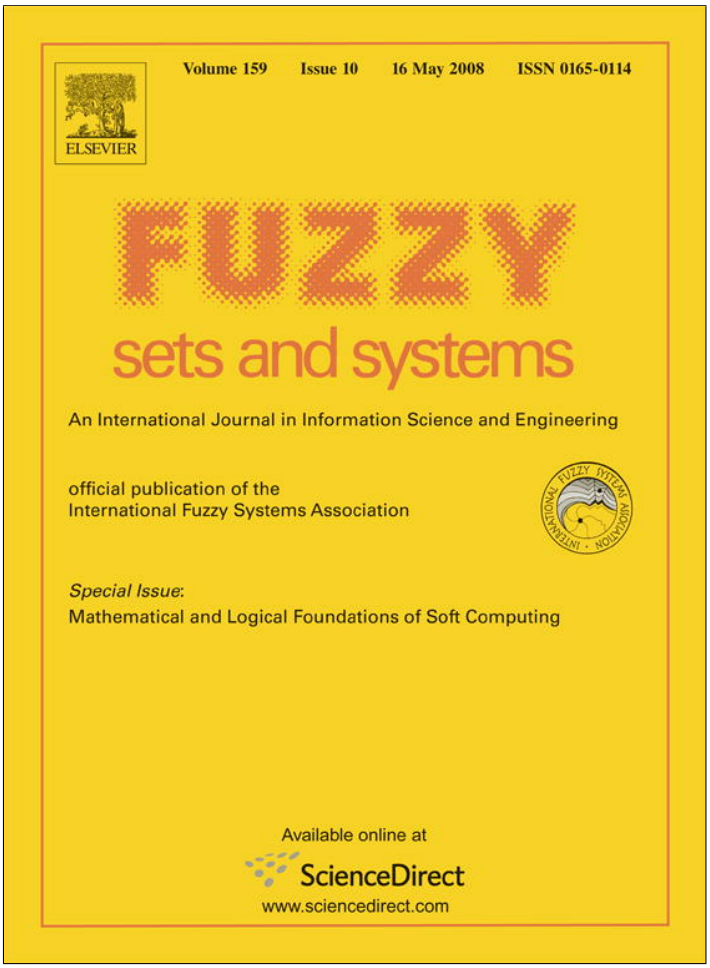

Available online at www.sciencedirect.com
ScienceDirect
Fuzzy Sets and Systems 159 (2008) 1208–1228
FUZZY sets and systems
www.elsevier.com/locate/fss

A logic programming framework for possibilistic argumentation: Formalization and logical properties☆

Teresa Alsinet[a,*], Carlos I. Chesñevar[c], Lluís Godo[b], Guillermo R. Simari[c]

[a]*Department of Computer Science, Universitat de Lleida, C/Jaume II, 69-25001 Lleida, Spain*
[b]*Artificial Intelligence Research Institute (IIIA), CSIC Campus UAB, Bellaterra, Spain*
[c]*Department of Computer Science and Engineering, Universidad Nacional del Sur Av. Alem 1253-8000 Bahía Blanca, Argentina*

Available online 15 December 2007

Abstract

In the last decade defeasible argumentation frameworks have evolved to become a sound setting to formalize commonsense, qualitative reasoning. The logic programming paradigm has shown to be particularly useful for developing different argument-based frameworks on the basis of different variants of logic programming which incorporate defeasible rules. Most of such frameworks, however, are unable to deal with both explicit uncertainty and vague knowledge, as defeasibility is directly encoded in the object language. This paper presents possibilistic defeasible logic programming (P-DeLP), a new logic programming language which combines features from argumentation theory and logic programming, incorporating as well the treatment of possibilistic uncertainty. Such features are formalized on the basis of PGL, a possibilistic logic based on Gödel fuzzy logic. One of the applications of P-DeLP is providing an intelligent agent with non-monotonic, argumentative inference capabilities. In this paper we also provide a better understanding of such capabilities by defining two non-monotonic operators which model the expansion of a given program by adding new weighed facts associated with argument conclusions and warranted literals, respectively. Different logical properties for the proposed operators are studied.

Keywords: Possibilistic logic; Vague knowledge; Defeasible argumentation; Intelligent systems

1. Introduction and motivations

In the last decade defeasible argumentation frameworks [18,35] have evolved to become a sound setting to formalize commonsense, qualitative reasoning from incomplete and potentially inconsistent knowledge. The logic programming paradigm has shown to be particularly useful for developing different argument-based frameworks on the basis of different variants of logic programming which enrich their object language with *defeasible rules* (e.g. [26,7]). Most of such frameworks, however, are unable to deal with both explicit uncertainty and vague knowledge, as defeasibility is directly encoded in the object language.

Possibilistic defeasible logic programming (P-DeLP) is a new logic programming language which combines features from argumentation theory and logic programming, incorporating as well the treatment of possibilistic uncertainty.

☆ This paper revises and extends two previous authors' papers [20,22].

* Corresponding author.
E-mail addresses: tracy@diei.udl.cat (T. Alsinet), cic@cs.uns.edu.ar (C.I. Chesñevar), godo@iiia.csic.es (L. Godo), grs@cs.uns.edu.ar (G.R. Simari).

0165-0114/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.fss.2007.12.013

These knowledge representation features are formalized on the basis of possibilistic Gödel logic (PGL), a possibilistic logic based on Gödel fuzzy logic [2,3]. In PGL formulas are built over fuzzy propositional variables and the certainty degree of formulas is expressed with a necessity measure. In a logic programming setting, the proof method for PGL is based on a complete calculus for determining the maximum degree of possibilistic entailment of a fuzzy goal. In P-DeLP formulas will be supported by arguments, which will have an attached necessity measure associated with the supported conclusion. The ultimate answer to queries will be based on the existence of warranted arguments, computed through a qualitative dialectical analysis. The top-down proof procedure of P-DeLP is based on the one used in *defeasible logic programming* [26,15], which has already been integrated in a number of real-world applications such as intelligent web search [17,19], clustering [27], natural language processing [16] and knowledge management [12], among others.

In the last years, argument-based approaches have proven to be an attractive setting for modelling knowledge and inference in intelligent agents who need to perform defeasible inferences in a computationally effective way [42,32,36]. Given its characteristics, P-DeLP is particularly useful as a tool to achieve this goal, as dealing with uncertainty and fuzziness are useful requirements when formalizing multiagent systems (MAS). We show a case study where P-DeLP is used in the context of formalizing an agent's beliefs and perceptions, along with an argumentative inference procedure to determine which of the agent's beliefs are ultimately accepted (or *warranted*). In order to provide a better understanding of such reasoning capabilities, we also define two non-monotonic *expansion operators* which model the expansion of a given program $\mathcal{P}$ by adding new weighed facts associated with argument conclusions and warranted literals, respectively. Different logical properties for the proposed operators are studied and contrasted with a traditional SLD-based Horn logic. We show that this analysis provides useful comparison criteria that can be extended and applied to other argumentation frameworks.

The rest of the paper is structured as follows. First in Section 2 we will discuss the knowledge representation features provided by P-DeLP, including its syntax and semantics at object-language level. Section 3 presents the central notion of *argument* in P-DeLP as well as an associated procedural mechanism for obtaining them. In Section 4 we formalize the notions of attack among arguments and the top-down proof procedure for computing ultimately undefeated arguments (or *warrants*). Section 5 presents a worked example, showing how P-DeLP can be used to model beliefs and reasoning capabilities of an intelligent agent. Section 6 introduces two non-monotonic *expansion operators* which will allow to analyze the behavior of the P-DeLP framework when a given program is expanded by adding new weighed facts. Section 7 discusses related work, and Section 8 concludes. To make this article self-contained, Appendix A provides a brief summary of the fundamentals of non-monotonic inference relationships and their logical properties.

2. Preliminaries on possibilistic Gödel logic (PGL)

As already pointed out our objective is to formalize P-DeLP, a system that combines features from argumentation theory, logic programming and a unified treatment of possibilistic uncertainty and fuzziness. To achieve this objective we combine two logic programming frameworks: possibilistic logic programming based on infinitely valued propositional Gödel logic [2,3] and defeasible logic programming [26].

In this section we describe the basis of the possibilistic logic programming ingredient of P-DeLP. In a first step, as a suitable logical frame to treat (possibilistic) uncertainty and fuzziness, we consider PGL, a general possibilistic logic on top of propositional Gödel fuzzy logic introduced in [2], and that extends the well-known possibilistic logic (see e.g. [23]), defined on top of classical logic. Then, in order to have at ones disposal an efficient proof procedure, based on a complete calculus and oriented to goals (positive literals), we consider the Horn-rule fragment of PGL as the basis for the definition of a possibilistic logic programming language built up with fuzzy propositional variables [3].

There are three main reasons for choosing PGL as the underlying logic to model both uncertainty and fuzziness. First, it has been proved that many-valued Gödel logic is fully compatible with an already proposed and suitable extension of necessity measures for fuzzy events, in the sense that Gödel logic allows us to define a well-behaved and featured possibilistic semantics on top of it. Second, like in classical (propositional) logic programming systems, PGL enables us to define an efficient proof method by derivation based on a complete calculus for determining the maximum degree of possibilistic belief with which a fuzzy propositional variable can be entailed from a set of formulas. Finally, PGL can be extended with a partial matching mechanism between fuzzy propositional variables based on a necessity-like measure which preserves completeness for a particular class of formulas [3]. This is a key feature that justifies by itself the interest of such a logic programming system for defeasible argumentation under vague knowledge and possibilistic uncertainty.

We provide below a short description of main features of PGL and its Horn-rule fragment.

The *language* $\mathcal{L}_{\mathrm{G}}$ of propositional infinitely valued Gödel logic G is built in the usual way from a (countable) set *Var* of *propositional variables* $\{p, q, \ldots\}$, *connectives* $\wedge$ and $\rightarrow$, and the *truth* constant $\overline{0}$. A negation connective $\neg$ is definable as usual by stipulating that $\neg\varphi$ stands for $\varphi \rightarrow \overline{0}$. A formula of $\mathcal{L}_{\mathrm{G}}$ will be referred as a G-formula. The *semantics* of Gödel fuzzy logic is given by *interpretations* $\mathbf{I}$ of propositional variables from *Var* into the unit interval $[0, 1]$ which are extended to arbitrary formulas by means of the following rules:

$$\mathbf{I}(\overline{0}) = 0,$$

$$\mathbf{I}(\varphi \wedge \psi) = \min(\mathbf{I}(\varphi), \mathbf{I}(\psi)),$$

$$\mathbf{I}(\varphi \rightarrow \psi) = \begin{cases} 1 & \text{if } \mathbf{I}(\varphi) \leqslant \mathbf{I}(\psi), \\ \mathbf{I}(\psi) & \text{otherwise.} \end{cases}$$

Gödel logic G was axiomatized (Hilbert-style) with respect to the given semantics by Dummet in the fifties, but it can be also presented as the axiomatic extension of Hájek's fuzzy logic BL by the additional axiom $\varphi \rightarrow \varphi \wedge \varphi$, forcing the conjunction to be idempotent.

