A Quantitative Approach to Belief Revision in Structured Probabilistic Argumentation

Gerardo I. Simari · Paulo Shakarian · Marcelo A. Falappa

Received: July 30, 2015/ Accepted: date

Abstract Many real-world knowledge-based systems must deal with information coming from different sources that invariably leads to incompleteness, overspecification, or inherently uncertain content. The presence of these varying levels of uncertainty doesn't mean that the information is worthless – rather, these are hurdles that the knowledge engineer must learn to work with. In this paper, we continue work on an argumentation-based framework that extends the well-known Defeasible Logic Programming (DeLP) language with probabilistic uncertainty, giving rise to the Defeasible Logic Programming with Presumptions and Probabilistic Environments (DeLP3E) model. Our prior work focused on the problem of belief revision in DeLP3E, where we proposed a non-prioritized class of revision operators called **AFO** (Annotation Function-based Operators) to solve this problem. In this paper, we further study this class and argue that in some cases it may be desirable to define revision operators that take quantitative aspects into account, such as how the probabilities of certain literals or formulas of interest change after the revision takes place. To the best of our knowledge, this problem has not been addressed in the argumentation literature to date. We propose the **QAFO** (Quantitative Annotation Function-based Operators) class of operators, a subclass of **AFO**, and then go on to study the complexity of several problems related to their specification and application in revising knowledge bases. Finally, we present an algorithm for computing the probability that a literal is warranted in a DeLP3E knowledge base, and discuss how it could be applied towards implementing **QAFO**-style operators that compute approximations rather than exact operations.

G.I. Simari, M.A. Falappa
Dept. of Comp. Sci. and Eng., Universidad Nacional del Sur and CONICET, Argentina
E-mail: {gis,mfalappa}@cs.uns.edu.ar

P. Shakarian
Arizona State University, Tempe, AZ, USA
E-mail: shak@asu.edu

Keywords Structured Argumentation · Belief Revision · Reasoning under Probabilistic Uncertainty

1 Introduction and Motivation

Many real-world knowledge-based systems must deal with information coming from different sources that invariably leads to uncertain content, be it from gaps in knowledge (incompleteness), overspecification (inconsistency), or because the knowledge is inherently uncertain (such as weather forecasts or measurements that are necessarily imprecise). Far from considering such uncertain knowledge useless, knowledge engineers face the challenge of putting it to its best possible use when solving a wide range of problems. In particular, one basic problem that needs to be investigated in depth is that of *revising* such knowledge bases in a principled manner. In this paper, we continue work on that combines the well-known Defeasible Logic Programming (DeLP) language (actually, an extension called PreDeLP to handle presumptions) with probabilistic uncertainty, which led to the development of the DeLP3E model (Defeasible Logic Programming with Presumptions and Probabilistic Environments). In previous work [39,41], we studied a class of non-prioritized belief revision operators called **AFO** that allows changes to be made only to probabilistic annotations when addressing the incorporation of an epistemic input into the knowledge base. Here, we propose a subclass called **QAFO** that takes quantitative aspects into account when performing revisions, such as how the probabilities of certain literals or formulas of interest change after the revision takes place. To the best of our knowledge, this problem has not been addressed in the argumentation literature to date.

The main contributions presented in this paper are briefly summarized as follows:

- As building blocks to be used in our quantitative belief revision operators, we define: (i) warrant probability functions (WPFs), which have as domains either conjunctions or disjunctions of ground literals that are mapped to the probability that they are warranted in the DeLP3E program; and (ii) revision objective functions (ROFs), which take as input two DeLP3E programs $\mathcal{I}_1$, $\mathcal{I}_2$ and an epistemic input and return a numeric score that is interpreted as the value of obtaining $\mathcal{I}_2$ from $\mathcal{I}_1$ when revising it by the epistemic input. ROFs generally include WPFs in their definitions.
- We propose the **QAFO** class of operators, a subclass of **AFO** that allows modifications to the annotation function to be carried out as part of the belief revision operation, but focusing on optimizing a given ROF, as described above.
- We study the complexity of several problems related to our approach; in particular, we present:

(i) A new lower bound on the complexity of deciding the warrant status of a literal or a conjunction/disjunction of literals in a (classical) PreDeLP program;

(ii) Computing WPFs in general is #P-hard;

(iii) Point (ii) holds even when computing probabilities of worlds in the environmental model can be done in polynomial time;

(iv) Computing WPFs in the special case in which Nilsson's probabilistic logic [33] is used is NP-complete;

(v) By combining the intuition behind the proof of point (iv) and further conditions, we identify a class of instances for which WPFs can be computed in polynomial time; and

(vi) Under the same conditions as point (v), we show that **QAFO** revisions are NP-complete; furthermore, we show that the problem has the same complexity even when the revision objective function is constrained to be a simple sum of warranting probabilities for atoms in the AM.

– We present an algorithm for computing the probability that a literal is warranted in a DeLP3E knowledge base, and discuss its application towards implementing **QAFO**-style operators that compute approximations rather than perform exact operations.

The rest of this paper is organized as follows: Section 2 discusses preliminary concepts and the DeLP3E framework, which was first introduced in [39, 41]. Section 3 motivates the specialization of **AFO** operators to take into account quantitative aspects, and presents the class **QAFO**. Section 4 studies the complexity of several problems related to **QAFO** and the application of such operators in revising DeLP3E knowledge bases. Section 5 studies a novel approach to reasoning about probabilities of literals in DeLP3E, called warranting formulas, and their application in the implementation of **QAFO**-style operators that address the high computational cost issuess by applying heuristics the trade off optimality for tractability. Finally, Sections 6 and 7 present related work and conclusions, respectively.

Throughout the entire paper, we illustrate the presentation via a running example that is inspired on the use of DeLP3E in medical diagnosis, which is the kind of real-world application in which we envision our work being of most use; we have also explored its application in the related scenario of solving the attribution problem[1] in cyber security and cyber warfare [40], with encouraging feedback from the community as to its potential impact.

[1] Essentially, given a cyber event of interest, the attribution problem involves finding out who was responsible for it. This is especially well suited for argumentation and belief revision due to the ease with which a potential culprit can plant false or misleading clues, hence giving rise to an inconsistent knowledgebase. See [38] for further discussion.

2 Preliminaries on the DeLP3E Framework

The DeLP3E framework is comprised of two distinct, but interrelated models of the world. The first is called the *environmental model* (referred to from now on as the "EM"), and is used to describe uncertain knowledge about the domain that is subject to probabilistic events. The second one is called the *analytical model* (referred to as the "AM"), and contains knowledge that is either strict or defeasible, as described below – this will be useful in the analysis of competing hypotheses that can account for a given phenomenon (what we will generally call queries).

The AM is composed of a classical (that is, non-probabilistic) PreDeLP [32] program in order to allow for contradictory information, giving the system the capability to model competing explanations for a given query. In general, the EM contains knowledge such as evidence, uncertain facts, or knowledge about agents and systems. The AM, on the other hand, contains knowledge that may or may not be strictly valid, yet it does not depend on probabilistic events. Indeed, dividing knowledge between the AM and EM is a knowledge engineering task, and its adequate resolution will call for design decisions to be made; note that this separation also allows for a different kind of uncertainty to be modeled – defeasible rules and presumptions can be leveraged when there is no probabilistic information available but we still wish to maintain that a specific portion of the knowledge base is uncertain.

In the rest of this section, we formally describe these two models, as well as how knowledge in the AM can be annotated with information from the EM – these annotations specify the conditions under which the various statements in the AM can potentially be true.

Basic Language. We assume sets of variables and constants, denoted with $\mathbf{V}$ and $\mathbf{C}$, respectively. The language also contains a set of n-ary predicate symbols; the EM and AM use separate sets of predicate symbols, denoted with $\mathbf{P}_{EM}$ and $\mathbf{P}_{AM}$, respectively – the two models can, however, share variables and constants. As usual, a *term* is composed of either a variable or a constant. Given terms $t_1, ..., t_n$ and n-ary predicate symbol p, $p(t_1, ..., t_n)$ is called an *atom*; if $t_1, ..., t_n$ are constants, then the atom is said to be *ground*. The sets of all ground atoms for EM and AM are denoted with $\mathbf{G}_{EM}$ and $\mathbf{G}_{AM}$, respectively; finally, we also use notation $\mathbf{L}_{AM}$ to denote the set of all (ground) literals: $\{a \mid a \in \mathbf{G}_{AM}\} \cup \{\neg a \mid a \in \mathbf{G}_{AM}\}$. *Note that even though in general the language allows variables, for simplicity we will assume throughout this paper that all objects are ground (propositional).*

Example 1 Consider a medical diagnosis scenario[2] in which a patient goes to a hospital exhibiting certain symptoms: shortness of breath, sporadic fainting (loss of consciousness for brief periods of time), and some signs of memory

[2] This and all related examples in this paper, though inspired by potential real-world applications, make many simplifying assumptions in order to allow for a concise presentation of the key concepts that we wish to illustrate.

$\mathbf{P}_{EM}$:	*anx_risk*	The patient falls into the category of people at risk of suffering anxiety.
	dep_risk	The patient falls into the category of people at risk of suffering depression.
	FN-anx_test	Event associated with a false negative coming up when performing a test to see whether a patient is showing anxiety-related symptoms.
	FN-tox_screen	Event associated with a false negative coming up when performing a blood test for the presence of toxic substances.
$\mathbf{P}_{AM}$:	*mem_loss*	The patient shows signs of memory loss.
	short_breath	The patient suffers shortness of breath.
	fainting	The patient suffers short-term loss of consciousness.
	anxiety	The patient is diagnosed with an anxiety-related disorder.
	depression	The patient is diagnosed with a depression-related disorder.
	sleep_aid_misuse	Patient diagnosed with misuse of sleeping aids.
	neg_anx_test	Test proves absence of anxiety-related disorders.
	neg_tox_screen	Test proves absence of toxins in blood.
	pos_dep_test	Test proves presence of depression-related disorders.

Fig. 1 Explanation of the meaning of the predicates used in the running example.

loss. Figure 1 shows the predicates that we will use throughout the paper in the running example.

As shown in the figure (and discussed in more detail below), some of these predicates comprise the analytical model (AM), while others are part of the environmental model (EM). For instance, in our example, predicates stating the presence of symptoms such as memory loss and shortness of breath, as well as those representing diagnoses, such as anxiety and depression, are part of the analytical model. On the other hand, the environmental model contains predicates that are associated with uncertain events, such as false negatives coming up when a test is performed or the risk that the patient will be affected by anxiety-related disorders. Note that in the running example we make use of the term "at risk" according to the common use of this expression in the medical domain, i.e., characterized by high risk or susceptibility, such as to a certain disease. ∎

Given set of ground atoms, a *world* is any subset of atoms – those that belong to the set are said to be *true* in the world, while those that do not are *false*. Therefore, there are $2^{|\mathbf{G}_{EM}|}$ possible worlds in the EM and $2^{|\mathbf{G}_{AM}|}$ worlds in the AM. These sets are denoted with $\mathcal{W}_{EM}$ and $\mathcal{W}_{AM}$, respectively. In order to avoid worlds that do not model possible situations given a particular domain, we include *integrity constraints* of the form $\mathsf{oneOf}(\mathcal{A}')$, where $\mathcal{A}'$ is a subset of ground atoms. Intuitively, such a constraint states that any world

where more than one of the atoms from set $\mathcal{A}'$ appears is invalid. We use $\mathbf{IC}_{EM}$ and $\mathbf{IC}_{AM}$ to denote the sets of integrity constraints for the EM and AM, respectively, and the sets of worlds that conform to these constraints is denoted with $\mathcal{W}_{EM}(\mathbf{IC}_{EM})$ and $\mathcal{W}_{AM}(\mathbf{IC}_{AM})$, respectively.

Finally, logical formulas arise from the combination of atoms using the traditional connectives ($\wedge$, $\vee$, and $\neg$). As usual, we say that a world λ *satisfies* formula (f), written $\lambda \models f$, iff: (i) If f is an atom, then $\lambda \models f$ iff $f \in \lambda$; (ii) if $f = \neg f'$ then $\lambda \models f$ iff $\lambda \not\models f'$; (iii) if $f = f' \wedge f''$ then $\lambda \models f$ iff $\lambda \models f'$ and $\lambda \models f''$; and (iv) if $f = f' \vee f''$ then $\lambda \models f$ iff $\lambda \models f'$ or $\lambda \models f''$. We use the notation $form_{EM}, form_{AM}$ to denote the set of all possible (ground) formulas in the EM and AM, respectively; finally, we use $basic_{AM}$ to denote all possible conjunctions or disjunctions of literals from $\mathbf{L}_{AM}$, which we refer to as *basic formulas*.

2.1 Environmental Model

In this paper, we generalize the approach taken in our previous work [39,41] for the environmental model – here, we simply assume that we have a probabilistic model defined over $\mathbf{G}_{EM}$, which represents a probability distribution over $\mathcal{W}_{EM}$.

Definition 1 (Probabilistic Model) Given sets $\mathbf{P}_{EM}$, $\mathbf{V}$, and $\mathbf{C}$, and corresponding sets $\mathbf{G}_{EM}$ and $\mathcal{W}_{EM}$, a *probabilistic model* Π_{EM} is any function $Pr : \mathcal{W}_{EM} \rightarrow [0, 1]$ such that $\sum_{\lambda \in \mathcal{W}_{EM}} Pr(\lambda) = 1$.

Examples of probabilistic models that can be used are Bayesian networks (BNs) [34], Markov logic networks (MLNs) [37], extensions of first order logic such as Nilsson's probabilistic logic [33], or even ad-hoc representations of function Pr from Definition 1. The following is an example of a BN over the running example.

Example 2 Consider the set $\mathbf{P}_{EM}$ from Figure 1. The Bayesian network depicted in Figure 2 describes the probability distribution Pr over all possible worlds $\mathcal{W}_{EM}$ shown in Figure 3.

