathcal{Q}$ is also monotonic, and thus it has a greatest fixpoint (gfp). The following result is a direct consequence of those in [10].

Lemma 1. *Let $\varsigma = \mathsf{gfp}(\mathcal{Q})$. Then $\bigwedge_{q\in Q} in(q) \vee \varsigma(q)$ is a pinpointing formula.*

The results in [10] are in fact stronger, since they also set a bound, depending on the number of states and the number of final states, on the times the operators need to be applied before obtaining the fixpoints.

Lemma 2. *Let $n = |Q|, m = |F|$, and denote as $\widetilde{\top}, \widetilde{\bot}$ the functions that map every state in Q to $\top$ and $\bot$, respectively. The following two results hold:*

$$\mathsf{lfp}(\mathcal{O}_f) = \mathcal{O}_f^{n-m+1}(\widetilde{\top}), \qquad \mathsf{gfp}(\mathcal{Q}) = \mathcal{Q}^m(\widetilde{\bot}).$$

In order to construct a formula in CNF, we are going to simulate applications of the operators $\mathcal{O}_f$ and $\mathcal{Q}$, introducing new variables that will stand as abbreviations of the formulas constructed at each application. The total number of variables and clauses introduced this way will be polynomially bounded by the size of the automaton, due to Lemma 2.

We introduce the variables $x_{\zeta,q}, y^\eta_{\zeta,q}$, and $z^\eta_{\zeta,(q,q_1,\dots,q_k)}$. Intuitively, the variable $x_{\zeta,q}$ is an abbreviation for the formula $\mathcal{Q}^\zeta(\widetilde{\bot})(q)$. Likewise, the variable $y^\eta_{\zeta,q}$ represents the value of $\mathcal{O}^\eta_{\mathcal{Q}^\zeta(\widetilde{\bot})}(\widetilde{\top})(q)$. The other variables are used as auxiliary means for keeping the formula in CNF. The formula φ_{CNF} is composed by the following clauses, where $0 \leqslant \zeta < m, 0 \leqslant \eta < n - m + 1, q, q_1, \dots, q_k \in Q$:[9]

$$x_{0,q} \Leftrightarrow \bot, \qquad y^0_{\zeta,q} \Leftrightarrow \top,$$

$$x_{\zeta+1,q} \Leftrightarrow y^{n-m+1}_{\zeta,q}, \qquad y^{\eta+1}_{\zeta,q} \Leftrightarrow \bigwedge_{(q,q_1,\dots,q_k)\in Q^{k+1}} z^{\eta+1}_{\zeta,(q,q_1,\dots,q_k)},$$

[9] For brevity, we use double implications rather than clauses. These implications can easily be transformed in clausal form, thus yielding a CNF formula.

$$z^{\eta+1}_{\zeta,(q,q_1,\ldots,q_k)} \Leftrightarrow wt(q,q_1,\ldots,q_k) \vee \bigvee_{j=1}^{k} \mathsf{choice}^{\eta}_{\zeta}(q_j),$$

where

$$\mathsf{choice}^{\eta}_{\zeta}(q) = \begin{cases} x_{\zeta,q} & \text{if } q \in F \\ y^{\eta}_{\zeta,q} & \text{otherwise.} \end{cases}$$

Finally, we add for every $q \in Q$ the clause

$$in(q) \vee x_{m,q}.$$

Notice that the new variables are effectively nothing more than abbreviations for longer formulas. The truth value of each of them depends ultimately only on the truth value of the original propositional variables used for defining the function wt. The last clauses introduced simply use the definition of pinpointing formula from Lemma 1. The following result is a direct consequence of Lemmas 1 and 2.

Theorem 4. *Let ϕ be a pinpointing formula and φ_{CNF} the formula in CNF constructed above. Then, every valuation $\mathcal{V}$ satisfying ϕ can be uniquely extended to a valuation $\mathcal{V}'$ satisfying φ_{CNF}. Conversely, every valuation that satisfies φ_{CNF} satisfies also ϕ.*

Corollary 1. *Let φ_{CNF} be constructed as above, and wt built as for Theorem 3, and extended to the new literals by setting $\mathsf{wt}(\ell) = 1$ for all new literal ℓ. Then $\mathsf{SP}(\mathcal{T}, \alpha, \mathsf{lab}_M) = \mathsf{WMC}(\varphi_{\mathsf{CNF}}, \mathsf{wt})$.*

5 Conclusions

We have shown that some of the recently studied non-standard inference problems can be seen as instances of the general SumProd problem. We have also shown that the ideas of automata-based axiom pinpointing can be adapted to reduce the SumProd problem to a weighted model counting problem (with the input formula in CNF).