Fuzzy logics, and in particular Gödel logic, are suitable for evaluating the (partial) truth of vague expressions, as in the statement $p =$ "the engine speed is low", in complete states of information. For instance, if we know that in a given situation the speed is 330 r.p.m., and the concept of "low" is modelled by a fuzzy set with membership function $\mu_{\mathrm{low}} : U \rightarrow [0, 1]$, with e.g. $U = [0, 1200]$, then we can measure the truth of p by the value $\mu_{\mathrm{low}}(330)$, whatever this value could be. But when there is incomplete information, one cannot evaluate with certainty the truth of p. In order to allow an explicit representation of uncertainty (on top of the fuzziness), an extension of Gödel logic with possibilistic (meta) semantics, called PGL, was defined in [2]. An expression like

it is almost certain that the engine speed is low

can be represented in the setting of PGL by a *certainty-weighted fuzzy proposition*

$$(\mathit{speed\_low}, 0.9),$$

where the certainty value 0.9 expresses how much the fuzzy statement "the engine speed is low" is believed in terms of necessity measures. In general, certainty weights from $[0, 1]$ are employed in PGL to model statements of the form

φ is certain with a necessity of at least α

where φ is a G-formula which represents vague, incomplete or imprecise knowledge about the real world, and that will be represented by a pair (φ, α).

Definition 1. A *PGL formula* is a pair of the form (φ, α), where φ is a G-formula and $\alpha \in [0, 1]$ is a lower bound on the belief on φ in terms of necessity measures. A PGL-theory will be just a set of PGL formulas.

Within the possibilistic model of uncertainty, belief states are modelled by normalized possibility distributions on a set of Boolean interpretations. However, in our framework, the truth evaluation of a G-formula φ in each interpretation $\mathbf{I}$ is a value $\mathbf{I}(\varphi) \in [0, 1]$, and thus, each formula does not induce a crisp set of interpretations, but a fuzzy set of interpretations $[\varphi]$, defining $\mu_{[\varphi]}(\mathbf{I}) = \mathbf{I}(\varphi)$, for each interpretation $\mathbf{I}$. Therefore, in this setting, possibilistic models are normalized possibility distributions π on the set $\mathcal{I}$ of all possible (many-valued) Gödel interpretations $\mathbf{I}$. Then, to measure the uncertainty induced on a Gödel logic formula φ by a possibilistic model $\pi : \mathcal{I} \rightarrow [0, 1]$, where $\mathcal{I} = \{\mathbf{I} | \mathbf{I}$ is a (many-valued) Gödel interpretation over the set of propositional variables of $\varphi\}$, we have to consider some extension of the notion of necessity measure for fuzzy sets, in particular for fuzzy sets of interpretations. In [24] the authors propose to define

$$N([\varphi]|\pi) = \inf_{\mathbf{I} \in \mathcal{I}} \{\pi(\mathbf{I}) \Rightarrow \mu_{[\varphi]}(\mathbf{I})\},$$

where $\mu_{[\varphi]}(\mathbf{I}) = \mathbf{I}(\varphi) \in [0, 1]$ and $\Rightarrow$ is the reciprocal of Gödel's many-valued implication, which is defined as $x \Rightarrow y = 1$ if $x \leqslant y$ and $x \Rightarrow y = 1 - x$, otherwise. To simplify notation, we will simply write $N(\varphi|\pi)$ instead of

$N([\varphi]|\pi)$, when the set of interpretations $\mathcal{I}$ is fixed from the context. Note that, like in classical possibilistic logic, this necessity measure satisfies that $N(\varphi \wedge \psi|\pi) = \min(N(\varphi|\pi), N(\psi|\pi))$ and $N(\varphi \wedge \neg\varphi|\pi) = 0$.

Next we will present some formal definitions which relate the underlying possibilistic model of PGL in the context of our framework.

Definition 2. Let $\mathcal{I}$ be the set of (many-valued) Gödel interpretations over the set *Var* of propositional variables. A *possibilistic model* is a normalized possibility distribution $\pi : \mathcal{I} \to [0, 1]$ on the set of interpretations $\mathcal{I}$.

A possibility distribution π is normalized when there is at least one $\mathbf{I} \in \mathcal{I}$ such that $\pi(\mathbf{I}) = 1$. In other words, belief states modelled by normalized distributions are consistent belief states, in the sense that at least one interpretation (or state or possible world) has to be fully plausible.

Next we define the notion of possibilistic entailment for PGL.

Definition 3. A possibilistic model $\pi : \mathcal{I} \to [0, 1]$ *satisfies* a PGL formula (φ, α), written $\pi \models (\varphi, \alpha)$, iff $N([\varphi]|\pi) \geqslant \alpha$. We say that a set of PGL formulas K *entails* a PGL formula (φ, α), written $K \models (\varphi, \alpha)$, iff every possibilistic model $\pi : \mathcal{I} \to [0, 1]$ satisfying all the formulas in K also satisfies (φ, α).

When $\pi \models (\psi, \beta)$ for each $(\psi, \beta) \in K$, we say that π is a model of K and that K is satisfiable. Since weights in PGL formulas are understood as lower bounds, given a PGL theory K and a formula φ, we may be interested in knowing the maximum degree with which K possibilistically entails φ.

Definition 4. The *maximum degree of possibilistic entailment* of a formula φ from a PGL theory K, is the value $\|\varphi\|_K = \sup\{\alpha \in [0, 1] | K \models (\varphi, \alpha)\}$.

It is easy to show that $\|\varphi\|_K = \inf\{N(\varphi|\pi) | \pi \models K\}$, i.e. the maximum degree of possibilistic entailment of a G-formula φ from a set of PGL theory K is just the *least necessity evaluation* of φ given by the models of K.

In [2] the authors propose a Hilbert-style axiomatization of PGL, mimicking the axiomatization of (classical) possibilistic logic [23]. Axioms of PGL are axioms of Gödel fuzzy logic weighted by 1 plus the triviality axiom $(\varphi, 0)$, and PGL inference (deduction) rules are a generalized modus ponens rule for necessity measures (from (φ, α) and $(\varphi \to \psi, \beta)$ derive $(\psi, \min(\alpha, \beta))$) and a weight weakening rule (from (φ, α) derive (φ, β), for $\beta \leqslant \alpha$). The notion of proof in PGL is defined as usual relative to the set of axioms and inference rules. In [2] it is shown the soundness of this PGL axiomatic system, leaving the question of whether it is also complete as an open problem.

In order to define a sublanguage suitable for logic programming, that is, enabling a proof algorithm both efficient and complete for computing the maximum degree of possibilistic entailment of atomic propositions (called *goals*), the authors consider the *Horn-rule sublanguage* of PGL. We will refer to this sublanguage as $Horn_{\mathrm{PGL}}$, which consists of PGL formulas of the form

$$(p_1 \wedge \cdots \wedge p_k \to q, \alpha)$$

with $k \geqslant 0$, where $p_1, \ldots, p_k, q$ are propositional variables, in the traditional logic programming style. These weighted Horn rules of formulas will be called PGL *clauses*. As usual, we will refer to the conclusion q and the set of premises $p_1, \ldots, p_k$ as the *head* and the *body* of the rule, respectively. We distinguish between two types of formulas in this sublanguage: *facts* when $k = 0$ (empty body) and are simply written (q, α); and *rules*, written as $(p_1 \wedge \cdots \wedge p_k \to q, \alpha)$ otherwise.

For PGL clauses, a simple and efficient calculus is developed in [2] which does not need the whole logical apparatus of the general possibilistic logic PGL. Moreover, within the restricted framework of $Horn_{\mathrm{PGL}}$ a simple and complete calculus for determining the maximum degree of possibilistic entailment of atomic G-formulas can be defined only by means of the following particular instance of the *generalized modus ponens rule*:

$$\frac{(p_1 \wedge \cdots \wedge p_k \to q, \alpha) \quad (p_1, \beta_1), \ldots, (p_k, \beta_k)}{(q, \min(\alpha, \beta_1, \ldots, \beta_k))} \; [GMP].$$

Formally, we will write $P \vdash_{\text{gmp}} (q, \alpha)$, where P is a set of PGL clauses, $q \in Var$ and $\alpha > 0$, when there exists a finite sequence of PGL clauses $C_1, \ldots, C_m$ such that $C_m = (q, \alpha)$ and, for each $i \in \{1, \ldots, m\}$, either $C_i \in P$ or C_i is obtained by applying the *GMP* rule to previous clauses in the sequence. The corresponding syntactic counterpart of maximum degree of possibilistic entailment is then as follows.

Definition 5. The *maximum degree of deduction* of a goal q from a set of PGL clauses P is $|q|_P = \sup\{\alpha \in [0, 1] | P \vdash_{\text{gmp}} (q, \alpha)\}$.

As the only inference rule of our proof method is the generalized modus ponens, if P is a finite set of PGL clauses, there exists a finite number of proofs of a propositional variable q from P, and thus, the above definition turns into $|q|_P = \max\{\alpha \in [0, 1] | P \vdash_{\text{gmp}} (q, \alpha)\}$. Finally, following [2,3], *completeness* reads as follows: for any finite set of PGL clauses P and any $q \in Var$ it holds that

$$\|q\|_P = |q|_P.$$

Two important consequences of the completeness result are the following ones:

- if $P = \{(p, \beta), (p \to q, \gamma)\}$, $\|q\|_P = \min(\beta, \gamma)$, whenever $p \neq q$; and
- if for some P it holds that $\alpha = \|r\|_P$, then $\|q\|_P = \|q\|_{\{(r,\alpha),(r\to q,\gamma)\}}$, whenever $(r \to q, \gamma) \in P$ and $\|q\|_P > \|q\|_{P\setminus\{(r\to q,\gamma)\}}$.

3. Argumentation in P-DeLP

After recalling in the previous section the main features of the PGL logic and its Horn-rule fragment $Horn_{\text{PGL}}$, we are ready to formalize P-DeLP by extending $Horn_{\text{PGL}}$ with defeasible logic programming argumentative capabilities. In argumentation frameworks, the negation connective allows to identify conflicts among pieces of information, that will be the subject of a deliberative process to determine which one ultimately prevails over the others. Therefore, in order to formalize P-DeLP, we need to introduce in the language a *well-behaved* negation connective $\sim$. By well-behaved we only mean that $\mathbf{I}(q)$ and $\mathbf{I}(\sim q)$ cannot simultaneously take the value 1 for any interpretation $\mathbf{I}$. Remark that in Gödel logic, although the defined negation ($\neg\varphi$ is $\varphi \to \overline{0}$) is well-behaved, it is rather special, since $\neg\varphi$ is always a crisp proposition. Our aim of considering well-behaved negation connectives is that q and $\sim q$ model contradictory information in the following sense.