So, for instance, the probability that false negatives do not arise in any of the two tests, and that the patient is at risk for both anxiety- and depression-related disorders (world λ_4) is 0.29808. ∎

2.2 Analytical Model

The DeLP3E formalism adopts a structured argumentation framework [36] for the AM. While the EM contains probabilistic information about the state of the world, the AM must allow for a different kind of information; in particular, it must be able to represent contradictory knowledge. This approach allows

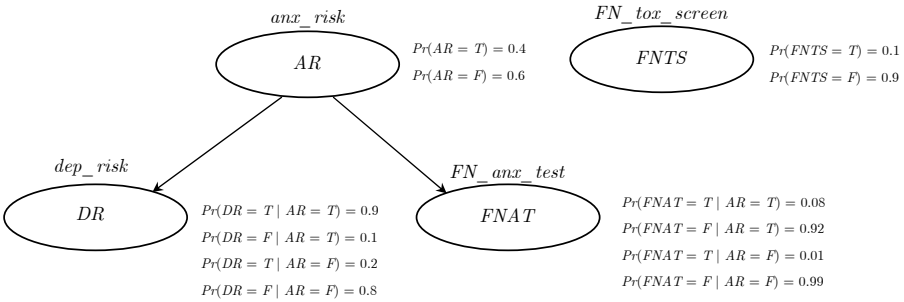

Fig. 2 Bayesian network used in the EM of the running example. The names of the random variables are simply the abbreviations of their corresponding atoms: $AR \mapsto$ *anx_risk*, $DR \mapsto$ *dep_risk*, $FNAT \mapsto$ *FN_anx_test*, and $FNTS \mapsto$ *FN_tox_screen*.

World	*anx_risk*	*dep_risk*	*FN_anx_test*	*FN_tox_screen*	Probability
λ_1	T	T	T	T	0.00288
λ_2	T	T	T	F	0.02592
λ_3	T	T	F	T	0.03312
λ_4	T	T	F	F	0.29808
λ_5	T	F	T	T	0.00032
λ_6	T	F	T	F	0.00288
λ_7	T	F	F	T	0.00368
λ_8	T	F	F	F	0.03312
λ_9	F	T	T	T	0.00048
λ_{10}	F	T	T	F	0.00432
λ_{11}	F	T	F	T	0.04752
λ_{12}	F	T	F	F	0.42768
λ_{13}	F	F	T	T	0.00012
λ_{14}	F	F	T	F	0.00108
λ_{15}	F	F	F	T	0.01188
λ_{16}	F	F	F	F	0.10692

Fig. 3 Probability distribution for the worlds in the running example.

for the creation of *arguments* that may compete with each other in order to reach a conclusion regarding a given query. This is known as a *dialectical process*, where arguments defeat each other based on a *comparison criterion*. Resulting from this process, certain arguments are *warranted*, while others are *defeated*. Argumentation-based reasoning has been proposed and studied in depth as a natural way to manage a set of inconsistent information – its strength lies in the fact that it closely resembles the way humans settle disputes (consider, for instance, how convictions are decided in trials). Another highly desirable characteristic of structured argumentation frameworks is that, once a conclusion is reached, the process also yields an explanation of *how we arrived* at it, as well as information about why a given argument is warranted. In the following, we first recall the basics of the underlying argumentation framework used, and then go on to introduce the analytical model (AM).

Defeasible Logic Programming with Presumptions (PreDeLP)

PreDeLP, first introduced in [32], is a formalism combining logic programming with defeasible argumentation. We will briefly recall the basics of PreDeLP, and refer the reader to [17,32] for the complete presentation.

The formalism contains several different constructs: facts, presumptions, strict rules, and defeasible rules. Facts are statements that always hold (such as a patient's symptom in our example), while presumptions are statements that may or may not be true (such as a medical diagnosis). Strict rules establish logical consequences (similar to material implication in first order logic, though the semantics is not exactly the same since the contrapositive does not follow from a strict rule). While strict rules, like facts, always hold, defeasible rules specify logical consequences that may be assumed to be true when no contradicting information is available. These components are used in the construction of *arguments*, and together comprise PreDeLP programs.

Formally, we use the notation $\Pi_{AM} = (\Theta, \Omega, \Phi, \Delta)$ to denote a PreDeLP program, where:

- Ω is the set of strict rules of the form $L_0 \leftarrow L_1, \ldots, L_n$, where L_0 is a ground literal and $\{L_i\}_{i>0}$ is a set of ground literals;
- Θ is the set of facts, written simply as atoms;
- Δ is the set of defeasible rules of the form $L_0 \prec L_1, \ldots, L_n$, where L_0 is a ground literal and $\{L_i\}_{i>0}$ is a set of ground literals, and
- Φ is the set of presumptions, which are written as defeasible without a body.

For simplicity, we will sometimes refer to Π_{AM} as a set corresponding to the union of its components.

Recall that all atoms in the AM must be formed with a predicate from the set $\mathbf{P}_{AM}$, and note that in both strict and defeasible rules, *strong negation* (i.e., classical negation as in first-order logic) is allowed in the head, and thus may be used to represent contradictory knowledge. The following is an example of a PreDeLP program over the running example.

Example 3 Consider again the medical diagnosis scenario from our running example; the DeLP3E program in Figure 4 encodes some basic knowledge that the attending physician might use to diagnose their patient. For instance, strict rule ω_1 states that based on a negative result when administering a test for toxins in the blood we can conclude that the patient is not misusing sleeping aids. On the other hand, defeasible rule δ_1 states that memory loss and depression can lead to such a misuse. In this example, the set of presumptions is empty. ∎

Arguments. Given a query in the form of a ground atom, the goal is to derive arguments for and against its validity – derivation follows the same mechanism of logic programming [30], and we denote such derivation with the symbol "$\vdash$". In the following, we say that such a derivation is "strict" if it only uses facts

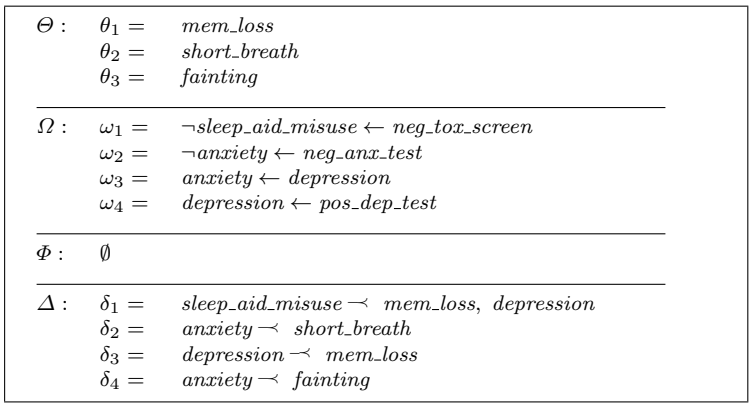

Fig. 4 A ground argumentation framework.

and strict rules; otherwise, we say that the derivation is "defeasible". Likewise, we say that a literal is strictly (defeasible) derived if the derivation is strict (defeasible). Finally, we say that an argument is "factual" if no presumptions are used in it.

Since rule heads can contain strong negation, it is possible to defeasibly derive contradictory literals from a program. For the treatment of contradictory knowledge, PreDeLP incorporates a defeasible argumentation formalism that allows the identification of the pieces of knowledge that are in conflict and, through the dialectical process discussed above, decides which information prevails as warranted. This dialectical process involves the construction and evaluation of arguments, formally defined next.

Definition 2 (Argument) An *argument* $\langle \mathcal{A}, L \rangle$ for a literal L is a pair of the literal and a (possibly empty) set of the AM ($\mathcal{A} \subseteq \Pi_{AM}$) that provides a minimal proof for L meeting the following requirements: (i) L is defeasibly derived from $\mathcal{A}$; (ii) $\Omega \cup \Theta \cup \mathcal{A}$ is not inconsistent; and (iii) $\mathcal{A}$ is a minimal subset of $\Delta \cup \Phi$ satisfying (i) and (ii), denoted $\langle \mathcal{A}, L \rangle$.

Literal L is called the *conclusion* supported by the argument, and $\mathcal{A}$ is the *support* of the argument. An argument $\langle \mathcal{B}, L \rangle$ is a *subargument* of $\langle \mathcal{A}, L' \rangle$ if $\mathcal{B} \subseteq \mathcal{A}$. An argument $\langle \mathcal{A}, L \rangle$ is *presumptive* if $\mathcal{A} \cap \Phi$ is not empty. We will also use $\Omega(\mathcal{A}) = \mathcal{A} \cap \Omega$, $\Theta(\mathcal{A}) = \mathcal{A} \cap \Theta$, $\Delta(\mathcal{A}) = \mathcal{A} \cap \Delta$, and $\Phi(\mathcal{A}) = \mathcal{A} \cap \Phi$.

Our definition differs slightly from that of [42], where DeLP is introduced, as we include strict rules and facts as part of arguments. We make this change because in our framework the strict rules and facts used to construct a given argument *may only be true in certain worlds in the EM*. Hence, the entire set of facts and strict rules need not be consistent in our framework. We shall discuss how portions of the AM are assigned EM worlds in the next section.

Example 4 Figure 5 shows example arguments based on the PreDeLP program from Figure 4. Argument $\mathcal{A}_3$ uses an additional component not present in the

$\langle \mathcal{A}_1, anxiety \rangle$	$\mathcal{A}_1 = \{\theta_1, \delta_3, \omega_3\}$
$\langle \mathcal{A}_2, anxiety \rangle$	$\mathcal{A}_2 = \{\theta_3, \delta_4\}$
$\langle \mathcal{A}_3, \neg sleep\_aid\_misuse \rangle$	$\mathcal{A}_3 = \{\omega_1\} \cup \{neg\_tox\_screen \prec\}$

Fig. 5 Example arguments based on the running example scenario.

original program, and states that if we can assume a negative result for a tox screen, we can conclude that the patient is not misusing sleeping aids. ∎

Given an argument $\langle \mathcal{A}_1, L_1 \rangle$, counter-arguments are arguments that contradict it. Argument $\langle \mathcal{A}_2, L_2 \rangle$ is said to *counterargue* or *attack* $\langle \mathcal{A}_1, L_1 \rangle$ at a literal L' iff there exists a subargument $\langle \mathcal{A}, L'' \rangle$ of $\langle \mathcal{A}_1, L_1 \rangle$ such that the set $\Omega(\mathcal{A}_1) \cup \Omega(\mathcal{A}_2) \cup \Theta(\mathcal{A}_1) \cup \Theta(\mathcal{A}_2) \cup \{L_2, L''\}$ is inconsistent. A *proper defeater* of an argument $\langle A, L \rangle$ is a counter-argument that – by some criterion – is considered to be better than $\langle A, L \rangle$; if the two are incomparable according to this criterion, the counterargument is said to be a *blocking* defeater. An important characteristic of PreDeLP is that the argument comparison criterion is modular, and thus the most appropriate criterion for the domain that is being represented can be selected; the default criterion used in classical defeasible logic programming (from which PreDeLP is derived) is *generalized specificity* [44], though an extension of this criterion is required for arguments using presumptions [32]. We briefly recall this criterion next – the first definition is for generalized specificity, which is subsequently used in the definition of presumption-enabled specificity.

Definition 3 (DeLP Argument Preference) Let $\Pi_{AM} = (\Theta, \Omega, \Phi, \Delta)$ be a PreDeLP program and let $\mathcal{F}$ be the set of all literals that have a defeasible derivation from Π_{AM}. An argument $\langle \mathcal{A}_1, L_1 \rangle$ is *preferred to* $\langle \mathcal{A}_2, L_2 \rangle$, denoted with $\mathcal{A}_1 \succ_{PS} \mathcal{A}_2$ if:

(1) For all $H \subseteq \mathcal{F}$, $\Omega(\mathcal{A}_1) \cup \Omega(\mathcal{A}_2) \cup H$ is consistent: if there is a derivation for L_1 from $\Omega(\mathcal{A}_2) \cup \Omega(\mathcal{A}_1) \cup \Delta(\mathcal{A}_1) \cup H$, and there is no derivation for L_1 from $\Omega(\mathcal{A}_1) \cup \Omega(\mathcal{A}_2) \cup H$, then there is a derivation for L_2 from $\Omega(\mathcal{A}_1) \cup \Omega(\mathcal{A}_2) \cup \Delta(\mathcal{A}_2) \cup H$; and

(2) there is at least one set $H' \subseteq \mathcal{F}$, $\Omega(\mathcal{A}_1) \cup \Omega(\mathcal{A}_2) \cup H'$ is consistent, such that there is a derivation for L_2 from $\Omega(\mathcal{A}_1) \cup \Omega(\mathcal{A}_2) \cup H' \cup \Delta(\mathcal{A}_2)$, there is no derivation for L_2 from $\Omega(\mathcal{A}_1) \cup \Omega(\mathcal{A}_2) \cup H'$, and there is no derivation for L_1 from $\Omega(\mathcal{A}_1) \cup \Omega(\mathcal{A}_2) \cup H' \cup \Delta(\mathcal{A}_1)$.

Intuitively, the principle of specificity says that, in the presence of two conflicting lines of argument about a proposition, the one that uses more of the available information is more convincing. The following extension for presumptive arguments was first introduced in [32].

Definition 4 (Presumptive Argument Preference) Given PreDeLP program $\Pi_{AM} = (\Theta, \Omega, \Phi, \Delta)$, an argument $\langle \mathcal{A}_1, L_1 \rangle$ is *preferred to* $\langle \mathcal{A}_2, L_2 \rangle$, denoted with $\mathcal{A}_1 \succ \mathcal{A}_2$ if any of the following conditions hold:

(1) $\langle \mathcal{A}_1, L_1 \rangle$ and $\langle \mathcal{A}_2, L_2 \rangle$ are both factual and $\langle \mathcal{A}_1, L_1 \rangle \succ_{PS} \langle \mathcal{A}_2, L_2 \rangle$.

(2) $\langle \mathcal{A}_1, L_1 \rangle$ is a factual argument and $\langle \mathcal{A}_2, L_2 \rangle$ is a presumptive argument.

(3) $\langle \mathcal{A}_1, L_1 \rangle$ and $\langle \mathcal{A}_2, L_2 \rangle$ are presumptive arguments, and

(a) $\Phi(\mathcal{A}_1) \subsetneq \Phi(\mathcal{A}_2)$ or,

(b) $\Phi(\mathcal{A}_1) = \Phi(\mathcal{A}_2)$ and $\langle \mathcal{A}_1, L_1 \rangle \succ_{PS} \langle \mathcal{A}_2, L_2 \rangle$.

Generally, if $\mathcal{A}, \mathcal{B}$ are arguments with rules X and Y, resp., and $X \subset Y$, then $\mathcal{A}$ is stronger than $\mathcal{B}$. This also holds when $\mathcal{A}$ and $\mathcal{B}$ use presumptions P_1 and P_2, resp., and $P_1 \subset P_2$.