As future work we would like to find more non-standard inferences that fall into the framework described in this paper, for distinct inference relations, also beyond the realm of DL. Additionally, we would like to empirically test our approach by introducing the formula φ_{CNF} into a state-of-the-art weighted model counter. We want then to compare the execution time to other *ad-hoc* implementations, such as the black-box method for computing the boundary in access control [17] or the algorithm for MUV from [18].

References

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

2. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. J. of Web Sem. **1**(1) (2003) 7–26
3. Haarslev, V., Möller, R.: RACER system description. In: Proc. of IJCAR'01. (2001)
4. Sirin, E., Parsia, B.: Pellet: An OWL DL reasoner. In: Proc. of DL'04. (2004) 212–213
5. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. In: Proc. of IJCAR'06. Volume 4130 of LNCS., Springer (2006) 292–297
6. Baader, F., Lutz, C., Suntisrivaraporn, B.: CEL—a polynomial-time reasoner for life science ontologies. In: Proc. of IJCAR'06. Volume 4130 of LNAI., Springer (2006) 287–291
7. Motik, B., Shearer, R., Horrocks, I.: Optimized reasoning in description logics using hypertableaux. In: Proc. of CADE'07. Volume 4603 of LNCS., Springer (2007) 67–83
8. Kazakov, Y.: Consequence-driven reasoning for horn shiq ontologies. In Boutilier, C., ed.: Proc. of IJCAI'09, Pasadena, California (2009) 2040–2045
9. Baader, F., Peñaloza, R.: Axiom pinpointing in general tableaux. Journal of Logic and Computation (2010) Special Issue: Tableaux'07. To appear.
10. Baader, F., Peñaloza, R.: Automata-based axiom pinpointing. Journal of Automated Reasoning (2010) Special Issue: IJCAR 2008. To appear.
11. Bobillo, F., Straccia, U.: Fuzzy description logics with general t-norms and datatypes. Fuzzy Sets and Systems **160**(23) (2009) 3382–3402
12. Lukasiewicz, T.: Expressive probabilistic description logics. Artif. Intel. **172**(6-7) (2008) 852–883
13. Bacchus, F., Dalmao, S., Pitassi, T.: Solving #sat and bayesian inference with backtracking search. J. of Art. Intel. Research **34** (2009) 391–442
14. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. Artif. Intel. **48**(1) (1991) 1–26
15. Baader, F., Peñaloza, R.: Axiom pinpointing in general tableaux. In: Proc. of TABLEAUX 2007. Volume 4548 of LNAI., Springer (2007) 11–27
16. Baader, F., Peñaloza, R.: Automata-based axiom pinpointing. In: Proc. of IJCAR 2008. Volume 4667 of LNAI., Springer (2008) 226–241
17. Baader, F., Knechtel, M., Peñaloza, R.: A generic approach for large-scale ontological reasoning in the presence of access restrictions to the ontology's axioms. In: Proc. of ISWC 2009. (2009)
18. Ragone, A., Noia, T.D., Donini, F.M., Sciascio, E.D., Wellman, M.P.: Weighted description logics preference formulas for multiattribute negotiation. In Godo, L., Pugliese, A., eds.: Proc. of SUM'09. Volume 5785 of LNCS., Washington, DC, USA, Springer (2009) 193–205
19. Ragone, A., Noia, T.D., Donini, F.M., Sciascio, E.D., Wellman, M.P.: Computing utility from weighted description logic preference formulas. In Baldoni, M., Bentahar, J., Lloyd, J., van Riemsdijk, M.B., eds.: Proc. of DALT'09, Budapest, Hungary, Springer (2009)
20. Zadeh, L.A.: Fuzzy sets. Information and Control **8**(3) (1965) 338–353
21. Hájek, P.: Metamathematics of Fuzzy Logic. Kluwer, Dordrecht (2001)
22. Peñaloza, R.: Reasoning with weighted ontologies. In Grau, B.C., Horrocks, I., Motik, B., Sattler, U., eds.: Proc. of DL'09. Volume 477 of CEUR-WS. (2009)
23. Tseitin, G.S.: On the complexity of derivations in the propositional calculus. In: Studies in Mathematics and Mathematical Logic, Part II. (1968)
24. Sebastiani, R., Vescovi, M.: Axiom pinpointing in lightweight description logics via horn-sat encoding and conflict analysis. In: Proc. of CADE 2009. (2009) 84–99