Proposition 6. *Let Γ be a set of satisfiable PGL clauses. For any well-behaved negation $\sim$, $\Gamma \models (q, \alpha)$ and $\Gamma \models (\sim q, \beta)$ iff either $\alpha = 0$ or $\beta = 0$.*

Proof. For any possibilistic model π such that $\pi \models \Gamma$, it must be that $N(q|\pi) \geqslant \alpha$ and $N(\sim q|\pi) \geqslant \beta$. This means that, for each interpretation $\mathbf{I} \in \mathcal{I}$, $\pi(\mathbf{I}) \Rightarrow \mathbf{I}(q) \geqslant \alpha$ and $\pi(\mathbf{I}) \Rightarrow \mathbf{I}(\sim q) \geqslant \beta$. Thus, $\pi(\mathbf{I}) \Rightarrow \min(\mathbf{I}(q), \mathbf{I}(\sim q)) \geqslant \min(\alpha, \beta)$. Since possibilistic models are normalized possibility distributions, there exists at least one interpretation $\mathbf{I}_0 \in \mathcal{I}$ such that $\pi(\mathbf{I}_0) = 1$, but since $\min(\mathbf{I}(q), \mathbf{I}(\sim q)) \neq 1$ for all interpretation $\mathbf{I} \in \mathcal{I}$, we have that $1 - \pi(\mathbf{I}_0) \geqslant \min(\alpha, \beta)$, and therefore it follows necessarily that $\min(\alpha, \beta) = 0$. □

Because of this observation and the fact that we want to keep the simple calculus of $Horn_{\text{PGL}}$, we adopt the following design decisions:

P-DeLP language: The language of P-DeLP is the one of $Horn_{\text{PGL}}$ but over an extended set Var^* of propositional variables where a new propositional variable "$\sim p$" is added for each $p \in Var$. Note that the symbol $\sim$, although intuitively standing for *negation*, is *not* considered as a proper negation connective. It is only syntactic sugar, as "$\sim p$" is treated just as another propositional variable but with a particular status with respect to p, since it will be only used to detect contradictions at the syntactical level.

Conflict: A set of clauses Γ (in the extended language) will be deemed as *contradictory*, denoted $\Gamma \vdash \bot$, if $\Gamma \vdash_{\text{gmp}} (q, \alpha)$ and $\Gamma \vdash_{\text{gmp}} (\sim q, \beta)$, with $\alpha > 0$ and $\beta > 0$, for some atom $q \in Var$.

From now on we shall refer to positive and negative propositional variables from Var^* as P-DeLP literals. Then, P-DeLP clauses will be just PGL clauses defined over the extended set of variables Var^*, and will be written in the

form $(q \leftarrow p_1 \wedge \cdots \wedge p_k, \alpha)$ more in the style of logic programming languages. The notion of proof for P-DeLP will be therefore the same as for $Horn_{\mathrm{PGL}}$, but will simply write $\vdash$ instead of $\vdash_{\mathrm{gmp}}$.

In order to develop an argumentative framework for P-DeLP, we distinguish between *certain* and *uncertain* clauses. A clause (φ, α) is referred as certain if $\alpha = 1$ and uncertain, otherwise. Moreover, a P-DeLP program is a set of P-DeLP clauses in which we distinguish certain from uncertain information. As additional requirement, certain knowledge is required to be *non-contradictory*. Formally:

Definition 7. A P-DeLP program $\mathcal{P}$ (or just program $\mathcal{P}$) is a pair (Π, Δ), where Π is a non-contradictory finite set of certain clauses, and Δ is a finite set of uncertain clauses.

Next we will introduce the notion of *argument* in P-DeLP. Informally, an argument $\mathcal{A}$ is a tentative proof (as it relies to some extent on uncertain, possibilistic information) from a consistent set of clauses supporting a given conclusion Q with a necessity measure α.

Definition 8. Given a P-DeLP program $\mathcal{P} = (\Pi, \Delta)$, a set $\mathcal{A} \subseteq \Delta$ of uncertain clauses is an *argument* for a goal Q with necessity degree $\alpha > 0$, denoted $\langle \mathcal{A}, Q, \alpha \rangle$, iff:

(1) $\Pi \cup \mathcal{A}$ is non-contradictory;
(2) $\alpha = \max\{\beta \in [0, 1] | \Pi \cup \mathcal{A} \vdash (Q, \beta)\}$, i.e. α is the greatest degree of deduction of Q from $\Pi \cup \mathcal{A}$;
(3) $\mathcal{A}$ is minimal w.r.t. set inclusion, i.e. there is no $\mathcal{A}_1 \subset \mathcal{A}$ such that $\Pi \cup \mathcal{A}_1 \vdash (Q, \alpha)$.

Let $\langle \mathcal{A}, Q, \alpha \rangle$ and $\langle \mathcal{S}, R, \beta \rangle$ be two arguments. We will say that $\langle \mathcal{S}, R, \beta \rangle$ is a *subargument* of $\langle \mathcal{A}, Q, \alpha \rangle$ iff $\mathcal{S} \subseteq \mathcal{A}$. Notice that the goal R may be a subgoal associated with the goal Q in the argument $\mathcal{A}$.

Note that an argument must satisfy certain requirements. First, it should not be the case that $\mathcal{A}$ together with Π turns out to be contradictory. Second, the conclusion (Q, α) should follow from $\Pi \cup \mathcal{A}$ and α is the maximum degree with which this happens. The third requirement operates as a kind of Occam's razor principle [38] on the uncertain information used for concluding Q. It must be remarked that the three conditions in the above definition are inherited from similar definitions in the argumentation literature [38,11,18]. Moreover notice that from the above definition of argument, on the basis of a P-DeLP program $\mathcal{P}$, there may exist *different* arguments $\langle \mathcal{A}_1, Q, \alpha_1 \rangle, \langle \mathcal{A}_2, Q, \alpha_2 \rangle, \ldots, \langle \mathcal{A}_k, Q, \alpha_k \rangle$ supporting a given goal Q, with (possibly) different necessity degrees $\alpha_1, \alpha_2, \ldots, \alpha_k$.

Given a program $\mathcal{P} = (\Pi, \Delta)$, and from a procedural point of view, an argument $\langle \mathcal{A}, Q, \alpha \rangle$ for a given goal Q can be built by computing $\mathcal{A}$ and α through the (recursive) application of the following construction rules:

(1) Building arguments from facts (INTF):
  (a) **If** $(Q, 1) \in \Pi$
  then $\mathcal{A} = \emptyset$ and $\alpha = 1$
  (b) **If** $(Q, \beta) \in \Delta$ and $\Pi \cup \{(Q, \beta)\} \nvdash \bot$ and $\Pi \nvdash (Q, 1)$.
  then $\mathcal{A} = \{(Q, \alpha)\}$ and $\alpha = \beta$
(2) Building arguments from program rules by applying the modus ponens rule (MPA):
  (a) **If** $(Q \leftarrow L_1 \wedge \cdots \wedge L_k, 1) \in \Pi$ and $\langle \mathcal{A}_1, L_1, \beta_1 \rangle, \ldots, \langle \mathcal{A}_k, L_k, \beta_k \rangle$ are arguments and $\Pi \cup \bigcup_{i=1}^{k} \mathcal{A}_i \nvdash \bot$ and there is no $\mathcal{B} \subset \bigcup_{i=1}^{k} \mathcal{A}_i$ such that $\Pi \cup \mathcal{B} \vdash (Q, \gamma)$ with $\gamma \geqslant \min(\beta_1, \beta_2, \ldots, \beta_k)$
  then $\mathcal{A} = \bigcup_{i=1}^{k} \mathcal{A}_i$ and $\alpha = \min(\beta_1, \beta_2, \ldots, \beta_k)$
  (b) **If** $(Q \leftarrow L_1 \wedge L_2 \wedge \cdots \wedge L_k, \beta) \in \Delta$ and $\langle \mathcal{A}_1, L_1, \beta_1 \rangle, \ldots, \langle \mathcal{A}_k, L_k, \beta_k \rangle$ are arguments and $\Pi \cup \{(Q \leftarrow L_1 \wedge L_2 \wedge \cdots \wedge L_k, \beta)\} \cup \bigcup_{i=1}^{k} \mathcal{A}_i \nvdash \bot$ and there is no $\mathcal{B} \subset \bigcup_{i=1}^{k} \mathcal{A}_i \cup (Q \leftarrow L_1 \wedge L_2 \wedge \cdots \wedge L_k, \beta)$ such that $\Pi \cup \mathcal{B} \vdash (Q, \gamma)$ with $\gamma \geqslant \min(\beta, \beta_1, \beta_2, \ldots, \beta_k)$
  then $\mathcal{A} = \bigcup_{i=1}^{k} \mathcal{A}_i \cup (Q \leftarrow L_1 \wedge L_2 \wedge \cdots \wedge L_k, \beta)$ and $\alpha = \min(\beta, \beta_1, \beta_2, \ldots, \beta_k)$

For simplicity, we will write $\mathcal{P} |\sim_{\triangle} \langle \mathcal{A}, Q, \alpha \rangle$ to denote that there is a (finite) sequence of applications of the INTF and MPA rules leading to the argument $\langle \mathcal{A}, Q, \alpha \rangle$.

The basic idea with the argument construction procedure is to keep a trace of the set $\mathcal{A}$ of all uncertain information used to derive a given goal Q with (maximum) necessity degree α. Appropriate preconditions ensure that the obtained argument is in accordance with conditions (1)–(3) in Definition 8. Namely, given a program $\mathcal{P}$, rule INTF allows to construct arguments from facts. An empty argument can be obtained for any certain fact in $\mathcal{P}$. An argument concluding

an uncertain fact (Q, α) in $\mathcal{P}$ can be derived whenever assuming (Q, α) is not contradictory w.r.t. the set Π in $\mathcal{P}$ and that Q cannot be proved from Π with a necessity degree greater or equal than α. Rule MPA accounts for the use of modus ponens, both with certain and uncertain rules. It assumes the existence of an argument for every literal in the antecedent of the rule. Then, in such a case, the MPA rule is applicable whenever no contradiction results when putting together Π, the sets $\mathcal{A}_1, \ldots, \mathcal{A}_k$ corresponding to the arguments for the antecedents of the rule and the rule $(Q \leftarrow L_1 \wedge \cdots \wedge L_k, \beta)$ when $\beta < 1$, and whenever it is strictly necessary to consider all these clauses in order to prove Q with a greater necessity degree.