Note: The specificity criterions used here are *not transitive* [48], and therefore should not be assumed to define an ordering over arguments. This, however, does not pose a problem for our framework, as the criterion is only ever used to compare pairs of arguments to see which one "wins out". We remind the reader that the comparison criterion in our framework is modular; if transitivity is required, the one proposed in [48] is an option.

A sequence of arguments called an *argumentation line* thus arises from this attack relation, where each argument defeats its predecessor. To avoid undesirable sequences, which may represent circular argumentation lines, in DeLP an *argumentation line* is *acceptable* if it satisfies certain constraints (see [17]). A literal L is *warranted* if there exists a non-defeated argument $\mathcal{A}$ supporting L.

Clearly, there can be more than one defeater for a particular argument $\langle \mathcal{A}, L \rangle$. Therefore, many acceptable argumentation lines could arise from $\langle \mathcal{A}, L \rangle$, leading to a tree structure. The tree is built from the set of all argumentation lines rooted in the initial argument. In a dialectical tree, every node (except the root) represents a defeater of its parent, and leaves correspond to undefeated arguments. Each path from the root to a leaf corresponds to a different acceptable argumentation line. A dialectical tree provides a structure for considering all the possible (maximal) acceptable argumentation lines that can be generated for deciding whether an argument is defeated. We call this tree *dialectical* because it represents an exhaustive dialectical (in the sense of providing reasons for and against a position) analysis for the argument in its root. For a given argument $\langle \mathcal{A}, L \rangle$, we denote the corresponding dialectical tree as $\mathcal{T}(\langle \mathcal{A}, L \rangle)$.

Given a literal L and an argument $\langle \mathcal{A}, L \rangle$, in order to decide whether or not a literal L is warranted, every node in the dialectical tree $\mathcal{T}(\langle \mathcal{A}, L \rangle)$ is recursively marked as "D" (*defeated*) or "U" (*undefeated*), obtaining a marked dialectical tree $\mathcal{T}^*(\langle \mathcal{A}, L \rangle)$ as follows:

1. All leaves in $\mathcal{T}^*(\langle \mathcal{A}, L \rangle)$ are marked as "U"s, and
2. Let $\langle \mathcal{B}, L_q \rangle$ be an inner node of $\mathcal{T}^*(\langle \mathcal{A}, L \rangle)$. Then $\langle \mathcal{B}, L_q \rangle$ will be marked as "U" iff every child of $\langle \mathcal{B}, L_q \rangle$ is marked as "D". The node $\langle \mathcal{B}, L_q \rangle$ will be marked as "D" iff it has at least a child marked as "U".

Given an argument $\langle \mathcal{A}, L \rangle$ obtained from Π_{AM}, if the root of $\mathcal{T}^*(\langle \mathcal{A}, L \rangle)$ is marked as "U", then we will say that $\mathcal{T}^*(\langle \mathcal{A}, L \rangle)$ *warrants* L and that L is *warranted* from Π_{AM}. (Warranted arguments correspond to those in the

grounded extension of a Dung argumentation system [11].) There is a further requirement when the arguments in the dialectical tree contains presumptions – the conjunction of all presumptions used in even (respectively, odd) levels of the tree must be consistent. This can give rise to multiple trees for a given literal, as there can potentially be different arguments that make contradictory assumptions.

We can then extend the idea of a dialectical tree to a *dialectical forest*. For a given literal L, a dialectical forest $\mathcal{F}(L)$ consists of the set of dialectical trees for all arguments for L. We shall denote a marked dialectical forest, the set of all marked dialectical trees for arguments for L, as $\mathcal{F}^*(L)$. Hence, for a literal L, we say it is *warranted* if there is at least one argument for that literal in the dialectical forest $\mathcal{F}^*(L)$ that is labeled as "U", *not warranted* if there is at least one argument for the literal $\neg L$ in the dialectical forest $\mathcal{F}^*(\neg L)$ that is labeled as "U", and *undecided* otherwise. We shall refer to whether literal L is warranted, not warranted, or undecided as L's "warrant status", and sometimes refer to the "warranted" status as "Yes" and the "not warranted" status as "No".

2.3 The DeLP3E Framework

Our framework, originally proposed in [39], is the result of combining the environmental and analytical models (which we denote with Π_{EM} and Π_{AM}, respectively). Intuitively, given Π_{AM}, every element of $\Omega \cup \Theta \cup \Delta \cup \Phi$ only hold in certain worlds in the set $\mathcal{W}_{EM}$ – i.e., these elements are subject to probabilistic events. Each element of $\Omega \cup \Theta \cup \Delta \cup \Phi$ is thus associated with a formula over $\mathbf{G}_{EM}$ (using conjunction, disjunction, and negation, as usual) – we use $form_{EM}$ to denote the set of all such formulas. The notion of an *annotation function* associates elements of $\Omega \cup \Theta \cup \Delta \cup \Phi$ with elements in $form_{EM}$.

Definition 5 (Annotation Function [39]) An *annotation function* is any function of the form $af : \Omega \cup \Theta \cup \Delta \cup \Phi \rightarrow form_{EM}$. We use $[af]$ to denote the set of all annotation functions.

We will sometimes denote annotation functions as sets of pairs $(f, af(f))$ in order to simplify the presentation. Figure 6 shows an example of an annotation function for our running example; for instance, the annotation for rule δ_2 means that this rule only holds whenever the probabilistic event *anx_risk* is true. If annotations are "True", this means that they hold in all possible worlds.

Definition 6 (DeLP3E Program) Given environmental model Π_{EM}, analytical model Π_{AM}, and annotation function af, a *DeLP3E program* is of the form $\mathcal{I} = (\Pi_{EM}, \Pi_{AM}, af)$. We use notation $[\mathcal{I}]$ to denote the set of all possible programs.

In the following, given DeLP3E program $\mathcal{I} = (\Pi_{EM}, \Pi_{AM}, af)$ and $\lambda \in \mathcal{W}_{EM}$, we use notation $\Pi_{AM}(\lambda) = \{f \in \Pi_{AM} \text{ s.t. } \lambda \models af(f)\}$. This gives rise to a *decomposed view* of DeLP3E programs, as illustrated next.

$af(\theta_1) =$	True
$af(\theta_2) =$	True
$af(\theta_3) =$	True
$af(\omega_1) =$	True
$af(\omega_2) =$	True
$af(\omega_3) =$	True
$af(\omega_4) =$	True
$af(\delta_1) =$	True
$af(\delta_2) =$	*anx_risk*
$af(\delta_3) =$	*dep_risk*
$af(\delta_4) =$	*anx_risk*

Fig. 6 An example of an annotation function over the running example.

λ_1 : $\{\theta_1, \theta_2, \theta_3, \omega_1, \omega_2, \omega_3, \omega_4, \delta_1, \delta_2, \delta_3, \delta_4\}$	λ_2 : $\{\theta_1, \theta_2, \theta_3, \omega_1, \omega_2, \omega_3, \omega_4, \delta_1, \delta_2, \delta_3, \delta_4\}$	λ_3 : $\{\theta_1, \theta_2, \theta_3, \omega_1, \omega_2, \omega_3, \omega_4, \delta_1, \delta_2, \delta_3, \delta_4\}$	λ_4 : $\{\theta_1, \theta_2, \theta_3, \omega_1, \omega_2, \omega_3, \omega_4, \delta_1, \delta_2, \delta_3, \delta_4\}$
λ_5 : $\{\theta_1, \theta_2, \theta_3, \omega_1, \omega_2, \omega_3, \omega_4, \delta_1, \delta_2, \delta_4\}$	λ_6 : $\{\theta_1, \theta_2, \theta_3, \omega_1, \omega_2, \omega_3, \omega_4, \delta_1, \delta_2, \delta_4\}$	λ_7 : $\{\theta_1, \theta_2, \theta_3, \omega_1, \omega_2, \omega_3, \omega_4, \delta_1, \delta_2, \delta_4\}$	λ_8 : $\{\theta_1, \theta_2, \theta_3, \omega_1, \omega_2, \omega_3, \omega_4, \delta_1, \delta_2, \delta_4\}$
λ_9 : $\{\theta_1, \theta_2, \theta_3, \omega_1, \omega_2, \omega_3, \omega_4, \delta_1, \delta_3\}$	λ_{10} : $\{\theta_1, \theta_2, \theta_3, \omega_1, \omega_2, \omega_3, \omega_4, \delta_1, \delta_3\}$	λ_{11} : $\{\theta_1, \theta_2, \theta_3, \omega_1, \omega_2, \omega_3, \omega_4, \delta_1, \delta_3\}$	λ_{12} : $\{\theta_1, \theta_2, \theta_3, \omega_1, \omega_2, \omega_3, \omega_4, \delta_1, \delta_3\}$
λ_{13} : $\{\theta_1, \theta_2, \theta_3, \omega_1, \omega_2, \omega_3, \omega_4, \delta_1\}$	λ_{14} : $\{\theta_1, \theta_2, \theta_3, \omega_1, \omega_2, \omega_3, \omega_4, \delta_1\}$	λ_{15} : $\{\theta_1, \theta_2, \theta_3, \omega_1, \omega_2, \omega_3, \omega_4, \delta_1\}$	λ_{16} : $\{\theta_1, \theta_2, \theta_3, \omega_1, \omega_2, \omega_3, \omega_4, \delta_1\}$

Fig. 7 A depiction of how the DeLP3E program in the running example can be decomposed into one classical PreDeLP program for each possible EM world (cf. Figure 3 for the definition of worlds λ_1–λ_{16} in terms of the random variables in the EM).

Example 5 Consider the different examples presented so far: the EM from Example 2 (with the worlds from Figure 3), Π_{AM} from Figure 4, the arguments in Figure 5, and the annotation function from Figure 6 – these components give rise to a DeLP3E program $\mathcal{I} = (\Pi_{EM}, \Pi_{AM}, af)$ Figure 7 shows how $\mathcal{I}$ can be decomposed into one classical PreDeLP program $\Pi_{AM}(\lambda)$ for each world $\lambda \in \mathcal{W}_{EM}$.

For instance, $\Pi_{AM}(\lambda_7)$ contains θ_1, θ_2, θ_3, ω_1, ω_2, ω_3, ω_4, and δ_1 because the annotation function associates condition True to all of these components; it contains δ_2 and δ_4 because condition *anx_risk* is true in λ_7, and it does not contain δ_3 because condition *dep_risk* is false in λ_7. ∎

The most direct way of considering consequences of DeLP3E programs is thus to consider what happens in each world in $\mathcal{W}_{EM}$; that is, the defeat relationship among arguments depends on the current state of the (EM) world.

Definition 7 (Existence of an Argument in a World) Given DeLP3E program $\mathcal{I} = (\Pi_{EM}, \Pi_{AM}, af)$, argument $\langle \mathcal{A}, L \rangle$ is said to *exist in world* $\lambda \in \mathcal{W}_{EM}$ if $\forall c \in \mathcal{A}, \lambda \models af(c)$.

The notion of existence is extended to argumentation lines, dialectical trees, and dialectical forests in the expected way (for instance, an argumentation line exists in λ iff all arguments that comprise that line exist in λ).

Example 6 Consider the different examples presented so far: the worlds in Figure 3, Π_{AM} from Figure 4, the arguments in Figure 5, and the annotation function from Figure 6.

Since argument $\mathcal{A}_1$ uses defeasible rule δ_3, and $af(\delta_3) = dep\_risk$ (while the other two components have annotation "True"), we can conclude that this argument exists in worlds in which *dep_risk* is true, *i.e.*,, λ_1–λ_4 and λ_9–λ_{12}. ■

The idea of a dialectical tree is also extended w.r.t. worlds; so, for a given world $\lambda \in \mathcal{W}_{EM}$, the dialectical (resp., marked dialectical) tree induced by λ is denoted with $\mathcal{T}_\lambda\langle\mathcal{A}, L\rangle$ (resp., $\mathcal{T}^*_\lambda\langle\mathcal{A}, L\rangle$). We require that all arguments and defeaters in these trees exist in λ. Likewise, we extend the notion of dialectical forests in the same manner (denoted with $\mathcal{F}_\lambda(L)$ and $\mathcal{F}^*_\lambda(L)$, respectively). Based on these concepts, we introduce the notion of *warranting scenario*.

Definition 8 (Warranting Scenario) Given DeLP3E program $\mathcal{I} = (\Pi_{EM}, \Pi_{AM}, af)$ and literal L formed with a ground atom from $\mathbf{G}_{AM}$, a world $\lambda \in \mathcal{W}_{EM}$ is said to be a *warranting scenario* for L (denoted $\lambda \vdash_{\mathsf{war}} L$) if there is a dialectical forest $\mathcal{F}^*_\lambda(L)$ in which L is warranted and $\mathcal{F}^*_\lambda(L)$ exists in λ.

The idea of a warranting scenario is used to formally define DeLP3E entailment. The set of worlds in the EM where a literal L in the AM *must* be true is exactly the set of warranting scenarios — these are the "necessary" worlds: $nec(L) = \{\lambda \in \mathcal{W}_{EM} \mid (\lambda \vdash_{\mathsf{war}} L)\}$. Now, the set of worlds in the EM where AM literal L *can* be true is the following — these are the "possible" worlds: $poss(L) = \{\lambda \in \mathcal{W}_{EM} \mid \lambda \nvdash_{\mathsf{war}} \neg L\}$.

In the following we use notation $for(\lambda) = \bigwedge_{a\in\lambda} a \wedge \bigwedge_{a\notin\lambda} \neg a$, which denotes the formula that has λ as its only model. We now extend this notation to sets of worlds: $for(W) = \bigvee_{\lambda\in W} for(\lambda)$. Entailment can then be defined as follows:

Definition 9 (DeLP3E Entailment) Given DeLP3E program $\mathcal{I} = (\Pi_{EM}, \Pi_{AM}, af)$, AM literal L and probability interval $p \in [\ell, u]$, we say that $\mathcal{I}$ entails L with probability $p \in [\ell, u]$ if the probability distribution Pr yielded by Π_{EM} is such that $\ell \leq \sum_{\lambda\in nec(L)} \mathsf{Pr}(\lambda)$ and $\sum_{\lambda\in poss(L)\lambda} \mathsf{Pr}(\lambda) \leq u$.

2.4 Consistency of DeLP3E Programs

Finally, one of the central topics of this paper is that of inconsistencies, which can arise in our framework in more than one way [39]. In this paper, we assume that the probabilistic model is consistent, and focus on inconsistencies that arise in the AM. Towards this end, we use the following notion of (classical) consistency: PreDeLP program Π is said to be *consistent* if there does not exist a ground literal a s.t. $\Pi \vdash a$ and $\Pi \vdash \neg a$. For DeLP3E programs, the interaction between the annotation function and facts and strict rules may cause conflicts, as defined next.

Definition 10 (Consistency of DeLP3E Programs) A DeLP3E program $\mathcal{I} = (\Pi_{EM}, \Pi_{AM}, af)$, with $\Pi_{AM} = \langle \Theta, \Omega, \Phi, \Delta \rangle$, is *consistent* if: given probability distribution Pr for Π_{EM}, if there exists a world $\lambda \in \mathcal{W}_{EM}$ such that $\bigcup_{x \in \Theta \cup \Omega \mid \lambda \models af(x)} \{x\}$ is inconsistent, then we have $Pr(\lambda) = 0$.

The intuition behind this definition is that subsets of facts and strict rules hold under the circumstances specified by the annotation function – such circumstances can be expressed as sets of EM worlds. Now, if there exists a world where (at least) two contradictory strict statements are true, then the EM must assign probability zero to this world.

Example 7 Let us return to the running example; consider Π_{AM} from Figure 4, Π_{EM} from Figure 2, and the annotation function from Figure 6, with the addition of fact $\theta_4 = pos\_dep\_test$ with $af(\theta_4) = \mathsf{True}$ and fact $\theta_5 = neg\_anx\_test$ with $af(\theta_5) = \neg FN\text{-}anx\_test$. It is now clear that the program is inconsistent, since there exists world λ_3 (among several others) such that $\bigcup_{x \in \Theta \cup \Omega \mid \lambda_3 \models af(x)} \{x\}$ warrants both *anxiety* (via argument with θ_4 and ω_3) and $\neg$*anxiety* (via argument with θ_5 and ω_2). ∎

3 Revision of DeLP3E Programs based on Quantitative Aspects

We have finally arrived at the main problem we address in this paper – revising knowledge bases. This problem can be generically stated as: given DeLP3E program $\mathcal{I} = (\Pi_{EM}, \Pi_{AM}, af)$, with $\Pi_{AM} = \Omega \cup \Theta \cup \Delta \cup \Phi$ and a pair (f, af') where f is either an atom or a rule and af' is equivalent to af, except for its expansion to include f[3], obtain a new program $\mathcal{I}'$ called the *revised* knowledge base that addresses the incorporation of the *epistemic input* (f, af') into the original program; we denote this operation with the symbol "$\bullet$" – *i.e.*,, $\mathcal{I}' = \mathcal{I} \bullet (f, af')$.

Now, the problem statement as presented above is quite vague, since we did not give any details as to how the operator "addresses the incorporation" of the epistemic input. There are many approaches in the literature (cf. Section 6) that address this problem quite differently; one of the main properties that characterize revision operators is whether or not they satisfy the *Success* property, which states that the epistemic input must be a consequence of the revised knowledge base. Here, we will adopt a cautious stance and assume that this property does not hold in general; therefore, we focus on so-called *non-prioritized* revision operators.

The basic issue that revision operators must deal with is inconsistency (we will discuss this in more depth shortly); as we saw in Section 2.4, inconsistency in DeLP3E programs involves worlds that have non-zero probability and an associated PreDeLP program that is inconsistent. In our previous work [39, 41] we identified three basic approaches that can be taken towards solving this problem:

[3] That is, $af'(x) = af(x)$ for all $x \in dom(af)$, and $dom(af') = dom(af) \cup \{f\}$.

- *Modifying the EM:* Perhaps the simplest approach is to fix the problem of inconsistency by forcing the probabilistic model to assign probability zero to all worlds that cause inconsistencies to arise.
- *Modifying the AM:* Alternatively, for each world in which the corresponding PreDeLP program is inconsistent, we can modify this program to remove the problem.
- *Modifying the annotation function.* Finally, as a finer-grained approach compared to the previous one, we can alter the annotation function.

In the following, we will assume that epistemic inputs involve only strict components (facts or rules), since defeasible components can always be added without inconsistencies arising. Regarding these three possible approaches, in this paper we will focus on the third one since it is a generalization of the second – if we only allow removing elements from the AM, such an operation will have the same effect as not removing the element but modifying the annotation function so that it associates the formula "$\bot$" to it. The generalization of annotation function-based revision lies in that there is not always the need for such a drastic measure; see [41] for further discussions on this, including examples. Furthermore, operations of the first kind alone do not suffice to perform revisions, as can be seen in the following simple example.

Example 8 Consider the following DeLP3E program, where the EM consists of two worlds $\{a\}$ and $\{\neg a\}$, each with probability 0.5:

$$\begin{array}{lll} \omega_1: & p \leftarrow q & af(\omega_1) = a \\ \theta_1: & \neg p & af(\theta_1) = a \\ \omega_2: & \neg p \leftarrow q & af(\omega_2) = \neg a \\ \theta_2: & p & af(\theta_2) = \neg a \end{array}$$

Now, suppose we wish to revise by formula $\theta_3 : q$ with $af(\theta_3) = \mathsf{True}$. Since both EM worlds are inconsistent with the formula, it is impossible to change the allocation of the probability mass in order to avoid inconsistencies; therefore, the only option is to reject the input. ∎

3.1 A Recap of Annotation Function-based Belief Revision

Since the quantitative approach that we propose in this paper is related to the **AFO** class of revision operators introduced in [39], in this section we provide a brief summary of the relevant material. After recalling the relevant postulates, we present the construction of **AFO** operators.

3.1.1 Rationality Postulates

The following properties characterize the behavior of operators for revising annotation functions; we briefly discuss them here in an informal manner, and refer the reader to [41] for their formal presentation.

- **Inclusion:** For any given element g in the AM, the worlds that satisfy g's annotation after the revision are a subset of the set of worlds satisfying g's annotation before the revision. That is, the constraints in the revised annotations can only become more restrictive.
- **Vacuity:** If simply adding the input to the program does not lead to inconsistencies, then the operator does precisely that.
- **Consistency Preservation:** If the original program is consistent, then so is the revised program.
- **Weak Success:** As in Vacuity, if adding the input to the program does not cause inconsistencies, then the input must be present "as is" in the revised program.
- **Relevance:** Given a specific EM world, if a part of its associated AM knowledge base is removed by the operator, then there exists a superset of the remaining knowledge base that is not consistent with the removed element and the input. That is, removed elements are always "relevant" to an inconsistency.
- **AF Uniformity:** If two different inputs are such that the same set of EM worlds lead to inconsistencies in a given AM knowledge base, and it is the case that analogous subsets taken in conjunction with their respective input lead to equivalent consistency/inconsistency, then the models removed from the annotations elements in the AM knowledge base are the same for both inputs. In other words, the operator must behave in the same way when presented with inputs that have an equivalent effect on the knowledge base.

We now continue briefly recalling the presentation of the annotation function-based operator of [39] by discussing its construction. In order to do so, we adopt the following notation: the program to revise is denoted with $\mathcal{I} = (\Pi_{EM}, \Pi_{AM}, af)$, with $\Pi_{AM} = \Omega \cup \Theta \cup \Delta \cup \Phi$, and the epistemic input is denoted with (f, af'), where f is either an atom or a rule and af' is equivalent to af, except for its expansion to include f. We denote the revision as follows: $\mathcal{I} \bullet (f, af') = (\Pi_{EM}, \Pi_{AM}, af'')$ where af'' is the revised annotation function. We will slightly abuse notation in order to make the presentation clearer, and use notation to convert sets of worlds to and from formulas. Moreover, we often refer to "removing elements of Π_{AM}" to refer to changes to the annotation function that cause certain elements of the Π_{AM} to not have their annotations satisfied in certain EM worlds. As we are looking to change the annotation function for a specific subset of facts and strict rules, we specify these subsets with the following notation:

- $\mathcal{I} \cup (f, af')$ to denote $\mathcal{I}' = (\Pi_{EM}, \Pi_{AM} \cup \{f\}, af')$.
- $(f, af') \in \mathcal{I} = (\Pi_{AM}, \Pi_{EM}, af)$ to denote $f \in \Pi_{AM}$ and $af = af'$.
- $\mathcal{W}^0_{EM}(\mathcal{I}) = \{\lambda \in \mathcal{W}_{EM} \mid \Pi^{\mathcal{I}}_{AM}(\lambda) \text{ is inconsistent}\}$ – this set contains all the EM worlds for a given program where the corresponding knowledge base in the AM is classically inconsistent.

- $\mathcal{W}^I_{EM}(\mathcal{I}) = \{\lambda \in \mathcal{W}^0_{EM} \mid Pr(\lambda) > 0\}$ – this is a subset of $\mathcal{W}^0_{EM}(\mathcal{I})$ containing worlds that are assigned a non-zero probability; i.e., the worlds where inconsistency in the AM arises.
- $wld(f) = \{\lambda \mid \lambda \models f\}$ – the set of worlds that satisfy formula f; and
- $for(\lambda) = \bigwedge_{a \in \lambda} a \wedge \bigwedge_{a \notin \lambda} \neg a$ – the formula that has λ as its only model.
- $\Pi^{\mathcal{I}}_{AM}(\lambda) = \{f \in \Theta \cup \Omega \mid \lambda \models af(f)\}$ – the subset of facts and strict rules in Π_{AM} whose annotations are true in EM world λ.

We will make use of this notation in the next section.

3.1.2 Operator Construction

The basis of the construction of the class of so-called annotation function-based operators is that any subset of Π_{AM} that is associated with a world in $\mathcal{W}^I_{EM}(\mathcal{I} \cup (f, af'))$ must be modified so that consistency is ensured. For each such world λ, we make use of the following set of "candidate replacement programs" for $\Pi_{AM}(\lambda)$:

$$\begin{aligned} CandPgm_{af}(\lambda, \mathcal{I}) = \{&\Pi'_{AM} \mid \Pi'_{AM} \subseteq \Pi_{AM}(\lambda) \text{ s.t. } \Pi'_{AM} \text{ is consistent and} \\ &\nexists \Pi''_{AM} \subseteq \Pi_{AM}(\lambda) \textit{ s.t. } \Pi''_{AM} \supset \Pi'_{AM} \text{ s.t. } \Pi''_{AM} \\ &\text{is consistent}\} \end{aligned}$$

The intuition is that each maximal consistent subset of $\Pi^{\mathcal{I}}_{AM}(\lambda)$ is a candidate for replacing the inconsistent program for that world. To specify the construction, we need to introduce some more notation. Let $\Gamma : \mathcal{W}_{EM} \rightarrow 2^{[\Theta] \cup [\Omega]}$; i.e., a function from EM worlds to subsets of the strict components of the AM – this function will be used to choose the revised AM for each world. For each component h in the AM there is a formula in $\Pi_{AM} \cup \{f\}$, where f is part of the epistemic input (i.e., what the operator is revising by). Given these elements, we define:

$$newFor(h, \Gamma, \mathcal{I}, (f, af')) = af'(h) \wedge \bigwedge_{\lambda \in \mathcal{W}^I_{EM}(\mathcal{I} \cup (f, af')) \mid h \notin \Gamma(\lambda)} \neg for(\lambda_i)$$

Intuitively, *newFor* provides a new annotation for each component $h \in \Pi_{AM}$; such formula can be seen as the result of selecting a maximally consistent subset of $\Pi_{AM}(\lambda)$ for each EM world λ. We are finally able to introduce the **AFO** class of operators:

Definition 11 (AF-based Operators [39, 41]) A belief revision operator $\bullet$ is an "annotation function-based" (or af-based) operator ($\bullet \in$ **AFO**) if given program $\mathcal{I} = (\Pi_{EM}, \Pi_{AM}, af)$ and input (f, af'), the revision is defined as $\mathcal{I} \bullet (f, af') = (\Pi_{EM}, \Pi_{AM} \cup \{f\}, af'')$, where:

$$\forall h, af''(h) = newFor(h, \Gamma, \mathcal{I}, (f, af'))$$

where $\forall \lambda \in \mathcal{W}_{EM}$, $\Gamma(\lambda) \in CandPgm_{af}(\lambda, \mathcal{I} \cup (f, af'))$.

		Analytical Model	Annotation Function
Θ :	$\theta_1 =$	*mem_loss*	True
	$\theta_2 =$	*short_breath*	True
	$\theta_3 =$	*fainting*	True
	$\theta_4 =$	*neg_anx_test*	$\{\neg$*FN-anx_test*$\}$
	$\theta_5 =$	*neg_tox_screen*	$\{\neg$*FN-tox_screen*$\}$
Ω :	$\omega_1 =$	$\neg$*sleep_aid_misuse* $\leftarrow$ *neg_tox_screen*	True
	$\omega_2 =$	$\neg$*anxiety* $\leftarrow$ *neg_anx_test*	True
	$\omega_3 =$	*anxiety* $\leftarrow$ *depression*	True
	$\omega_4 =$	*depression* $\leftarrow$ *pos_dep_test*	True
Φ :	$\emptyset$		
Δ :	$\delta_1 =$	*sleep_aid_misuse* $\prec$ *mem_loss*, *depression*	True
	$\delta_2 =$	*anxiety* $\prec$ *short_breath*	$\{$*anx_risk*$\}$
	$\delta_3 =$	*depression* $\prec$ *mem_loss*	$\{$*dep_risk*$\}$
	$\delta_4 =$	*anxiety* $\prec$ *fainting*	$\{$*anx_risk*$\}$

Fig. 8 The DeLP3E program from the running example, after adding facts θ_4 and θ_5. The annotation function is provided in a separate column for convenience.

In [39,41], we provide a representation theorem that states the equivalence between this construction and the operators discussed above. We refer the reader to these articles for a more complete and detailed presentation.