The completeness of the INTF and MPA rules for constructing arguments is formally stated in the next proposition.

Proposition 9. *Let $\mathcal{P} = (\Pi, \Delta)$ be a P-DeLP program, let $\mathcal{A} \subseteq \Delta$ be a set of uncertain clauses, let Q be a literal, and let $\alpha \in (0, 1]$. Then, $\langle \mathcal{A}, Q, \alpha \rangle$ is an argument if, and only if, $\mathcal{P} |\sim_{\triangle} \langle \mathcal{A}, Q, \alpha \rangle$.*

Proof. One direction is easy since, by definition, rules INTF and MPA yield arguments as conclusions (notice that the pre-conditions of the rules ensure this), hence if $\mathcal{P} |\sim_{\triangle} \langle \mathcal{A}, Q, \alpha \rangle$ then $\langle \mathcal{A}, Q, \alpha \rangle$ is an argument. Conversely, let us assume that $\langle \mathcal{A}, Q, \alpha \rangle$ is an argument. In particular then α is the greatest degree of deduction of Q from $\Pi \cup \mathcal{A}$, that is, $\Pi \cup \mathcal{A} \vdash (Q, \alpha)$ and hence there is a proof κ of (Q, α) from $\Pi \cup \mathcal{A}$ using only the GMP rule. Moreover, we can assume that the proof consists of sequence of clauses

$$\kappa := C_1, \ldots, C_m, C_{m+1},$$

where $C_i = (R_i, \beta_i)$ for $1 \leqslant i \leqslant k$ and $C_{m+1} = (Q, \alpha)$, such that:

- as usual, each clause C_i either belongs to $\Pi \cup \mathcal{A}$ or it has been obtained from previous ones by the application of the GMP rule,
- β_i is the maximum degree of deduction of R_i from $\Pi \cup \mathcal{A}$,
- the sequence cannot be simplified, in the sense that all clauses C_i are necessary.

For each $i = 1, \ldots, k$, let the set $\mathcal{A}_i$ be a minimal subset of $\mathcal{A}$ such that $\Pi \cup \mathcal{A}_i \vdash C_i$. Then $A_i = \langle \mathcal{A}_i, R_i, \beta_i \rangle$ is an argument. In particular, by this construction, one has $A_{m+1} = \langle \mathcal{A}, Q, \alpha \rangle$. And it is clear that then:

- for each clause $C_i = (R_i, \beta_i) \in \Delta$, the argument A_i can be built by the INTF rule;
- for each application of the GMP rule in the proof κ to derive a clause C_i $(1 \leqslant i \leqslant m + 1)$ from clauses $C_{i_1}, \ldots, C_{i_j}$ one can build the argument A_i from arguments $A_{i_1}, \ldots, A_{i_j}$ by the MPA rule.

Therefore, in this way we have shown that $\langle \mathcal{A}, Q, \alpha \rangle$ can be built by applications of the INTF and MPA rules, and hence that $\mathcal{P} |\sim_{\triangle} \langle \mathcal{A}, Q, \alpha \rangle$. □

4. Computing warrant in P-DeLP

Given a program and a particular context, it can be the case that there exist arguments supporting contradictory literals. This is formalized next by the notions of counterargument and defeat. In what follows, for a given goal Q, we will write $\sim Q$ as an abbreviation to denote "$\sim q$" if $Q \equiv q$ and "q" if $Q \equiv \sim q$.

Definition 10. Let $\mathcal{P}$ be a P-DeLP program, and let $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ and $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$ be two arguments w.r.t. $\mathcal{P}$. We will say that $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ counterargues $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$ iff there exists a subargument (called *disagreement subargument*) $\langle \mathcal{S}, Q, \beta \rangle$ of $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$ such that $Q_1 = \sim Q$.

Since arguments rely on uncertain and hence defeasible information, conflicts among arguments may be resolved by comparing their strength. Therefore, a notion of defeat amounts to establish a *preference criterion* on conflicting arguments. In our framework, it seems natural to define it on the basis of necessity degrees associated with arguments [20].

Definition 11. Let $\mathcal{P}$ be a P-DeLP program and let the argument $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ counterargue the argument $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$ with disagreement subargument $\langle \mathcal{A}, Q, \beta \rangle$. We say that $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ is a *proper* (respectively *blocking*) *defeater* for $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$ when $\alpha_1 > \beta$ (respectively $\alpha_1 = \beta$).

As already mentioned, argument-based inference involves a dialectical process in which arguments are compared in order to determine which beliefs are ultimately accepted (or *warranted*) on the basis of a given knowledge base. In the case of argument-based logic programming, such knowledge base is given by the underlying logic program (in our case, a P-DeLP program). Skeptical argument-based semantics [25,35] are commonly used for computing warranted arguments. The intuition behind such skeptical approaches to the notion of warrant can be defined as follows:

(1) An argument $\langle \mathcal{A}, Q, \alpha \rangle$ is warranted if $\langle \mathcal{A}, Q, \alpha \rangle$ has no defeaters.
(2) An argument $\langle \mathcal{A}, Q, \alpha \rangle$ is warranted if it has defeaters $\langle \mathcal{B}_1, Q_1, \beta_1 \rangle, \ldots, \langle \mathcal{B}_k, Q_k, \beta_k \rangle$, such that every defeater $\langle \mathcal{B}_i, Q_i, \beta_i \rangle$, $(1 \leqslant i \leqslant k)$ is in turn defeated by a warranted argument.

In P-DeLP, as in most argumentation systems [18,35], this intuition is formalized in terms of an exhaustive dialectical analysis of all possible argumentation lines rooted in a given argument. An *argumentation line* starting in an argument $\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle$ is a sequence of arguments $\lambda = [\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle, \langle \mathcal{A}_1, Q_1, \alpha_1 \rangle, \ldots, \langle \mathcal{A}_n, Q_n, \alpha_n \rangle, \ldots]$ that can be thought of as an exchange of arguments between two parties, a *proponent* (evenly indexed arguments) and an *opponent* (oddly indexed arguments). Each $\langle \mathcal{A}_i, Q_i, \alpha_i \rangle$ is a defeater for the previous argument $\langle \mathcal{A}_{i-1}, Q_{i-1}, \alpha_{i-1} \rangle$ in the sequence, $i > 0$. Moreover, in order to avoid *fallacious* reasoning, argumentation theory imposes three additional constraints on such an argument exchange to be considered rationally acceptable w.r.t. a P-DeLP program $\mathcal{P}$, namely:

(1) **Non-contradiction**: Given an argumentation line λ, the set of arguments of the proponent (respectively opponent) should be *non-contradictory* w.r.t. $\mathcal{P}$. [1]
(2) **Progressive argumentation**: (i) Every blocking defeater $\langle \mathcal{A}_i, Q_i, \alpha_i \rangle$ in λ with $i > 0$ is defeated by a proper defeater [2] $\langle \mathcal{A}_{i+1}, Q_{i+1}, \alpha_{i+1} \rangle$ in λ; and (ii) each argument $\langle \mathcal{A}_i, Q_i, \alpha_i \rangle$ in λ, with $i \geqslant 2$, is such that $Q_i \neq \sim Q_{i-1}$.

The non-contradiction condition disallows the use of contradictory information on either side (proponent or opponent). The first condition of progressive argumentation enforces the use of a proper defeater to defeat an argument which acts as a blocking defeater, while the second condition avoids non-optimal arguments in the presence of a conflict. Indeed, if we had a sequence of successively defeated arguments of the form

$$\lambda = [\ldots, \langle \mathcal{A}_i, Q, \alpha_i \rangle, \langle \mathcal{A}_{i+1}, \sim Q, \alpha_{i+1} \rangle, \langle \mathcal{A}_{i+2}, Q, \alpha_{i+2} \rangle, \ldots],$$

it would mean that $\langle \mathcal{A}_i, Q, \alpha_i \rangle$ could have been in fact replaced by a stronger argument taking into the information in $\langle \mathcal{A}_{i+2}, Q, \alpha_{i+2} \rangle$. An argumentation line satisfying the above restrictions is called *acceptable*, and can be proven to be finite. Given a program $\mathcal{P}$ and an argument $\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle$, the set of all acceptable argumentation lines starting in $\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle$ accounts for a whole dialectical analysis for $\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle$ (i.e. all possible dialogues rooted in $\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle$) that leads to the following definition of warranted goal with a necessity degree. This set is usually represented in a tree structure called *dialectical tree* for the argument $\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle$. In this dialectical tree every branch corresponds to a possible dialogue starting with $\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle$.

Definition 12. Given a program $\mathcal{P} = (\Pi, \Delta)$ and a goal Q, we will say that Q is *warranted* w.r.t. $\mathcal{P}$ with a *maximum necessity degree* α iff there exists an argument $\langle \mathcal{A}, Q, \alpha \rangle$, for some $\mathcal{A} \subseteq \Delta$, such that:

(1) every acceptable argumentation line starting with $\langle \mathcal{A}, Q, \alpha \rangle$ has an odd number of arguments; i.e. every argumentation line starting with $\langle \mathcal{A}, Q, \alpha \rangle$ finishes with an argument proposed by the proponent which is in favor of Q with at least a necessity degree α; and
(2) there is no other argument of the form $\langle \mathcal{A}_1, Q, \beta \rangle$, with $\beta > \alpha$, satisfying the above.

We will generalize the use of the term "warranted" for applying it to both goals and arguments: whenever a goal Q is warranted on the basis of a given argument $\langle \mathcal{A}, Q, \alpha \rangle$ as specified in Definition 12, we will also say that the argument $\langle \mathcal{A}, Q, \alpha \rangle$ is warranted. Moreover, we will write $\mathcal{P} |\!\sim_w \langle \mathcal{A}, Q, \alpha \rangle$ to denote that the argument $\langle \mathcal{A}, Q, \alpha \rangle$ is warranted w.r.t. $\mathcal{P}$.

[1] Non-contradiction for a set of arguments is defined as follows: a set $S = \bigcup_{i=1}^{n} \{\langle \mathcal{A}_i, Q_i, \alpha_i \rangle\}$ is *contradictory* w.r.t. $\mathcal{P}$ iff $\Pi \cup \bigcup_{i=1}^{n} \mathcal{A}_i$ is contradictory.

[2] It must be noted that the last argument in an argumentation line is allowed to be a blocking defeater for the previous one.

For a given program $\mathcal{P}$, a P-DeLP interpreter will find an answer for a goal Q by determining whether Q is supported by some warranted argument $\langle \mathcal{A}, Q, \alpha \rangle$. Different doxastic attitudes are distinguished when providing an answer for the goal Q (according to the associated status of warrant), and additional refinements have been also studied to speed up the inference procedure. An in-depth discussion of these aspects can be found elsewhere [21].

5. A case study: modelling an intelligent agent in P-DeLP

Next we will present an example which illustrates how P-DeLP can be used to model the beliefs and reasoning capabilities of an agent. Consider an intelligent agent controlling an engine with three switches $sw1$, $sw2$ and $sw3$. These switches regulate different features of the engine, such as pumping system, speed, etc. This agent may have the following certain and uncertain knowledge about how this engine works, e.g.:

(1) When $sw1$ is on, normally fuel is pumped properly.
(2) When fuel is pumped properly, fuel seems to work ok.
(3) When $sw2$ is on, usually oil is pumped.
(4) When oil is pumped, usually it works ok.
(5) When there is oil and fuel, usually the engine works ok.
(6) When there is heat, then the engine is usually not ok.
(7) When there is heat, normally there are oil problems.
(8) When fuel is pumped and speed is low, then there are reasons to believe that the pump is clogged.
(9) When $sw2$ is on, usually speed is low.
(10) When $sw2$ and $sw3$ are on, usually speed is not low.
(11) When $sw3$ is on, usually fuel is ok.
(12) If the pump is clogged, then the engine gets no fuel.

Suppose also that the agent knows some particular facts: $sw1$, $sw2$ and $sw3$ are on, and there is heat. The knowledge of such an agent can be modelled by the program $\mathcal{P}_{\text{eng}}$ shown in Fig. 1, where the finite set of certain clauses (i.e. Π) is from clause (12) to (16), and the finite set of uncertain clauses (i.e. Δ) is from clause (1) to (11). Note that uncertainty is assessed in terms of different necessity measures. From the P-DeLP program in Fig. 1 different *arguments* can be derived using the procedural rules defined in Section 3. Thus, for example, the argument $\langle \mathcal{B}, \mathit{fuel\_ok}, 0.3 \rangle$ can

$$
\begin{aligned}
&(1) && (\mathit{pump\_fuel} \leftarrow \mathit{sw1}, 0.6)\\
&(2) && (\mathit{fuel\_ok} \leftarrow \mathit{pump\_fuel}, 0.3)\\
&(3) && (\mathit{pump\_oil} \leftarrow \mathit{sw2}, 0.8)\\
&(4) && (\mathit{oil\_ok} \leftarrow \mathit{pump\_oil}, 0.8)\\
&(5) && (\mathit{engine\_ok} \leftarrow \mathit{fuel\_ok} \wedge \mathit{oil\_ok}, 0.3)\\
&(6) && (\sim \mathit{engine\_ok} \leftarrow \mathit{heat}, 0.95)\\
&(7) && (\sim \mathit{oil\_ok} \leftarrow \mathit{heat}, 0.9)\\
&(8) && (\mathit{pump\_clog} \leftarrow \mathit{pump\_fuel} \wedge \mathit{low\_speed}, 0.7)\\
&(9) && (\mathit{low\_speed} \leftarrow \mathit{sw2}, 0.8)\\
&(10) && (\sim \mathit{low\_speed} \leftarrow \mathit{sw2} \wedge \mathit{sw3}, 0.8)\\
&(11) && (\mathit{fuel\_ok} \leftarrow \mathit{sw3}, 0.6)\\
&(12) && (\sim \mathit{fuel\_ok} \leftarrow \mathit{pump\_clog}, 1)\\
&(13) && (sw1, 1)\\
&(14) && (sw2, 1)\\
&(15) && (sw3, 1)\\
&(16) && (\mathit{heat}, 1)
\end{aligned}
$$

Fig. 1. P-DeLP program $\mathcal{P}_{\text{eng}}$.

be derived from $\mathcal{P}_{\text{eng}}$ as follows:

(i) $\langle \emptyset, sw1, 1\rangle$ from (13) via INTF.
(ii) $\langle \mathcal{B}', pump\_fuel, 0.6\rangle$ from (1) and (i) via MPA.
(iii) $\langle \mathcal{B}, fuel\_ok, 0.3\rangle$ from (2) and (ii) via MPA.

Where

$$\mathcal{B}' = \{(pump\_fuel \leftarrow sw1, 0.6)\}$$

and

$$\mathcal{B} = \{(pump\_fuel \leftarrow sw1, 0.6);\ (fuel\_ok \leftarrow pump\_fuel, 0.3)\}.$$

Similarly, an argument $\langle \mathcal{C}, oil\_ok, 0.8\rangle$ can be derived from $\mathcal{P}_{\text{eng}}$ using the rules (14), (3) and (4) via INTF, MPA, and MPA, respectively, with [3]

$$\mathcal{C} = \{(pump\_oil \leftarrow sw2, 0.8); (oil\_ok \leftarrow pump\_oil, 0.8)\}.$$

Finally, an argument $\langle \mathcal{A}_1, engine\_ok, 0.3\rangle$ can be derived from $\mathcal{P}_{\text{eng}}$ as follows:

(i) $\langle \mathcal{B}, fuel\_ok, 0.3\rangle$ as shown above.
(ii) $\langle \mathcal{C}, oil\_ok, 0.8\rangle$ as shown above.
(iii) $\langle \mathcal{A}_1, engine\_ok, 0.3\rangle$ from (i), (ii) and (5) via MPA,

where $\mathcal{A}_1 = \{(engine\_ok \leftarrow fuel\_ok \wedge oil\_ok, 0.3)\} \cup \mathcal{B} \cup \mathcal{C}$. Note that the arguments $\langle \mathcal{C}, oil\_ok, 0.8\rangle$ and $\langle \mathcal{B}, fuel\_ok, 0.3\rangle$ are subarguments of the argument $\langle \mathcal{A}_1, engine\_ok, 0.3\rangle$ (see Definition 8).

Let us assume that the agent is in charge of controlling the engine, answering queries from other agents (e.g. a supervisor agent) about the status of the engine. For instance, the query $? - (engine\_ok, 0.8)$ corresponds with proving whether the engine works ok with a certainty degree of at least 0.8. In order to answer this query, the agent will apply the procedure described in the previous sections: first it will compute an argument supporting *engine_ok*, and then will perform a recursive analysis of defeaters for these arguments, computing its associated dialectical tree.

In this particular example, the agent will find an argument supporting the conclusion *engine_ok*, namely $\langle \mathcal{A}_1, engine\_ok, 0.3\rangle$. A counterargument (see Definition 10) for the argument $\langle \mathcal{A}_1, engine\_ok, 0.3\rangle$ can be found, namely the argument $\langle \mathcal{A}_2, \sim fuel\_ok, 0.6\rangle$, obtained from (13), (14), (1), (9), (8) and (12) by applying INTF, INTF, MPA, MPA, MPA, and MPA, respectively, with

$$\begin{aligned}\mathcal{A}_2 = \{&(pump\_fuel \leftarrow sw1, 0.6),\\ &(low\_speed \leftarrow sw2, 0.8),\\ &(pump\_clog \leftarrow pump\_fuel \wedge low\_speed, 0.7)\}.\end{aligned}$$

The argument $\langle \mathcal{A}_2, \sim fuel\_ok, 0.6\rangle$ is a counterargument for the argument $\langle \mathcal{A}_1, engine\_ok, 0.3\rangle$ as there exists a subargument $\langle \mathcal{B}, fuel\_ok, 0.3\rangle$ associated with argument $\langle \mathcal{A}_1, engine\_ok, 0.3\rangle$ such that the set

$$\Pi \cup \{(fuel\_ok, 0.3), (\sim fuel\_ok, 0.6)\}$$

is contradictory. It must be remarked as well that $\langle \mathcal{A}_2, \sim fuel\_ok, 0.6\rangle$ is a proper defeater (Definition 11) for $\langle \mathcal{A}_1, engine\_ok, 0.3\rangle$, as $\langle \mathcal{A}_2, \sim fuel\_ok, 0.6\rangle$ counterargues the argument $\langle \mathcal{A}_1, engine\_ok, 0.3\rangle$ with $\langle \mathcal{B}, fuel\_ok, 0.3\rangle$ as disagreement subargument, and $0.6 > 0.3$.

As the defeater $\langle \mathcal{A}_2, \sim fuel\_ok, 0.6\rangle$ is also an argument, a recursive analysis can be carried out by the agent, computing an *argumentation line* rooted in $\langle \mathcal{A}_1, engine\_ok, 0.3\rangle$. In fact, note that the argument $\langle \mathcal{A}_2, \sim fuel\_ok, 0.6\rangle$ has the subargument $\langle \mathcal{A}_2', low\_speed, 0.8\rangle$, with $\mathcal{A}_2' = \{(low\_speed \leftarrow sw2, 0.8)\}$. From the program $\mathcal{P}_{\text{eng}}$ given in Fig. 1 a blocking defeater for the argument $\langle \mathcal{A}_2, \sim fuel\_ok, 0.6\rangle$ can be derived, namely $\langle \mathcal{A}_3, \sim low\_speed, 0.8\rangle$, obtained from (14), (15) and (10) via INTF, INTF and MPA, respectively. In this case we have

$$\mathcal{A}_3 = \{(\sim low\_speed \leftarrow sw2 \wedge sw3, 0.8)\}.$$

[3] For the sake of clarity, we use semicolons to separate elements in an argument $\mathcal{A} = \{e_1, e_2, \ldots, e_k\}$.

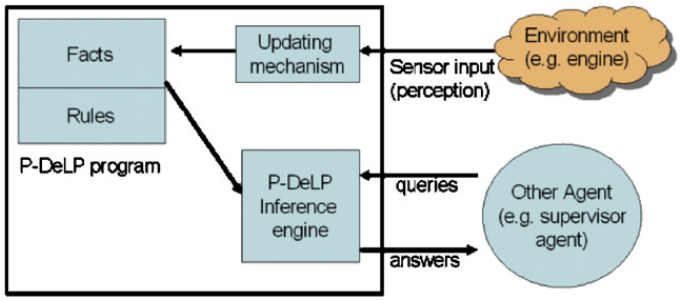

Fig. 2. A P-DeLP-based agent in an MAS context.