3.2 Towards a Quantitative Approach

Traditionally, belief revision has been addressed from a qualitative point of view rather than a quantitative one (cf. Section 6 for a discussion on related work). A simple example of this is the fact that, faced with the option of removing either both atoms a and b or only atom c, classical revision operators typically declare both options to be just as good, since neither is a subset of the other; it could be argued, then, that taking quantitative aspects into account (such as the number of elements removed) *may* lead to a better solution – of course, this may depend on other factors. As we will see, there are different ways in which such quantitative aspects can be incorporated into revision operations. For instance, in our setting, DeLP3E programs can be regarded in a world-by-world manner, and changes made in one world can be compared to those made in another. The **AFO** operators described in Section 3.1 make decisions for each world independently; we now wish to address the issue of taking into account different kinds of quantitative aspects when revising DeLP3E programs. The following example motivates our approach in our medical diagnosis scenario.

Example 9 Consider again the running example, and suppose the physician has decided to carry out as a first step two tests, a toxin screen and an anxiety test, and that both tests yielded negative results. Note that the validity of

these tests is subject to probabilistic events (in this case, false negatives). The new program is reproduced in Figure 8.

Figure 9 shows the world-by-world decomposition of the new program, and the atoms that are warranted in each case. From the information in this figure, we can compute the following probabilities for the hypotheses that the physician is contemplating (depression, anxiety, and misuse of sleeping aids):

Literal	**Probability**
depression	: 0.06672
sleep_aid_misuse	: 0.05088
$\neg$*sleep_aid_misuse*	: 0.93324
anxiety	: 0.06992
$\neg$*anxiety*	: 0.92888

Since they all have low probabilities after performing the tests, the doctor decides to test for depression and in this case receives a positive result (atom *pos_dep_test*). For the sake of this example, we will assume that the validity of the outcome of this test (unlike the other two) is not subject to probabilistic events – thus, we have $af(pos\_dep\_test) = \mathsf{True}$.

Now, while for the first two tests we were able to simply add the corresponding atoms and extend the annotation function accordingly, simply adding $\theta_6 = pos\_dep\_test$ with $af(\theta_6) = \mathsf{True}$ causes inconsistencies to arise in eight of the possible worlds (λ_3, λ_4, λ_7, λ_8, λ_{11}, λ_{12}, λ_{15}, and λ_{16}). Essentially, the problems arise because the negative anxiety test allows us to conclude that there is no anxiety, while the positive depression test would allow us to conclude that indeed there is anxiety. Since both derivations only involve strict components, this leads to an inconsistent AM. ∎

Example 9 shows an interesting case of belief revision in DeLP3E programs; when presented with new information that is in conflict with existing one, we must find a way to address its incorporation into the existing knowledge – non-prioritized operators are very flexible, since they always have the option of ignoring the new information. However, this flexibility also means that – in the case of DeLP3E programs – there is no guidance with respect to how revisions should be carried out *globally*, since each world is treated as a separate revision problem. Next, we discuss two kinds of functions that will prove to be useful in addressing this situation.

3.2.1 Two Building Blocks

We now introduce warrant probability functions and revision objective functions, which are later used in the definition of our new class of non-prioritized belief revision operators.

Warrant Probability Functions. As one of the building blocks to our quantitative approach, given a DeLP3E program $\mathcal{I}$ we define *warrant probability functions* (WPFs, for short).

World	Probability	Warranted Literals
λ_1	0.00288	$\{\theta_1, \theta_2, \theta_3\} \cup \{depression,\ sleep\_aid\_misuse,\ anxiety\}$
λ_2	0.02592	$\{\theta_1, \theta_2, \theta_3, \theta_5\} \cup \{depression,\ \neg sleep\_aid\_misuse,\ anxiety\}$
λ_3	0.03312	$\{\theta_1, \theta_2, \theta_3, \theta_4\} \cup \{depression,\ \neg sleep\_aid\_misuse,\ anxiety\}$
λ_4	0.29808	$\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\} \cup \{\neg sleep\_aid\_misuse,\ \neg anxiety\}$
λ_5	0.00032	$\{\theta_1, \theta_2, \theta_3\} \cup \{anxiety\}$
λ_6	0.00288	$\{\theta_1, \theta_2, \theta_3, \theta_5\} \cup \{\neg sleep\_aid\_misuse,\ anxiety\}$
λ_7	0.00368	$\{\theta_1, \theta_2, \theta_3, \theta_4\} \cup \{\neg anxiety\}$
λ_8	0.03312	$\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\} \cup \{\neg sleep\_aid\_misuse,\ \neg anxiety\}$
λ_9	0.00048	$\{\theta_1, \theta_2, \theta_3\} \cup \{depression,\ sleep\_aid\_misuse,\ anxiety\}$
λ_{10}	0.00432	$\{\theta_1, \theta_2, \theta_3, \theta_5\} \cup \{depression,\ \neg sleep\_aid\_misuse,\ anxiety\}$
λ_{11}	0.04752	$\{\theta_1, \theta_2, \theta_3, \theta_4\} \cup \{sleep\_aid\_misuse,\ \neg anxiety\}$
λ_{12}	0.42768	$\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\} \cup \{\neg sleep\_aid\_misuse,\ \neg anxiety\}$
λ_{13}	0.00012	$\{\theta_1, \theta_2, \theta_3\}$
λ_{14}	0.00108	$\{\theta_1, \theta_2, \theta_3, \theta_5\} \cup \{\neg sleep\_aid\_misuse\}$
λ_{15}	0.01188	$\{\theta_1, \theta_2, \theta_3, \theta_4\} \cup \{\neg anxiety\}$
λ_{16}	0.10692	$\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\} \cup \{\neg anxiety,\ \neg sleep\_aid\_misuse\}$

Fig. 9 Atoms that are warranted in each possible EM world, given the AM and annotation function in Figure 8.

Before introducing these formulas, we need to present a simple extension to the concept of "warrant status", which is up to now defined for literals. The following definition is a simple extension to conjunctions or disjunctions of literals.

Definition 12 (Warranting a Conjunction/Disjunction of Literals) Let Π_{AM} be a ground PreDeLP program and Q be either a conjunction or disjunction of ground literals $L_1, \ldots, L_n$. The *warrant status* of Q with respect to Π_{AM} is defined as follows:

1. If Q is a single literal L, then the warrant status of Q is the warrant status of L in Π_{AM}.
2. If $Q = Q_1 \wedge Q_2$ then the warrant status of Q is:
   - *Yes* iff the warrant status of both Q_1 and Q_2 is *Yes*;
   - *No* if the warrant status of either Q_1 or Q_2 is *No*; and
   - *Undecided* whenever neither of the above cases hold.
3. If $Q = Q_1 \vee Q_2$ then the warrant status of Q is:
   - *Yes* iff the warrant status of either Q_1 or Q_2 is *Yes*;
   - *No* if the warrant status of both Q_1 and Q_2 is *No*; and
   - *Undecided* whenever neither of the above cases hold.

Using Definition 12, we can easily extend the *nec* and *poss* notations (cf. Page 14) to conjunctions and disjunctions of literals.

The following result is a consequence of the fact that conflicting literals cannot be warranted in (Pre)DeLP [17].

Proposition 1 *Let Π_{AM} be a ground PreDeLP program and $Q = L_1 \wedge \ldots \wedge L_n$ be a conjunction of ground literals. Then,* only one *of the following cases holds: (i) $P \vdash_{war} Q$, (ii) $P \vdash_{war} \neg Q$, or (iii) the warrant status of Q is undecided.*

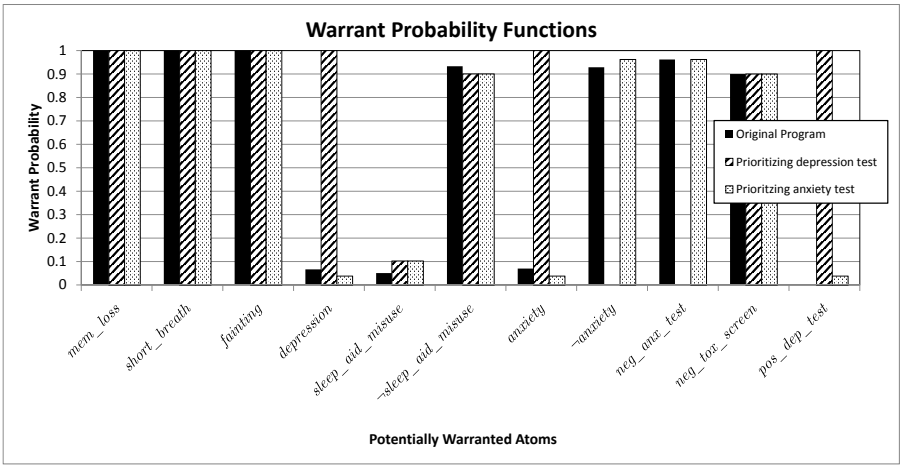

Fig. 10 Histogram depiction of the entailment probability functions for the programs of Example 9.

Proof In [17], a corresponding trichotomy result was shown for literals, *i.e.*, the warrant status for any literal is one and only one of Yes, No, and Undecided. Our result is a direct consequence of this and Definition 12. □

Warrant Probability Functions are then simply defined as partial mappings with signature:

$\Upsilon_{\mathcal{I}} : basic_{AM} \rightarrow [0, 1]$

such that for $f \in basic_{AM}$, $\Upsilon_{\mathcal{I}}(f) = p$ if and only if $\sum_{\lambda \in nec(f)} Pr(\lambda) = p$. [4] When the program is clear from context, we drop the subscript and write simply Υ. In the following, we use notation $dom(\Upsilon)$ to denote the set of formulas for which Υ is defined. The table shown in Example 9 is a simple example of a WPF, whose domain is a handful of literals. The following is another example along the same vein.

Example 10 Figure 10 shows three examples of WPFs in which the domains are fixed to the set of literals that can be warranted in the input program. These functions are related to the revision described in Example 9: the black bars show the original probabilities, the striped bars give the probabilities yielded by the program obtained by favoring the inclusion of the positive depression test, while the light gray bars depict the probabilities obtained by favoring the negative anxiety test. Figure 11 shows the three revised programs. ■

Revision Objective Functions. The other building block allows us to effectively quantify *how good* a revision is considered to be. Towards this end,

[4] Note that this definition can easily be extended to deal with probability intervals as well (*i.e.*, using both *nec* and *poss*); here, for simplicity of presentation, we adopt this definition in order to work with point probabilities.

we define *revision objective functions* (ROFs, for short) as functions that take two DeLP3E programs $\mathcal{I}_1$ and $\mathcal{I}_2$, along with an epistemic input (f, af), and returns a positive real number. We keep the definition of ROFs very general in order to allow different kinds of objectives to be specified. The following is a simple example of a ROF over our running example, which makes use of warranting probability functions.

Example 11 Let us return once again to the medical diagnosis example. Suppose that we take the three revised programs we presented (Figure 11) – call them $\mathcal{I}_1$, $\mathcal{I}_2$, and $\mathcal{I}_3$ – and that we wish to compare them with respect to the effect of the last revision over the warranted atoms, taking the probabilities yielded by $\mathcal{I}_1$ as the baseline. So, we define the following revision objective function:

$$\Psi(\mathcal{I}, \mathcal{I}', (f, af')) = e^{-\sum_{L \in \mathbf{L}_{AM}, L \neq f} |\Upsilon_{\mathcal{I}}(L) - \Upsilon_{\mathcal{I}'}(L)|}$$

where $\Upsilon_{\mathcal{I}}$ is the WPF for program $\mathcal{I}$.

Intuitively, this function sums up all the differences between the probabilities of literals entailed by the programs, but ignores the input (if it is a literal). In this way, a *distance* between the original program and the two possible revisions is obtained based on the effects that each revision had on the probabilities with which literals are derived. So, for our revisions, we get:

$$\Psi(\mathcal{I}_1, \mathcal{I}_2, (pos\_dep\_test, af_2)) \approx 0.0547$$
$$\Psi(\mathcal{I}_1, \mathcal{I}_3, (pos\_dep\_test, af_3)) \approx 0.8611$$

Therefore, we can conclude that the revision yielding $\mathcal{I}_3$ is preferred over the one yielding $\mathcal{I}_2$ when this ROF is adopted. ∎

Note that the function presented in Example 11 is just one possibility; the framework is very flexible and allows the user to express many different functions, depending on the specific way in which they wish to express distances between the original program and a given revised program.

3.2.2 The Class **QAFO**

Given the basic constructs introduced above, we can now define the class of *quantitative* annotation function-based revision operators.

Definition 13 (The Class QAFO) Let $\mathcal{I} = (\Pi_{EM}, \Pi_{AM}, af)$, with $\Pi_{AM} = \Omega \cup \Theta \cup \Delta \cup \Phi$ be a DeLP3E program, $\star \in$ **AFO** be an annotation function-based belief revision operator, and Ψ be a revision objective function. Operator $\star$ is said to be a *quantitative af-based* operator (denoted $\star \in$ **QAFO**) if:

> Given an epistemic input (f, af'), we have that if $\mathcal{I}' = \mathcal{I} \star (f, af')$ then there does not exist DeLP3E program $\mathcal{I}'' = \mathcal{I} \bullet (f, af')$ such that $\Psi(\mathcal{I}, \mathcal{I}'', (f, af')) > \Psi(\mathcal{I}, \mathcal{I}', (f, af'))$,

where $\bullet \in$ **AFO** is an arbitrary operator.

		Analytical Model	af_1	af_2	af_3
Θ :	$\theta_1 =$	*mem_loss*	True	True	True
	$\theta_2 =$	*short_breath*	True	True	True
	$\theta_3 =$	*fainting*	True	True	True
	$\theta_4 =$	*neg_anx_test*	$\neg$*FN-anx_test*	False	$af_1(\theta_4)$
	$\theta_5 =$	*neg_tox_screen*	$\neg$*FN-tox_screen*	$af_1(\theta_5)$	$af_1(\theta_5)$
	$\theta_6 =$	*pos_dep_test*	True	True	$\neg af_1(\theta_4)$
Ω :	$\omega_1 =$	$\neg$*sleep_aid_misuse* $\leftarrow$ *neg_tox_screen*	True	True	True
	$\omega_2 =$	$\neg$*anxiety* $\leftarrow$ *neg_anx_test*	True	True	True
	$\omega_3 =$	*anxiety* $\leftarrow$ *depression*	True	True	True
	$\omega_4 =$	*depression* $\leftarrow$ *pos_dep_test*	True	True	True
Φ :	$\emptyset$				
Δ :	$\delta_1 =$	*sleep_aid_misuse* $\prec$ *mem_loss*, *depression*	True	True	True
	$\delta_2 =$	*anxiety* $\prec$ *short_breath*	{*anx_risk*}	$af_1(\delta_2)$	$af_1(\delta_2)$
	$\delta_3 =$	*depression* $\prec$ *mem_loss*	{*dep_risk*}	$af_1(\delta_3)$	$af_1(\delta_3)$
	$\delta_4 =$	*anxiety* $\prec$ *fainting*	{*anx_risk*}	$af_1(\delta_4)$	$af_1(\delta_4)$

Fig. 11 The DeLP3E program from the running example, after performing three revisions: (i) The addition of the θ_4 and θ_5, as discussed in Example 9; (ii) The revision by *pos_dep_test* by prioritizing this input; and (iii) The same revision but prioritizing *neg_anx_test*.