This third defeater $\langle \mathcal{A}_3, \sim low\_speed, 0.8\rangle$ can be thought of as an answer of the proponent to the opponent which reinstates $\langle \mathcal{A}_1, engine\_ok, 0.3\rangle$, as it defeats the opponent's defeater $\langle \mathcal{A}_2, \sim fuel\_ok, 0.6\rangle$. The above situation can be expressed in the following argumentation line [4]:

$$[\langle \mathcal{A}_1, engine\_ok, 0.3\rangle, \langle \mathcal{A}_2, \sim fuel\_ok, 0.6\rangle, \langle \mathcal{A}_3, \sim low\_speed, 0.8\rangle].$$

In order for the preceding analysis to be exhaustive, every possible argumentation line rooted in $\langle \mathcal{A}_1, engine\_ok, 0.3\rangle$ should be analyzed. In this particular case, note that the argument $\langle \mathcal{A}_2, \sim fuel\_ok, 0.6\rangle$ has a second (blocking) defeater $\langle \mathcal{A}_4, fuel\_ok, 0.6\rangle$, computed from (15), (11) via INTF and MPA, respectively. The argument $\langle \mathcal{A}_1, engine\_ok, 0.3\rangle$ has also a second defeater $\langle \mathcal{A}_5, \sim engine\_ok, 0.95\rangle$, computed from (16), (6) via INTF and MPA, respectively. In this case we have

$$\mathcal{A}_4 = \{(fuel\_ok \leftarrow sw3, 0.9)\},$$

$$\mathcal{A}_5 = \{(\sim engine\_ok \leftarrow heat, 0.8)\}.$$

There are no more arguments to consider. As a consequence, there are three acceptable argumentation lines rooted in $\langle \mathcal{A}_1, engine\_ok, 0.3\rangle$, namely:

- $\lambda_1 = [\langle \mathcal{A}_1, engine\_ok, 0.3\rangle, \langle \mathcal{A}_2, \sim fuel\_ok, 0.6\rangle, \langle \mathcal{A}_3, \sim low\_speed, 0.8\rangle]$.
- $\lambda_2 = [\langle \mathcal{A}_1, engine\_ok, 0.3\rangle, \langle \mathcal{A}_2, \sim fuel\_ok, 0.6\rangle, \langle \mathcal{A}_4, fuel\_ok, 0.6\rangle]$.
- $\lambda_3 = [\langle \mathcal{A}_1, engine\_ok, 0.3\rangle, \langle \mathcal{A}_5, \sim engine\_ok, 0.95\rangle]$.

The argument $\langle \mathcal{A}_1, engine\_ok, 0.3\rangle$ is the only possible argument the agent can compute supporting the query $engine\_ok$. There are three argumentation lines rooted in the argument $\langle \mathcal{A}_1, engine\_ok, 0.3\rangle$, and one of them is of even length. Therefore the argument $\langle \mathcal{A}_1, engine\_ok, 0.3\rangle$ is not warranted (Definition 12). On the contrary, note that the agent can compute an argument $\langle \mathcal{A}_5, \sim engine\_ok, 0.95\rangle$ supporting $\sim engine\_ok$, and such argument has no defeaters. Consequently, there is only one argumentation line rooted in $\langle \mathcal{A}_5, \sim engine\_ok, 0.95\rangle$, namely $\lambda = [\langle \mathcal{A}_5, \sim engine\_ok, 0.95\rangle]$, which has odd length. Therefore we can conclude that $\langle \mathcal{A}_5, \sim engine\_ok, 0.95\rangle$ is warranted.

In a MAS context, intelligent agents will encode their knowledge about the world using a P-DeLP program. Fig. 2 outlines the different elements associated with a P-DeLP-based agent. Clearly, our agent will be usually performing its activities in a dynamic environment, so that it should also be able to reason, plan, and act according to new perceptions from the outside world. Such perceptions will be sensed by the agent, integrating them into its current beliefs. For the sake of simplicity, we will assume that such perceptions constitute new facts to be added to the agent's knowledge base. As already stated in the Introduction, fuzzy propositions provide us with a suitable representation model as our agent will probably have vague or imprecise information about the real world, as its sensors are not perfect devices.

---

[4] Note that the proponent's last defeater in the above sequence could be on its turn defeated by a blocking defeater $\langle \mathcal{A}_2', low\_speed, 0.8\rangle$, resulting in

$$[\langle \mathcal{A}_1, engine\_ok, 0.3\rangle, \langle \mathcal{A}_2, \sim fuel\_ok, 0.6\rangle, \langle \mathcal{A}_3, \sim low\_speed, 0.8\rangle, \langle \mathcal{A}_2', low\_speed, 0.8\rangle \ldots].$$

However, such line is *not acceptable*, as it violates the condition of non-circular argumentation.

Defining a generic procedure for updating the agent's knowledge base is not easy, as completely new incoming information (e.g. facts with new predicate names) might result in the strict knowledge Π becoming contradictory (see Definition 7). In some particular cases the agent will only perceive changes in the necessity measure of the already known facts (e.g. the agent has a fact $(heat, 1)$ in the knowledge base, but the sensed temperature has changed, modelled by a new fact $(heat, 0.8)$). In such cases, a simple but effective strategy can be applied, similar as the one suggested in [14]. We will make the assumption that new perceptions always supersede old ones, so that if a new perception $(p, value)$ is sensed at time t, and the agent has already a fact $(p, value')$ in its strict knowledge base Π, then the updated knowledge base will be computed as $(\Pi \setminus \{(p, value')\}) \cup \{(p, value)\}$.

6. Logical properties of argument and warrant in P-DeLP

As explained in the previous section, intelligent agents can encode their knowledge about the world in terms of a P-DeLP program $\mathcal{P}$. New perceptions can modify the agent's current knowledge, altering the set of facts in $\mathcal{P}$. In this context, it is interesting to analyze the existing inference abilities of our agent in terms of deduction, arguments and warrants.

In order to do this, we will define different inference operators associated with deduction based on strict knowledge, argument derivation and computation of warranted goals. We refer to such operators as *expansion operators* in order to stress the fact that their output is associated with considering all new weighted facts that can be obtained by computing deductions based on strict knowledge, arguments or warrants from a given program $\mathcal{P}$. Our aim is to study the inference abilities of a P-DeLP agent in terms of the logical properties associated with these operators.

For our characterization, we will identify three distinguished sets $Lit_{\vdash}$, $Lit_{\triangle}$, and Lit_w associated with any program $\mathcal{P}$, corresponding to those literals which are supported by deduction from strict knowledge, supported by arguments, and supported by warranted arguments, respectively. Formally:

Definition 13. Let $\mathcal{P}$ be a P-DeLP program. We define the sets $Lit_{\vdash}(\mathcal{P})$, $Lit_{\triangle}(\mathcal{P})$, and $Lit_w(\mathcal{P})$ associated with $\mathcal{P}$ as follows:

$$Lit_{\vdash}(\mathcal{P}) = \{(Q, 1) | \mathcal{P} \vdash (Q, 1)\},$$

$$Lit_{\triangle}(\mathcal{P}) = \{(Q, \alpha) | \mathcal{P} \mid\!\sim_{\triangle} \langle \mathcal{A}, Q, \alpha\rangle \text{ for some argument } \mathcal{A} \text{ for a goal } Q \text{ with necessity degree } \alpha\},$$

$$Lit_w(\mathcal{P}) = \{(Q, \alpha) | \mathcal{P} \mid\!\sim_{w} \langle \mathcal{A}, Q, \alpha\rangle \text{ for some argument } \mathcal{A} \text{ for a goal } Q \text{ with necessity degree } \alpha\}.$$

On the basis of these distinguished sets, we will define three expansion operators which, given a program $\mathcal{P}$, compute a new P-DeLP program $\mathcal{P}'$ whose facts are precisely those given by the three distinguished sets given in Definition 13, while the existing rules in $\mathcal{P}$ are maintained.

Definition 14. Let $\mathcal{P}$ be a P-DeLP program. We define the *expansion operators* $C_{\vdash}$, $C_{\triangle}$ and C_w associated with $\mathcal{P}$ as follows:

$$C_{\vdash}(\mathcal{P}) = rules(\mathcal{P}) \cup Lit_{\vdash}(\mathcal{P}),$$

$$C_{\triangle}(\mathcal{P}) = rules(\mathcal{P}) \cup Lit_{\triangle}(\mathcal{P}),$$

$$C_w(\mathcal{P}) = rules(\mathcal{P}) \cup Lit_w(\mathcal{P}).$$

Operator $C_{\vdash}$ computes a new program $\mathcal{P}'$ based on the rules of $\mathcal{P}$ along with those facts $(Q, 1)$ which can be deduced from the strict knowledge in $\mathcal{P}$.[5] Operator $C_{\triangle}$ computes a new program $\mathcal{P}'$ based on the rules of $\mathcal{P}$ along with those facts which are supported by arguments based on $\mathcal{P}$. The output $\mathcal{P}'$ incorporates a new uncertain fact (Q, α) whenever there exists an argument $\langle \mathcal{A}, Q, \alpha\rangle$ in $\mathcal{P}$. Finally, operator C_w computes a subset of $C_{\triangle}$, namely the program resulting from the

[5] Operator $C_{\vdash}$ defines in fact a consequence relationship, as it satisfies idempotence, cut and monotonicity. It can be seen as the SLD Horn resolution counterpart in the context of P-DeLP restricted to certain clauses.

rules in $\mathcal{P}$ along with all those new facts which correspond to conclusions of warranted arguments in $\mathcal{P}$. Notice that $C_\triangle$ may contain contradictory knowledge (i.e. it may be the case that two arguments $\langle \mathcal{A}_1, Q, \alpha\rangle$ and $\langle \mathcal{A}_2, \sim Q, \beta\rangle$ could be inferred from a given program $\mathcal{P}$).

Proposition 15. *Operators $C_\vdash$, $C_\triangle$ and C_w are well-defined* (*i.e. given a P-DeLP program $\mathcal{P}$ as input, the associated output is also a P-DeLP program $\mathcal{P}'$*). *Besides, they satisfy the following relationship*: $C_\vdash(\mathcal{P}) \subseteq C_w(\mathcal{P}) \subseteq C_\triangle(\mathcal{P})$.

Proof. Given a P-DeLP program $\mathcal{P} = (\Pi, \Delta)$, we want to determine that $C_\vdash(\mathcal{P})$, $C_\triangle(\mathcal{P})$ and $C_w(\mathcal{P})$ are also programs. From Definitions 13 and 14, it is clear that all operators return syntactically valid programs as their output, as $Lit_\vdash$, $Lit_\triangle$ and Lit_w correspond always to facts in P-DeLP. From Definition 7, it remains to check that the strict knowledge of the output of $C_\vdash(\mathcal{P})$ (analogously for $C_\triangle(\mathcal{P})$ and $C_w(\mathcal{P})$), written $C_\vdash(\mathcal{P})^\Pi$, is not a contradictory set of P-DeLP clauses. Let us suppose that $C_\vdash(\mathcal{P})^\Pi$ is contradictory. By definition of $C_\vdash$, this is only possible if Π is itself contradictory, which cannot be the case, as $\mathcal{P}$ is a P-DeLP program (absurd). Consequently, $C_\vdash(\mathcal{P})$ is a P-DeLP program. The same line of reasoning applies for $C_\triangle(\mathcal{P})$ and $C_w(\mathcal{P})$.