So, this subclass of **AFO** simply takes a revision objective function and uses it to obtain the best possible revised program. The following example, based on our previous work on applications of DeLP3E to problems in the cyber security domain [40], shows how **QAFO** operators can be applied to belief revision problems in real-world scenarios other than the running example.

Example 12 Suppose we are modeling a cyber security scenario in which a computer worm has been deployed and has infected millions of computers worldwide – by the time the worm is discovered, it is very difficult to reason about the origin and even the intended target of the attack. In this kind of situations, analysts are trying to solve the so-called *attribution problem*: given a cyber operation of interest, determine the party that was ultimately responsible for carrying it out [38].

Towards this end, we can model all knowledge available by means of a DeLP3E program $\mathcal{I} = (\Pi_{EM}, \Pi_{AM}, af)$, in which there is one distinguished predicate $condOp(A, O)$ in the AM that is intuitively read as "actor A conducted operation O". Furthermore, if we assume that only one actor is ever responsible for an operation (an assumption that can easily be removed), we have an integrity constraint of the form oneOf(C), where C is the set of all ground atoms built with the *condOp* predicate.

Given this setup, we can define a WPF with a domain consisting of some formulas of interest that reflect conditions that the analysts would like to remain relatively unaffected when incorporating new information. For instance,

suppose we define:

$$dom(\Upsilon) = \Big\{ \neg condOp(countryA, worm) \wedge \neg condOp(countryB, worm) \\ condOp(countryD, worm) \Big\},$$

denoting the fact that neither country A nor country B are responsible for deploying the worm, and that country D is. If we pair this WPF with the ROF from Example 11, the corresponding **QAFO** operator will prefer revisions that do not affect the conclusions already reached regarding the probabilities assigned to these statements.

In other words, this definition of $dom(\Upsilon)$, with the ROF in question, causes distances to be gauged relative to their effect on the probabilities assigned to the suspicions that (i) neither country A nor country B carried out the attack, and (ii) country D is behind the attack. Thus, such a setup causes the operator to prefer revisions that keep the probabilities assigned to such suspicions as close as possible to the ones yielded by the original program. ∎

In the next section, we study the computational complexity associated with this approach to belief revision in the DeLP3E setting.

4 Computational Complexity

In this section, we will focus on some of the computational aspects of quantitative af-based belief revision operations.

As a first observation, we have that the problem of deciding the warranting status in a (classical) PreDeLP program has not yet been pinpointed. In [4], the authors present a proof for the PSPACE-completeness of the problem of marking a *given* dialectical tree; PSPACE membership for deciding the warrant status is therefore a direct consequence of this result, since a dialectical tree can be built within this budget. As a step towards finding a lower bound for the complexity of the problem in general, we have the following.

Proposition 2 *Let Π_{AM} be a ground PreDeLP program and L be a ground literal. Deciding $\Pi_{AM} \vdash_{war} L$ is* NP*-hard.*

Proof By reduction from 3SAT-CNF; we therefore start with an input formula F with n variables $X_1, \ldots, X_n$ and m clauses $C_1, \ldots, C_m$. We produce a PreDeLP program Π_{AM} with the following predicates: f, $x_1, \ldots, x_n$, and $c_1, \ldots, c_m$. We then set:

$$\Phi = \{f, \neg f, x_1, \neg x_1, \ldots, x_n, \neg x_n\}$$

$$\Delta = \{c_j \prec x_i \mid X_i = T \text{ makes clause } C_j \text{ true}\} \cup \\ \{c_j \prec \neg x_i \mid X_i = F \text{ makes clause } C_j \text{ true}\} \cup \\ \{f \prec c_i \mid \text{for each clause } C_i\}$$

$$\Omega = \Theta = \emptyset$$

Next, we set the comparison criterion to specificity (the default in PreDeLP), except for the following exceptions:

$\langle\{x_i, c_j \prec x_i\}, c_j\rangle$ is always preferred over $\langle\{\neg x_i, c_j \prec \neg x_i\}, c_j\rangle$

$\langle\{\neg f\}, \neg f\rangle$ is preferred over $\langle\{f\}, f\rangle$

Now, we must show that $\Pi_{AM} \vdash_{\mathsf{war}} f$ if and only if there exists a satisfying assumption for formula F. Suppose $\Pi_{AM} \vdash_{\mathsf{war}} f$; the only way this can happen is if argument $\langle\{\neg f\}, \neg f\rangle$ (the only counterargument to $\langle\{f\}, \rangle$) is defeated, which happens if and only if all arguments that defeat this argument are in turn undefeated. Note that the only arguments capable of being in this situation are the ones using the rules with c_j in the head. Therefore, if all such arguments are undefeated it must be the case that either x_i or $\neg x_i$ can be chosen for each variable X_i in such a way that all clauses are satisfied, which holds if and only if there exists a satisfying assumption for F. □

As a corollary to Proposition 2, we have that deciding our extended notion of warrant status remains within the same complexity bounds.

Corollary 1 *Let Π_{AM} be a ground PreDeLP program and Q be either a conjunction or disjunction of ground literals. Deciding $\Pi_{AM} \vdash_{\mathsf{war}} Q$ is* NP*-hard and in* PSPACE*.*

Proof (Sketch) Applying Definition 12, the warrant status of Q can be determined in polynomial time based on the warrant status of its literals; therefore, the same complexity results for determining the warrant status of a single literal apply. □

Assumption: Since, as stated above, the precise complexity of deciding the warrant status of a literal in a PreDeLP program is not yet known, and with the objective of separating the complexity of this problem from the complexity of the problems inherent to quantitative belief revision in DeLP3E programs, in the following we will make the assumption that classical warranting in PreDeLP is decidable in polynomial time. This is not an unreasonable assumption if we consider the possibility of pre-compiling inferences [3] or having tractable approximation algorithms to address the problem. We call this the *polynomial-time warranting* (PTW) assumption. Note that, even though this assumption does not hold in general, it is a useful tool in the analysis of the complexity of the problems studied here; it is also with this spirit that we make use of the PTW assumption.

Unfortunately, our first result regarding the probabilistic extension of PreDeLP tells us that computing WPFs runs into a computational tractability hurdle.

Theorem 1 *Under the PTW assumption, computing the warrant probability function for a DeLP3E program is* #P*-hard.*

Proof We will prove the statement by reduction from #3CNF-SAT. Given formula F in 3CNF with n variables and m clauses, generate a DeLP3E program as follows: in the AM, there is one atom f, one atom c_i for each clause in F and two atoms for each variable V in, which we denote with pos_V and neg_V. For each clause C_i in F of the form $\hat{X} \vee \hat{Y} \vee \hat{Z}$, where $\hat{V}$ denotes either V or $\neg V$, we have strict rules:

$c_i \leftarrow \hat{x}$
$c_i \leftarrow \hat{y}$
$c_i \leftarrow \hat{z}$

where $\hat{v}$ denotes pos_V if V is positive in the clause and neg_V if it is negative. Next, we have the strict rule:

$f \leftarrow c_1, \ldots, c_m$

and facts pos_V and neg_V for each variable V in the input formula.

In the EM we have atoms *event-*pos_V and *event-*neg_V for each variable V in F. The probability distribution assigns probability 0.5 to each individual event, probability 0 to the conjunction of both events for the same variable, and probability 1 to their disjunction.

Finally, the annotation function θ assigns formula True to all components of the AM, except the facts, for which we have $af(pos_V) =$ *event-*pos_V and $af(neg_V) =$ *event-*neg_V.

Now we can see that, with this DeLP3E program, if the warranting probability function Υ assigns probability p to atom f, we have that:

$\sum_{\lambda \in nec(f)} Pr(\lambda) = p$.

Clearly, from our construction we know that $\lambda \in nec(f)$ iff λ corresponds to a satisfying assignment for formula F. Therefore, $p = \frac{k}{2^n}$, where k is the number of satisfying assignments for F. Solving for k, we have $k = p \cdot 2^n$. □

The complexity class #P contains problems related to *counting* solutions (or, in Turing machine terms, accepting paths) to problems in NP. The decision version of this class is called PP, and contains problems decidable by a probabilistic Turing machine in polynomial time, with error probability less than a certain proportion (say, 1/2). Unfortunately, Toda's theorem [47] tells us that a polynomial-time Turing machine with either a PP or #P oracle can solve all problems in the polynomial hierarchy.

Though it might be surmised that the #P-hardness is caused solely by the computation of probabilities (as is the case in many probabilistic formalisms), by analyzing the proof of Theorem 1 we can quickly arrive at the following conclusion.

Observation 1 *Computing the warrant probability function for a DeLP3E program is* #P*-hard even in the special case in which probabilities associated with EM worlds can be computed in* PTIME.

Though this intractability holds in general, restricting the EM can soften the impact on complexity. For instance, if we assume that Nilsson's probabilistic logic [33] is used then the complexity is lowered, as we show next; first, we introduce a lemma that will be used in the proof of this result:

Lemma 1 ([6,12]) *If a system of m linear equalities and/or inequalities has a nonnegative solution, then it has a nonnegative solution with at most m positive variables.*

This result was first introduced in [6], and later used in [12] to show that deciding the validity of a formula in their logic is NP-complete. We can now state our result.

Proposition 3 *Under the PTW assumption, and assuming that Nilsson's probabilistic logic is used in the EM, computing the warrant probability function for a DeLP3E program is* NP*-complete.*

Proof
Membership: Lemma 1 states that a solution to a linear program is guaranteed to exist where only a number of the variables that is linear in the number of constraints in the EM are set to a non-zero value. This implies the existence of a number of EM worlds with non-zero probability that is linear in the number of statements in the probabilistic model. A witness can therefore be verified in polynomial time.

Hardness: NP-hardness is a consequence of the well-known fact that SAT is reducible to computing probabilities in Nilsson logic (cf. [27] for a detailed proof). □

The previous result gives us a hint towards reaching the next one: if we combine the simplifying assumption that probabilities can be computed tractably with the further assumption that the number of EM worlds that have non-zero probability is bounded by a polynomial (condition 1 below), then we are guaranteed that computing WPFs is also tractable.

Corollary 2 *Let $\mathcal{I} = (\Pi_{EM}, \Pi_{AM}, af)$, with $\Pi_{AM} = \Omega \cup \Theta \cup \Delta \cup \Phi$, be a DeLP3E program. If we make the following assumptions:*

1. *$|\{\lambda \mid \lambda \in \mathcal{W}_{EM}$ and $Pr(\lambda) > 0\}| \in O(poly(n))$, where n represents the size of the input;*
2. *$Pr(\lambda)$ can be computed in* PTIME *for any $\lambda \in \mathcal{W}_{EM}$; and*
3. *the PTW assumption holds,*

then warrant probability functions for $\mathcal{I}$ can also be computed in PTIME*.*

Proof Direct consequence of the assumption that we have access to the polynomially many worlds that have non-zero probability. We thus simply keep an accumulator for each element in the domain of the WPF and iterate through the set of worlds, adding the probability of the world to each formula's accumulator if and only if it is warranted in the AM induced by that world. □

Unfortunately, the following result states that even in this scenario we still face an intractable hurdle when computing optimal revisions.

Theorem 2 *Let $\mathcal{I} = (\Pi_{EM}, \Pi_{AM}, af)$, with $\Pi_{AM} = \Omega \cup \Theta \cup \Delta \cup \Phi$, be a DeLP3E program, $\star \in$ **QAFO**, and Ψ be a revision objective function that can be computed in polynomial time. If we have that:*

1. *$|\{\lambda \mid \lambda \in \mathcal{W}_{EM}$ and $Pr(\lambda) > 0\}| \in O(poly(n))$, where n represents the size of the input;*
2. *$Pr(\lambda)$ can be computed in* PTIME *for any $\lambda \in \mathcal{W}_{EM}$; and*
3. *the PTW assumption holds,*

then deciding if $\Psi(\mathcal{I}, \mathcal{I}\star(f, af'), (f, af')) \leq k$ for some $k \in \mathbb{R}$, is NP*-complete.*

Proof
Membership: Given $\mathcal{I}'$, we can check in polynomial whether each world with non-zero probability induces a maximal consistent subset of $\mathcal{I}$ in that world; by construction, **AFO** operators are only constrained to make such changes in worlds with probability greater than zero. Furthermore, since by hypothesis we know that Ψ can be computed in polynomial time, we can decide whether or not the witness satisfies the constraints.

Hardness: By reduction from SUBSET-SUM; we therefore start from an instance of this problem, which consists of a set of n integers $x_1, \ldots, x_n$ and another integer c; the goal is to verify if there exists a subset of the numbers that add up to c. Lets then create an instance of our problem, starting with a DeLP3E program with one atom num_i in the AM for each x_i in the input instance, as well as an auxiliary atom p; there will also be a corresponding atom *event-num*$_i$ in the EM for each such atom. The probability distribution associated with these random variables is set to an arbitrary one satisfying the condition that only a number polynomial in n has non-zero probability.

Add as facts all atoms num_i, with the annotation function defined as $af(num_i) =$ *event-num*$_i$. Next, add the strict rule:

$$\omega : \neg num_i \leftarrow p$$

such that $af(\omega) =$ True. Set the epistemic input to fact θ_{in} with $af(\theta_{in}) =$ *event-num*$_1 \vee \ldots \vee$ *event-num*$_n$. Finally, set the function to optimize to:

$$obj(\mathcal{I}, \mathcal{I}') = \begin{cases} 1 & \text{if } res = \{num_i \mid num_i \in \Pi_{AM}, af''(num_i) \not\models \bot\} \\ & \text{is such that } \sum_{num_i \in res} x_i = c \\ 0 & \text{otherwise.} \end{cases}$$

That is, the objective function only considers the revised DeLP3E program, and takes value 1 if and only if the atoms that belong to this program correspond to numbers that, taken together, sum up to c. Thus, all we have to do is check if the optimal revision yields a value of 1 for the objective function in order to decide the given subset-sum instance. □

The reader may note that the construction used in the proof of Theorem 2 uses a very powerful objective function that essentially encodes the NP-hard problem; furthermore, this objective function is not based on WPFs. We now provide an alternative result that proves NP-completeness under the same conditions, but assumes that the objective function is simply the sum of the probabilities assigned by the WPF to the set of ground atoms in the language associated with the AM.

Theorem 3 *Let $\mathcal{I} = (\Pi_{EM}, \Pi_{AM}, af)$, with $\Pi_{AM} = \Omega \cup \Theta \cup \Delta \cup \Phi$, be a DeLP3E program, $\star \in$ **QAFO**, and Ψ be a revision objective function that can be computed in polynomial time. If we have that:*

1. *$|\{\lambda \mid \lambda \in \mathcal{W}_{EM}$ and $Pr(\lambda) > 0\}| \in O(poly(n))$, where n represents the size of the input;*
2. *$Pr(\lambda)$ can be computed in* PTIME *for any $\lambda \in \mathcal{W}_{EM}$; and*
3. *the PTW assumption holds,*

then deciding if $\Psi(\mathcal{I}, \mathcal{I} \star (f, af'), (f, af')) \leq k$ for some $k \in \mathbb{R}$, is NP*-complete even when Ψ is defined as $\sum_{a \in \mathbf{G}_{AM}} \Upsilon(a)$.*

Proof
Membership: As the ROF can still be computed in polynomial time, the membership result of Theorem 2 still holds.

Hardness: We show NP-hardness by reduction from SIMPLE MAX CUT (SMC) [19]. The SMC problem takes as input an undirected graph $G = (V, E)$ and $k \geq 0$, and decides if there exist sets $V_1, V_2 \subseteq V$ such that $|\{(u, v) \in E : u \in V_1, v \in V_2\}| \geq k$.

Let x be a the only atom in $\mathbf{G}_{EM}$. We will define a simple Π_{EM} that sets the probability of atom x to 1.0. We specify the atoms in $\mathbf{G}_{AM}$ as follows: for each $v_i \in V$ we create two atoms, $set_1(v_i), set_2(v_i)$. For each edge $(v_i, v_j) \in E$ we create atom $edge(v_i, v_j)$ (we assume that each bi-directional edge is specified by one atom an that the order of the arguments for the *edge* predicate is arbitrary but consistent). We will also have an additional atom *query* in $\mathbf{G}_{AM}$, which will also act as the formula for the epistemic input. We create Π_{AM} with the following elements:

- For each $v_i \in V$ add the following strict rules:
  - $set_1(v_i) \leftarrow query$
  - $\neg set_1(v_i) \leftarrow query$
  - $set_2(v_i) \leftarrow \neg set_1(v_i)$
- For each edge $(v_i, v_j) \in E$ add strict rules $edge(v_i, v_j) \leftarrow set_1(v_i), set_2(v_j)$ and $edge(v_i, v_j) \leftarrow set_2(v_i), set_1(v_j)$

For each element $y \in \Pi_{AM}$, we define the annotation function $af(y) =$ True. For the epistemic input, let af' be the extension of af such that $af'(query) = x$. Finally, let the ROF be defined as in the statement of the theorem. Further, for ease of notation, let af^* be the annotation function returned after the belief revision operation takes place.

Clearly, this construction can be performed in polynomial time. Further, note that the original program is consistent, since none of the rules in Π_{AM} ever fire since there are no facts.

Observations: We notice right away that any valid belief revision operator must return an annotation function af^* such that $\{x\} \models af^*(query)$ – as this atom is needed to ensure the objective function has a non-zero value (which is clearly possible). Further, any optimal solution where $af^*(query) \equiv \mathsf{True}$ can be replaced with a solution where $af^*(query) = x$. We also note that in any optimal solution, the size of the set $\{set_X(v_i) \mid \Upsilon(set_X(v_i) = 1.0\}$ is equal to $|V|$ – this is how we capture the notion that for each vertex v_i, exactly one of $set_1(v_i), set_2(v_i)$ will be warranted under world $\{x\}$); hence, we capture the requirement from the instance of SMC that the sets V_1, V_2 is a partition of V. We also note that the annotation function in the solution af^* must only modify the return values for strict rules of the form $set_1(v_i) \leftarrow query$ and $\neg set_1(v_i) \leftarrow query$.

Claim 1: Let V_1, V_2 be an optimal solution to an instance of SMC. Then, the optimal solution to the corresponding revision problem has an objective function whose value is greater than or equal to $|\{(u, v) \in E : u \in V_1, v \in V_2\}|+|V|+1$. This can clearly be achieved by a solution where for each $v_i \in V_1$, $af^*(set_1(v_i) \leftarrow query) = x$ and for each $v_i \in V_2$, $af^*(\neg set_1(v_i) \leftarrow query) = \neg x$.

Claim 2: Let R be the value of the objective function returned in an optimal solution to the revision problem. Then, the number of edges returned in the corresponding instance of SMC is greater than or equal to $R - |V| - 1$. By the aforementioned observations, this claim is equivalent to saying that the number of positive literals of the form $edge(v_i, v_j)$ that are warranted in the world $\{x\}$ is less than or equal to the number of edges returned in the optimal solution to the corresponding instance of SMC. Suppose, by way of contradiction, that the number of literals of that form that are warranted under $\{x\}$ is greater than the number of edges in the optimal solution to the corresponding instance of SMC. By the construction, for each literal of the form $edge(v_i, v_j)$, exactly one of the following pairs of literals must also be warranted: $set_1(v_i), set_2(v_j)$ or $set_2(v_i), set_1(v_j)$. Therefore, we can partition the corresponding vertices from SMC into two sets – V_1', V_2' and the number of edges in the set $\{(u, v) \in E : u \in V_1', v \in V_2'\}$ must then be greater than the number of edges in the optimal solution to the SMC problem. However, this is not possible, as this is also (by construction) the same objective function that is optimized in that problem – hence, we reach a contradiction.

The proof of hardness follows directly from Claims 1 and 2. □

So, the proof of Theorem 3 illustrates that the quantified revision problem is still NP-hard when the EM and the number of EM worlds, and (hence) the computation of the WPF is not a source of complexity – even when the ROF used is a simple aggregate over WPFs of atoms. Further, as we can embed the Simple Max Cut problem, the ROF – even a simple sum over WPFs – will not necessarily be monotonic, even when using a revision operator that satisfies

Algorithm warrantFormula($\mathcal{F}$)

1. For each tree $(V, E) \in \mathcal{F}$ with $(V, E) = \mathcal{T}^*{}_i(\langle \mathcal{A}, L \rangle)$ do
2. For each $v \in V$ with $label(v) = \bigwedge_{f_j \in \mathcal{A}'} af(f_j)$ do
3. While $|V| > 1$ do
4. For each $v = \langle \mathcal{A}', L' \rangle \in \{v' \in V \mid children(v') \subseteq leaves(V)\}$ do
5. $label(v) := label(v) \wedge \neg \bigwedge_{v' \in children(v)} \neg label(v')$;
6. End for;
7. $V := V \setminus leaves(V)$;
8. End while;
9. $f_i := label\big(root(\mathcal{T}^*{}_i(\langle \mathcal{A}, L \rangle))\big)$;
10. End for;
11. End for;
12. Return $\bigvee_i f_i$.

Fig. 12 An algorithm that takes a classical dialectical forest and computes a logical formula specifying the possible worlds under which a given literal is warranted.

the Inclusion postulate (where the set of worlds satisfying $af^*(y) \subseteq af'(y)$). This also shows NP-completeness when the belief revision operator performs modifications to Π_{AM} (by removing elements, as discussed in [41]) as setting $af^*(y) = \neg x$ can be viewed as an operation that is equivalent to removing it from Π_{AM}.

We also note that the related problem of consolidation or contraction by *falsum*, where we start with an inconsistent program and then must adjust the annotation function to make it consistent, can also be shown to be NP-complete by a simple modification to the proof: we fix the epistemic input to True, and change the rules of the form $set_1(v_i) \leftarrow query, \neg set_1(v_i) \leftarrow query$ to facts of the form $set_1(v_i), \neg set_1(v_i)$.

5 Warranting Formulas

We now focus on an algorithmic approach that can be used to compute approximate solutions and therefore address the computational intractability that we have seen in the results above.

In the following, given a dialectical forest $\mathcal{F}(L)$ and a node V corresponding to argument A, we will use the notation $label(V) = \bigwedge_{c \in A} af(c)$. For a given probabilistic argumentation framework, literal, and dialectical tree, Algorithm warrantFormula in Figure 12 computes the formula describing the set of possible worlds that are warranting scenarios for the literal. Intuitively, this algorithm creates a formula for every dialectical tree in the forest associated with an argument – the algorithm iteratively builds a formula associated with the environmental conditions under which the argument in the root of the tree is undefeated. It then returns the disjunction of all such formulas in that forest. We refer to this disjunction as the *warranting formula* for the literal.

The following result states the correctness of the warrantFormula algorithm.

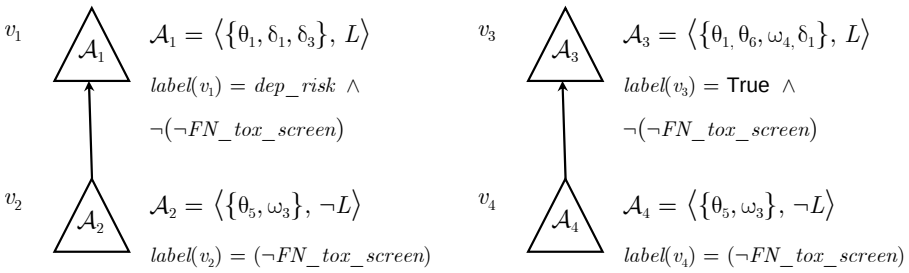

Fig. 13 Dialectical forest for literal $L = sleep\_aid\_misuse$ composed of trees $\mathcal{T}_1$ (left) and $\mathcal{T}_2$ (right).

Proposition 4 *Given forest* $\mathcal{F}^*(L)$,

$$nec(L) = \Big\{\lambda \in \mathcal{W}_{EM} \mid \lambda \models \mathsf{warrantFormula}\big(\mathcal{F}^*(L)\big)\Big\}$$
$$poss(L) = \Big\{\lambda \in \mathcal{W}_{EM} \mid \lambda \not\models \mathsf{warrantFormula}\big(\mathcal{F}^*(\neg L)\big)\Big\}.$$

Proof (Sketch)

Claim 1: $nec(L) \subseteq \{\lambda \in \mathcal{W}_{EM} \mid (\lambda \models \mathsf{warrantFormula}(\mathcal{F}^*(L)))\}$. To prove this claim, it suffices to show that if $\mathcal{T}_\lambda\langle \mathcal{A}, L\rangle$ is a valid dialectical tree for L then $\lambda \models \mathsf{warrantFormula}(\mathcal{F}^*(L))$. Suppose, BWOC, $\mathcal{T}_\lambda\langle \mathcal{A}, L\rangle$ is such a tree where $\lambda \not\models \mathsf{warrantFormula}(\mathcal{F}^*(L))$. We note that this tree shares a root and is a subtree of a tree in $\mathcal{F}^*(L)$. Also, for each node v in the tree, at line 2, we have $\lambda \models label(v)$. Hence, we can continue the proof by replacing line 2 with $\forall v, label(v) = for(\lambda)$ and showing that $\mathsf{warrantFormula}(\mathcal{F}^*(L)) = false$. However, we can conclude that $\mathsf{warrantFormula}(\mathcal{F}^*(L)) = for(\lambda)$ since the tree has an odd depth by definition – this is a contradiction.

Claim 2: $nec(L) \supseteq \{\lambda \in \mathcal{W}_{EM} \mid (\lambda \models \mathsf{warrantFormula}(\mathcal{I}, L))\}$. Suppose, by way of contradiction, that the claim is false. Then there must exist a root-sharing subtree of an element of $\mathcal{F}^*(L)$ such that for each v, at step 4 $\lambda \models label(v)$ and does not exist in λ. However, this is a contradiction by Definition 7.

Claim 3: The second part of the statement follows directly from Claims 1-2 and the fact that $poss(L) = \mathcal{W}_{EM} \setminus nec(\neg L)$. □

Even though warranting formulas are another way of solving the problem of computing probabilities exactly, our main motivation for developing it was to explore options for pursuing tractable algorithms, as discussed next.

The following is an example of the warranting formula approach in the setting of our running example.

Example 13 Consider the DeLP3E in our running example as shown in Figure 11 with annotation function af_1. If we run Algorithm warrantFormula for literal *sleep_aid_misuse*, we start with the dialectical forest shown in Figure 13.

Suppose that the algorithm begins with tree $\mathcal{T}_1$ (on the left); the only leaf of this tree corresponds to vertex v_2 for argument $\mathcal{A}_2$, and its label remains

the conjunction of all annotations of elements in the argument – $label(v_2) = \neg FN\_tox\_screen$. The algorithm then moves to the next node up, which is already the root, and updates the label by adding the conjunction with the negation of its child, which yields:

$$label(v_1) = dep\_risk \wedge \neg\big(\neg FN\_tox\_screen\big) = dep\_risk \wedge FN\_tox\_screen.$$

Processing tree $\mathcal{T}_2$ similarly yields:

$$label(v_3) = \mathsf{True} \wedge \neg\big(\neg FN\_tox\_screen\big) = FN\_tox\_screen.$$

Finally, the algorithm outputs the disjunction of these two formulas, which is simply FN_tox_screen. ∎

Outlook: Towards Tractable Computations

By applying the warrantFormula algorithm to the dialectical forest for a given literal L, we can obtain the sets $nec(L)$ and $poss(L)$ with a running time proportional to the size of the forest and the annotation formulas – though the worst-time complexity has not been determined exactly, it is safe to conjecture that the worst case is intractable. However, the warranting formula approach opens the door to several possibilities for heuristics and approximate computations that either avoid exhaustively enumerating worlds in $\mathcal{W}_{EM}$ or working with full forests (or both). When combined with existing heuristics for classical argumentation (the AM) and probabilistic models (the EM), this provides us with a much more efficient way to compute warranting probability functions. Experimental evaluations for such hypotheses are currently underway.

The use of the warranting formula approach can have several impacts in the implementation of specific **QAFO** operators. First, warrant probability functions Υ in this setting can now be redefined to map elements in their domain to warranting formulas instead of probabilities as in their original formulation. Revision objective functions now have at their disposal formulas instead of raw numbers. This opens up the possibility for specific implementations to leverage optimizations such as applying SAT algorithms to decide whether $Pr_{\mathcal{I}_1}(L) \geq Pr_{\mathcal{I}_2}(L)$ (which can be decided via the SAT check $\Upsilon_{\mathcal{I}_1}(L) \Rightarrow \Upsilon_{\mathcal{I}_2}(L)$). Such an approach is clearly compatible with heuristic optimizations that may, for instance, sacrifice precision for greater tractability.

An alternative class of operators can thus be defined based on the same ideas as **QAFO** except that approximations are allowed instead of exact computations. There is much work to be done in this direction, which is outside the scope of the current paper.

6 Related Work

This paper continues the research line that began with two works on belief revision in structured probabilistic argumentation. In [39], we introduced the

DeLP3E formalism (which is called P-PreDeLP in that work) and annotation-function based belief revision (the class **AFO**), while in [40] we studied a special case of entailment queries and showed how the framework can be applied to a cyber-attribution problem.

As we have seen, the main problem studied in belief revision is the study of how epistemic states should be changed in response to epistemic inputs. Traditionally, epistemic states have taken the form of either belief sets (sets of formulas closed under consequence) [1,18] or belief bases [24,23] (which are not closed). Our goal is to ultimately apply our results to real-world domains, and therefore we focus our attention on belief bases. Epistemic states in our case consist of formulas over which argumentation-based reasoning is carried out and to which we couple a general probabilistic model. The relationship between argumentation and belief revision can be traced back to [10]; in this regard, the work that is most closely related to how we approach their combination is that of [14], where explanation-based revision operators are studied. For a discussion on the relationship between the two areas of study, see [15] and [13].

Regarding argumentation systems that feature some quantitative form of reasoning under uncertainty, we point out that the combination of probabilistic reasoning with argumentation systems has been the focus of several works in the recent past. A significant portion of this work, however, has adopted abstract argumentation systems as the basis for the probabilistic extension [29, 45,25,16]. Contrary to structured argumentation systems like the one adopted in this work, abstract argumentation is focused on the study of attacks among arguments without inspecting their composition. There are also some works that combine structured argumentation approaches with models for reasoning under uncertainty – the first of these was [22]; the work of [28], which was developed even earlier, combines structured argumentation with abstract uncertainty measures but does not explicitly handle probability. Several other works followed; for instance, in [5], the authors develop a possibilistic extension to DeLP, and [26] presents an approach based on probabilistic logic. The main difference between these works and our own is that here we adopt a bipartite knowledge base, where one part models the knowledge that is not inherently probabilistic – uncertain knowledge is modeled separately, thus allowing a clear separation of interests between the two kinds of models. This kind of approach is not novel; it has been adopted in several frameworks, such as the Independent Choice Logic [35] or probabilistic ontology languages for the Semantic Web (see [20], and references within).

From a quantitative take on belief revision, which is the specific topic of this paper, there hasn't been much work in the precise direction taken here. Perhaps the earliest proposal with such an idea in mind is that of maxichoice revision operators [2], which ensure that minimal changes are made to the knowledge base when revisions are performed; in fact, our class of **AFO** operators (and therefore also **QAFO**) perform maxichoice revision operations in each possible EM world. In the related setting of belief contraction operators, David Makinson [31] has defended the use of maxichoice operators, explaining that some counterintuitive behaviors that this approach leads to is

due to its "misapplication" to belief sets (which, we recall, are closed under consequence). Another quantitative proposal is the one presented in [7], where revisions are carried out according to a notion of distance between worlds, such as the number of propositional symbols that separate one model from another. These notions, however, do not correspond directly with the one adopted here, since in our setting we are pursuing a revision that is optimal from the point of view of its effect on the probabilistic aspect of the consequences of the knowledge base, while the knowledge bases in [2] are non-probabilistic. Another interesting approach to belief revision from a quantitative standpoint is ranking theory [43], which is a normative theory of the dynamics of belief. Again, this approach is not directly related to the one taken here; however, exploring how this well-studied approach can be applied in conjunction with argumentation in the way that probability theory is applied in this work is an interesting avenue for future work.

Along the same vein of seeking to minimize information loss, and in the closely related setting of inconsistency management, the work of [21] proposes the notion of preferred repair based on so-called "consistency scores"; our quantitative approach to performing belief revision operations is loosely based on this proposal. Also in the inconsistency management literature, the recent work of [9,8] proposes to resolve conflicts at a global level to minimize information loss but using incision functions instead. Another work in inconsistency management is that of [46] proposes measures of inconsistency for probabilistic logics. Apart from [21], this is perhaps the closest work in spirit to the one presented here, though the underlying language used (probabilistic conditional logic) is quite different and that work does not address the problem of belief revision – their measures, however, could be applied to efforts in line with our own. The adaptation of measures of probabilistic inconsistency such as the ones proposed in [46] to DeLP3E and their use in quantitative belief revision operators is the topic of ongoing and future work.

7 Conclusions and Future Work

In this work, we tackled the problem of incorporating a new piece of information to an existing knowledge base; specifically, we adopted the DeLP3E model, which is an extension of the structured argumentation language DeLP with presumptions (PreDeLP) via the incorporation of annotations that refer to events for which we have underlying probabilistic information.

The main focus of this paper was to further explore a class of belief revision operators called **AFO** (for annotation function-based revision) that we proposed recently in [39,41] by considering operators in this class that have the further requirement of "quantitative optimality" – this gave rise to the **QAFO** class of operators. Though this optimality criterion was kept as general as possible so that knowledge engineers can specify their preferences, we explored the computational complexity of the approach in general, arriving at a host of results that range from intractability for the general case to polynomial-time

special cases. finally, we presented an algorithm designed to compute the probability with which a literal is warranted via so-called warranting formulas, and provide some initial discussion regarding how this approach could be applied in the implementation of **QAFO** operators or approximations of them that trade theoretical guarantees for tractability in practice.

Future work along this line of research involves continuing with efforts to bridge the gap between the theoretical developments that have been steadily coming from the belief revision community and practical implementations that can be applied in real-world domains such as medical diagnosis (the topic of our running example) and the related problem of solving the attribution problem in cyber security and cyber warfare, as proposed in [40]. We are also investigating the use of different kinds of belief revision operators, for instance ones that are based on argumentation [14]. Finally, we are currently in the final stages of developing a fully-functional implementation of DeLP3E; incorporating the belief revision operators developed in [39, 41] and in this paper is the next step in the implementation effort.

Acknowledgements This work was supported by UK EPSRC grant EP/J008346/1—"PrOQAW", ERC grant 246858—"DIADEM", by NSF grant #1117761, by the Army Research Office under the Science of Security Lablet grant (SoSL) and project 2GDATXR042, DARPA project R.0004972.001, and funds provided by CONICET and Universidad Nacional del Sur, Argentina. The opinions in this paper are those of the authors and do not necessarily reflect the opinions of the funders, the U.S. Military Academy, or the U.S. Army.

References

1. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: Partial meet contraction and revision functions. J. Sym. Log. **50**(2), 510–530 (1985)
2. Alchourrón, C.E., Makinson, D.: On the logic of theory change: Contraction functions and their associated revision functions. Theoria **48**(1), 14–37 (1982)
3. Capobianco, M., Chesñevar, C.I., Simari, G.: Argumentation and the dynamics of warranted beliefs in changing environments. Intl. Journal on Autonomous Agents and Multiagent Systems (JAAMAS) **11**, 127–151 (2005)
4. Cecchi, L.A., Simari, G.R.: El marcado de un árbol dialéctico en DeLP es PSPACE-completo. In: Proc. of Congreso Argentino de Ciencias de la Computación (CACIC) (2011)
5. Chesñevar, C.I., Simari, G.R., Alsinet, T., Godo, L.: A logic programming framework for possibilistic argumentation with vague knowledge. In: Proc. of UAI 2004, pp. 76–84 (2004)
6. Chvátal, V.: Linear Programming. W.H.Freeman, New York (1983)
7. Dalal, M.: Investigations into a theory of knowledge base revision: Preliminary report. In: Proc. of AAAI, pp. 475–479 (1988)
8. Deagustini, C.A.D., Martinez, M.V., Falappa, M.A., Simari, G.R.: Improving inconsistency resolution by considering global conflicts. In: Proc. of SUM. Springer (2014, *To Appear*)
9. Deagustini, C.A.D., Martinez, M.V., Falappa, M.A., Simari, G.R.: Inconsistency resolution and global conflicts. In: Proc. of ECAI (2014, *To Appear*)
10. Doyle, J.: A truth maintenance system. Artif. Intell. **12**(3), 231–272 (1979)
11. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif. Intell. **77**, pp. 321–357 (1995)

12. Fagin, R., Halpern, J.Y., Megiddo, N.: A logic for reasoning about probabilities. Information and Computation **87**(1/2), 78–128 (1990)
13. Falappa, M.A., Garcia, A.J., Kern-Isberner, G., Simari, G.R.: On the evolving relation between belief revision and argumentation. The Knowledge Engineering Review **26**(01), 35–43 (2011)
14. Falappa, M.A., Kern-Isberner, G., Simari, G.R.: Explanations, belief revision and defeasible reasoning. Artif. Intell. **141**(1/2), 1–28 (2002)
15. Falappa, M.A., Kern-Isberner, G., Simari, G.R.: Belief revision and argumentation theory. In: Argumentation in artificial intelligence, pp. 341–360. Springer (2009)
16. Fazzinga, B., Flesca, S., Parisi, F.: On the complexity of probabilistic abstract argumentation. In: Proc. of IJCAI 2013 (2013)
17. García, A.J., Simari, G.R.: Defeasible logic programming: An argumentative approach. TPLP **4**(1-2), 95–138 (2004)
18. Gardenfors, P.: Knowledge in flux: modeling the dynamics of epistemic states. MIT Press, Cambridge, Mass. (1988)
19. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York (1979)
20. Gottlob, G., Lukasiewicz, T., Martinez, M.V., Simari, G.I.: Query answering under probabilistic uncertainty in Datalog+/– ontologies. AMAI (2013)
21. Gottlob, G., Lukasiewicz, T., Martinez, M.V., Simari, G.I.: Query answering under probabilistic uncertainty in Datalog+/- ontologies. AMAI (2013)
22. Haenni, R., Kohlas, J., Lehmann, N.: Probabilistic argumentation systems. Springer (1999)
23. Hansson, S.: Semi-revision. J. of App. Non-Classical Logics **7**(1-2), 151–175 (1997)
24. Hansson, S.O.: Kernel contraction. J. Symb. Log. **59**(3), 845–859 (1994)
25. Hunter, A.: Some foundations for probabilistic abstract argumentation. In: Proc. of COMMA 2012, pp. 117–128 (2012)
26. Hunter, A.: A probabilistic approach to modelling uncertain logical arguments. Int. J. Approx. Reasoning **54**(1), 47–81 (2013)
27. Khuller, S., Martinez, M.V., Nau, D.S., Sliva, A., Simari, G.I., Subrahmanian, V.S.: Computing most probable worlds of action probabilistic logic programs: scalable estimation for $10^{30,000}$ worlds. AMAI **51**(2-4), 295–331 (2007)
28. Krause, P., Ambler, S., Elvang-Gørannson, M., Fox, J.: A logic of argumentation for reasoning under uncertainty. Computational Intelligence **11 (1)**, 113–131 (1995)
29. Li, H., Oren, N., Norman, T.J.: Probabilistic argumentation frameworks. In: Proc. of TAFA, pp. 1–16 (2011)
30. Lloyd, J.W.: Foundations of Logic Programming, 2nd Edition. Springer (1987)
31. Makinson, D.: On the status of the postulate of recovery in the logic of theory change. Journal of Philosophical Logic **16**(4), 383–394 (1987)
32. Martinez, M.V., García, A.J., Simari, G.R.: On the use of presumptions in structured defeasible reasoning. In: Proc. of COMMA, pp. 185–196 (2012)
33. Nilsson, N.J.: Probabilistic logic. Artif. Intell. **28**(1), 71–87 (1986)
34. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference (1988)
35. Poole, D.: The independent choice logic for modelling multiple agents under uncertainty. Artif. Intell. **94**(1-2), 7–56 (1997)
36. Rahwan, I., Simari, G.R.: Argumentation in Artificial Intelligence. Springer (2009)
37. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning **62**, 107–136 (2006)
38. Shakarian, P., Shakarian, J., Ruef, A.: Introduction to Cyber-Warfare: A Multidisciplinary Approach. Syngress (2013)
39. Shakarian, P., Simari, G.I., Falappa, M.A.: Belief revision in structured probabilistic argumentation. In: Proc. of FoIKS 2014, pp. 324–343
40. Shakarian, P., Simari, G.I., Moores, G., Parsons, S., Falappa, M.A.: An argumentation-based framework to address the attribution problem in cyber-warfare. In: Proc. of Cyber Security 2014 (2014)

41. Shakarian, P., Simari, G.I., Moores, G., Paulo, D., Parsons, S., Falappa, M.A., Aleali, A.: Belief revision in structured probabilistic argumentation: Model and application to cyber security. Under review – available at: http://www.delp3e.webs.com/Shakarian-etal-DeLP3E.pdf (2014)
42. Simari, G.R., Loui, R.P.: A mathematical treatment of defeasible reasoning and its implementation. Artif. Intell. **53**(2-3), 125–157 (1992)
43. Spohn, W.: The laws of belief: Ranking theory and its philosophical applications. Oxford University Press (2012)
44. Stolzenburg, F., García, A., Chesñevar, C.I., Simari, G.R.: Computing Generalized Specificity. Journal of Non-Classical Logics **13**(1), 87–113 (2003)
45. Thimm, M.: A probabilistic semantics for abstract argumentation. In: Proc. of ECAI 2012, pp. 750–755 (2012)
46. Thimm, M.: Inconsistency measures for probabilistic logics. Artif. Intell. **197**, 1–24 (2013)
47. Toda, S.: On the computational power of PP and $\oplus$P. In: Proc. of FOCS, pp. 514–519 (1989)
48. Wirth, C., Stolzenburg, F.: David Poole's specificity revised. In: Proc. of KR (2014)