The inclusion relationship $C_\vdash(\mathcal{P}) \subseteq C_\triangle(\mathcal{P})$ holds as it is straightforward to see that $\mathcal{P}\vdash(Q, 1)$ iff $\mathcal{P}|\!\sim_\triangle \langle\emptyset, Q, 1\rangle$. Since every warranted argument is an argument w.r.t. $\mathcal{P}$, a similar analysis applies to conclude that $C_w(\mathcal{P}) \subseteq C_\triangle(\mathcal{P})$. □

Next we will analyze different logical properties for deduction from strict knowledge, argument construction and warrant in P-DeLP, on the basis of the operators defined before. A summary of the logical properties under consideration can be found in Appendix A.

6.1. Logical properties for $C_\triangle$

Clearly inclusion does not hold for $C_\triangle$. A counterexample suffices; consider $\mathcal{P} = \{(p, 0.5), (\sim p, 1)\}$. Then $C_\triangle(\mathcal{P}) = \{(\sim p, 1)\}$. Hence $\mathcal{P} \not\subseteq C_\triangle(\mathcal{P})$.

Proposition 16. *The operator $C_\triangle$ satisfies idempotence, i.e. $C_\triangle(\mathcal{P}) = C_\triangle(C_\triangle(\mathcal{P}))$.*

Proof (*By double inclusion*). (1) By definition of $C_\triangle$ (Definition 14), $C_\triangle(\mathcal{P}) = rules(\mathcal{P}) \cup Lit_\triangle(\mathcal{P})$ and $C_\triangle(C_\triangle(\mathcal{P})) = rules(C_\triangle(\mathcal{P})) \cup Lit_\triangle(C_\triangle(\mathcal{P}))$. Therefore, $C_\triangle(C_\triangle(\mathcal{P})) = rules(\mathcal{P}) \cup Lit_\triangle(rules(\mathcal{P}) \cup Lit_\triangle(\mathcal{P}))$. Obviously, $Lit_\triangle(\mathcal{P}) \subseteq Lit_\triangle(rules(\mathcal{P}) \cup Lit_\triangle(\mathcal{P}))$, and hence $C_\triangle(\mathcal{P}) \subseteq C_\triangle(C_\triangle(\mathcal{P}))$.

(2) We must prove that $C_\triangle(C_\triangle(\mathcal{P})) \subseteq C_\triangle(\mathcal{P})$. In other words, we must prove that if $(R, \beta) \in C_\triangle(C_\triangle(\mathcal{P}))$, then $(R, \beta) \in C_\triangle(\mathcal{P})$. By definition,

$$C_\triangle(C_\triangle(\mathcal{P})) = C_\triangle(\mathcal{P}) \cup \{(Q, \alpha) | C_\triangle(\mathcal{P}) |\!\sim_\triangle \langle \mathcal{A}_1, Q, \alpha\rangle\}. \tag{1}$$

Clearly, if $(R, \beta) \in C_\triangle(\mathcal{P})$, there is nothing to prove. We can reduce our original problem to the following formulation:

$$\text{if } (R, \beta) \in \{(Q, \alpha) | C_\triangle(\mathcal{P}) |\!\sim_\triangle \langle \mathcal{A}_1, Q, \alpha\rangle\} \text{ then } (R, \beta) \in C_\triangle(\mathcal{P}). \tag{2}$$

We will prove this statement by induction on the number of steps required for the proof in $\langle \mathcal{A}_1, Q, \alpha\rangle$.

Base case: suppose that $\langle \mathcal{A}_1, Q, \alpha\rangle$ is proven in one inference step. (i.e. $C_\triangle(\mathcal{P}) |\!\sim_\triangle^1 \langle \mathcal{A}_1, Q, \alpha\rangle$ via INTF). But note that from the definition of INTF, it follows that

$$C_\triangle(\mathcal{P}) |\!\sim_\triangle \langle \mathcal{A}, Q, \alpha\rangle \quad \text{iff} \quad (Q, \alpha) \in C_\triangle(\mathcal{P}). \tag{3}$$

Hence Eq. (2) holds.

Inductive hypothesis: Let us assume that Eq. (2) holds for all proofs of length $1 < k$. We will show that it also holds for proofs of length k. Let $(R, \beta) \in \{(Q, \alpha) |\, C_\triangle(\mathcal{P}) |\!\sim_\triangle^{k+1} \langle \mathcal{A}_1, Q, \alpha\rangle\}$. Then it is the case that $\langle \mathcal{A}_1, Q, \alpha\rangle$ was derived

via MPA. If that is the case, then

$$C_{\triangle}(\mathcal{P}) \mid\!\sim_{\triangle}^{k} \langle \mathcal{A}_1, L_1, \alpha_1\rangle, \langle \mathcal{A}_2, L_2, \alpha_2\rangle, \ldots, \langle \mathcal{A}_k, L_k, \alpha_k\rangle, \tag{4}$$

and there is a rule

$$(L_0 \leftarrow L_1 \wedge L_2 \wedge \cdots \wedge L_k, \gamma) \tag{5}$$

with $\gamma \leqslant 1$, such that $\Pi \cup \{(L_0 \leftarrow L_1 \wedge L_2 \wedge \cdots \wedge L_k, \gamma)\} \cup \bigcup_{i=1}^{k} \mathcal{A}_i \not\vdash \bot$, with $\beta = \min(\alpha_1, \ldots, \alpha_k, \gamma)$. Clearly, $\langle \mathcal{A}_1, Q, \alpha\rangle$ follows from Eqs. (4) and (5). By inductive hypothesis, this means that

$$(L_1, \alpha_1), \ldots, (L_k, \alpha_k) \in C_{\triangle}(\mathcal{P}). \tag{6}$$

But the rule in Eq. (5) also applies here, and hence $(L_0, \beta) \in C_{\triangle}(\mathcal{P})$, with $\beta = \min(\alpha_1, \ldots, \alpha_k, \gamma)$, as we wanted to show. □

Monotonicity does not hold for $C_{\triangle}$, as expected. As a counterexample consider the program $\mathcal{P} = \{(q, 1), (p \leftarrow q, 0.9)\}$. Then $(p, 0.9) \in C_{\triangle}(\mathcal{P})$, as there is an argument $\langle \mathcal{A}, p, 0.9\rangle$ on the basis of $\mathcal{P}$ for concluding $(p, 0.9)$, with $\mathcal{A} = \{(p \leftarrow q, 0.9)\}$. However, $(p, 0.9) \notin C_{\triangle}(\mathcal{P} \cup \{(\sim p, 1)\})$ (as no argument for $(p, 0.9)$ could exist, as condition (2) in Definition 8 would be violated). Semi-monotonicity is an interesting property for analyzing non-monotonic consequence relationships. It is satisfied if all defeasible consequences from a given theory are preserved when the theory is augmented with new *defeasible* information.

Proposition 17. *The operator $C_{\triangle}$ satisfies semi-monotonicity when new defeasible information is added, i.e. $C_{\triangle}(\mathcal{P}_1) \subseteq C_{\triangle}(\mathcal{P}_1 \cup \mathcal{P}_2)$ for $\mathcal{P}_2 = (\emptyset, \Delta)$.*

Proof. Follows directly from the structure of the inference rules for $\mid\!\sim_{\triangle}$. Suppose $\mathcal{P}_1 \mid\!\sim_{\triangle} \langle \mathcal{A}, Q, \alpha\rangle$, and consider the program $\mathcal{P}_1 \cup \mathcal{P}_2$ with $\mathcal{P}_2 = (\emptyset, \Delta)$. Clearly, $\langle \mathcal{A}, Q, \alpha\rangle$ can be derived from $\mathcal{P}_1 \cup \mathcal{P}_2$ by applying the same sequence of steps as in $\mathcal{P}_1 \mid\!\sim_{\triangle} \langle \mathcal{A}, Q, \alpha\rangle$, since all preconditions in inference rules are defined w.r.t. ${\mathcal{P}_1}^{\Pi}$, the set of strict knowledge in $\mathcal{P}_1$, and by hypothesis, $(\mathcal{P}_1 \cup \mathcal{P}_2)^{\Pi} = {\mathcal{P}_1}^{\Pi}$. □

Proposition 18. *The operator $C_{\triangle}$ satisfies cummulativity, i.e. $\gamma \in C_{\triangle}(\Gamma)$ implies $\phi \in C_{\triangle}(\Gamma \cup \{\gamma\})$ iff $\phi \in C_{\triangle}(\Gamma)$.*

Proof (*Sketch*). Without loss of generality, we can assume $\gamma = (Q, \alpha)$ is not in Γ (otherwise the proof is straightforward). By hypothesis, $(Q, \alpha) \in C_{\triangle}(\Gamma)$ and there is a sequence $s_1^Q, s_2^Q, \ldots, s_t^Q$ of application of inference rules in {INTF, MPA} such that $\Gamma \mid\!\sim_{\triangle} \langle \mathcal{A}_1, Q, \alpha\rangle$. Let us assume now that $(R, \beta) \in C_{\triangle}(\Gamma \cup \{(Q, \alpha)\})$. This means that there is a sequence $r_1, r_2, \ldots, r_n$ of application of inference rules as before such that $\Gamma \mid\!\sim_{\triangle} \langle \mathcal{A}_2, R, \beta\rangle$. Suppose now that $\langle \mathcal{A}_2, R, \beta\rangle$ does not include $\langle \mathcal{A}_1, Q, \alpha\rangle$ as a subargument. This happens iff from the structure of inference rules for $\mid\!\sim_{\triangle}$, (Q, α) will not be required as intermediate step in the proof of (R, β) iff $(R, \beta) \in C_{\triangle}(\Gamma)$. Suppose now that $\langle \mathcal{A}_2, R, \beta\rangle$ does include $\langle \mathcal{A}_1, Q, \alpha\rangle$ as a subargument. This happens iff in the sequence $r_1, r_2, \ldots, r_n$ we have that $r_{i+k} = s_i^Q$, for $i = 1 \ldots t$, for some $1 \leqslant k \leqslant n$. But from the initial hypothesis this sequence can be built from Γ alone. Hence $\Gamma \mid\!\sim_{\triangle} \langle \mathcal{A}_2, R, \beta\rangle$ or equivalently $(R, \beta) \in C_{\triangle}(\Gamma)$. □

Note that the property of right weakening cannot be considered (in a strict sense) in P-DeLP, since the underlying logic does not allow the application of the deduction theorem. Therefore, well-formed formulas of the form $(x \leftarrow y, \alpha)$ cannot be derived. However, an alternative approach can be intended, introducing a new property in which right weakening is restricted to Horn-like clauses:

Proposition 19. *The operator $C_{\triangle}$ satisfies (Horn) supraclassicality w.r.t. $C_{\vdash}$ (i.e. $C_{\vdash}(\mathcal{P}) \subseteq C_{\triangle}(\mathcal{P})$), and (Horn) right weakening (i.e. if $(Y, \alpha) \in C_{\triangle}(\mathcal{P})$ and $(X \leftarrow Y, 1) \in C_{\vdash}(\mathcal{P})$, then $(X, \alpha) \in C_{\triangle}(\mathcal{P})$).*

Proof. Supraclassicality follows from Proposition 15. For the case of right weakening, let us suppose $(Y, \alpha) \in C_{\triangle}(\mathcal{P})$, i.e. $\mathcal{P}|\sim_{\triangle} \langle \mathcal{A}_1, Y, \alpha\rangle$, for some argument $\langle \mathcal{A}_1, Y, \alpha\rangle$. If $(X \leftarrow Y, 1) \in C_{\vdash}(\mathcal{P})$, by definition of $C_{\vdash}$, necessarily $(X \leftarrow Y, 1) \in \mathcal{P}^{\Pi}$, the set of strict knowledge in $\mathcal{P}$. From $(X \leftarrow Y, 1) \in \mathcal{P}$ and $\mathcal{P}|\sim_{\triangle} \langle \mathcal{A}_1, Y, \alpha\rangle$, by applying inference rule MPA we get $\langle \mathcal{A}_1, X, \alpha\rangle$. □

Proposition 20. *The operator $C_{\triangle}$ satisfies subclassical cummulativity, i.e. $\mathcal{P}_1 \subseteq \mathcal{P}_2 \subseteq C_{\vdash}(\mathcal{P}_1)$ implies $C_{\triangle}(\mathcal{P}_1) = C_{\triangle}(\mathcal{P}_2)$.*

Proof. Clearly, by hypothesis and by the definition of $C_{\vdash}$, it holds that $rules(\mathcal{P}_1) = rules(\mathcal{P}_2)$. Therefore, since $\mathcal{P}_1 \subseteq C_{\vdash}(\mathcal{P}_1)$, the program $\mathcal{P}_1$ contains no uncertain facts, and hence $\mathcal{P}_2$ as well. This means that $\mathcal{P}_1$ and $\mathcal{P}_2$ are of the form $\mathcal{P}_1 = (\Pi_1, \Delta_1)$ and $\mathcal{P}_2 = (\Pi_2, \Delta_2)$ with $\Pi_1 \subseteq \Pi_2$ and $\Delta_1 = \Delta_2$ only contain uncertain rules. Moreover one can show that $Lit_{\vdash}(\mathcal{P}_1) = Lit_{\vdash}(\mathcal{P}_2)$. Indeed, since $C_{\vdash}$ satisfies monotonicity and idempotence, we have $C_{\vdash}(\mathcal{P}_1) \subseteq C_{\vdash}(\mathcal{P}_2) \subseteq C_{\vdash}(C_{\vdash}(\mathcal{P}_1)) = C_{\vdash}(\mathcal{P}_1)$, and since $rules(\mathcal{P}_1) = rules(\mathcal{P}_2)$, it follows that $Lit_{\vdash}(\mathcal{P}_1) = Lit_{\vdash}(\mathcal{P}_2)$. Therefore, for any $A \subseteq \Delta_1 (= \Delta_2)$, A is consistent with Π_1 iff it is consistent with Π_2, and $A \cup \Pi_1 \vdash (Q, \alpha)$ iff $A \cup \Pi_2 \vdash (Q, \alpha)$. In particular this holds for those sets A which are minimal according to the conditions given in the definition of argument (Definition 8). Hence $\langle A, Q, \alpha\rangle$ is an argument w.r.t. $\mathcal{P}_1$ iff it is an argument w.r.t. $\mathcal{P}_2$. □

Most non-pure logical properties for $C_{\triangle}$ do not hold. In particular, $C_{\triangle}$ does not satisfy the properties of left-logical equivalence (LL), conjunction of conclusions (CC), left absorption (LA), right absorption (RA), rational negation (RN), rational monotonicity (RM), disjunctive rationality (DR), as shown next.

- LL: Given two programs $\mathcal{P}_1$ and $\mathcal{P}_2$, $C_{\vdash}(\mathcal{P}_1) = C_{\vdash}(\mathcal{P}_2)$ does not imply $C_{\triangle}(\mathcal{P}_1) = C_{\triangle}(\mathcal{P}_2)$. Consider $\mathcal{P}_1 = \{(y, 1)\}$ and $\mathcal{P}_2 = \mathcal{P}_1 \cup \{(x \leftarrow y, 0.9)\}$.
- CC: Arguments supporting conjunctions of conclusions cannot be expressed in P-DeLP language, as goals are restricted to literals.
- LA: Consider $\mathcal{P} = \{(Q, \alpha)\}$, where Q is a literal, $\alpha < 1$. Then $C_{\vdash}(C_{\triangle}(\mathcal{P})) = C_{\vdash}(\{(Q, \alpha)\}) = \emptyset \neq C_{\triangle}(\mathcal{P})$.
- RA: Consider the same counterexample given for LA. Analogously, $C_{\triangle}(C_{\vdash}(\mathcal{P})) = C_{\triangle}(\emptyset) = \emptyset \neq C_{\triangle}(\mathcal{P})$.
- RN: Consider $\mathcal{P}_1 = \{(\sim p \leftarrow x, 1), (\sim p \leftarrow \sim x, 1), (r, 1), (z \leftarrow p, 1), (p \leftarrow r, 0.9)\}$. Then $\mathcal{P}_1|\sim_{\triangle} \langle \mathcal{A}_1, z, 0.9\rangle$, with $\mathcal{A}_1 = \{(p \leftarrow r, 0.9)\}$. However, $\mathcal{P}_1 \cup \{(x, 1)\} \not|\sim_{\triangle} \langle \mathcal{A}_1, z, 0.9\rangle$, and $\mathcal{P}_1 \cup \{(\sim x, 1)\} \not|\sim_{\triangle} \langle \mathcal{A}_1, z, 0.9\rangle$.
- RM: Consider the same counterexample as given for RN. Then $\mathcal{P}_1|\sim_{\triangle} \langle \mathcal{A}_1, z, 0.9\rangle$, but it is not the case that $\mathcal{P}_1 \cup \{(x, 1)\}|\sim_{\triangle} \langle \mathcal{A}_1, z, 0.9\rangle$ nor $\mathcal{P}_1|\sim_{\triangle} (\sim x, 1)$.
- DR: Actually, the property of disjunctive rationality falls out of the scope of $C_{\triangle}$ as disjunctions cannot be expressed as well-formed formulas in the P-DeLP object language.

6.2. Logical properties for C_w

In this section we will analyze the behavior of the C_w operator for capturing warrant. In fact, the C_w operator only satisfies (Horn) supraclassicality, as it is a direct consequence of Proposition 15, and subclassical cummulativity.

Proposition 21. *The operator C_w satisfies (Horn) supraclassicality w.r.t. $C_{\vdash}$, i.e. $C_{\vdash}(\mathcal{P}) \subseteq C_w(\mathcal{P})$.*

Proof. It suffices to show that $Lit_{\vdash}(\mathcal{P}) \subseteq Lit_w(\mathcal{P})$, or equivalently, we want to show that if $(Q, 1) \in Lit_{\vdash}(\mathcal{P})$ then $(Q, 1) \in Lit_w(\mathcal{P})$.

Clearly, if $(Q, 1) \in Lit_{\vdash}(\mathcal{P})$ then it can be obtained by successive applications of INTF and MPA which do not involve any uncertain clause (otherwise, we would obtain (Q, α), with $\alpha < 1$). As $(Q, 1)$ follows directly from the certain clauses in the program $\mathcal{P}$, it follows that $(Q, 1)$ is supported by an empty argument $\langle \emptyset, Q, 1\rangle$.

Suppose there is a defeater for $\langle \emptyset, Q, 1\rangle$, let us say $\langle \mathcal{A}', \sim Q, \alpha'\rangle$. Clearly α' must be 1 (otherwise $\langle \mathcal{A}', \sim Q, \alpha'\rangle$ would not be a defeater). But from the same line of reasoning as above, $\mathcal{A}'$ should be empty. Therefore the only possible defeater is $\langle \emptyset, \sim Q, 1\rangle$. But this implies that $(\sim Q, 1) \in Lit_{\vdash}(\mathcal{P})$ and by hypothesis $(Q, 1) \in Lit_{\vdash}(\mathcal{P})$. But this implies that the set Π of certain clauses is contradictory, and hence $\mathcal{P}$ is not a program (absurd). Consequently, the argument $\langle \emptyset, Q, 1\rangle$ has no defeaters, and therefore it is warranted, so that it follows that $(Q, 1) \in Lit_w(\mathcal{P})$. □

$$
\begin{array}{llll}
(1) & (\sim y \leftarrow p \wedge \sim r, 1) & (5) & (q \leftarrow z, 0.7) \\
(2) & (y, 1) & (6) & (z \leftarrow p, 0.7) \\
(3) & (p, 0.7) & (7) & (\sim q \leftarrow r, 0.8) \\
(4) & (r, 0.8) & (8) & (\sim r, 0.9)
\end{array}
$$

Fig. 3. Program $\mathcal{P}_{\text{idem}}$ (see Examples 24–26).

Proposition 22. *The operator C_w satisfies subclassical cummulativity, i.e. $\mathcal{P}_1 \subseteq \mathcal{P}_2 \subseteq C_{\vdash}(\mathcal{P}_1)$ implies $C_w(\mathcal{P}_1) = C_w(\mathcal{P}_2)$.*

Proof. The proof follows the same line of reasoning as in Proposition 20. □

Next we will see, through different counterexamples, that other logical properties do not hold for C_w.

Proposition 23. *The operator C_w does not satisfy inclusion, monotonicity nor semi-monotonicity.*

Proof. Different counterexamples suffice.

- *Inclusion*: Let $\mathcal{P} = \{(p, 0.7), (\sim p, 0.9)\}$. Then $C_w(\mathcal{P}) = \{(\sim p, 0.9)\}$, and clearly $\mathcal{P} \not\subseteq C_w(\mathcal{P})$.
- *Monotonicity*: Monotonicity does not hold, as can be seen from the counterexample used for monotonicity in $C_{\triangle}$; in that case, $(q, 0.9) \in C_w(\mathcal{P})$, but $(q, 0.9) \notin C_w(\mathcal{P} \cup \{(\sim p, 1)\})$.
- *Semi-monotonicity*: Semi-monotonicity does not hold either for C_w, as adding new defeasible clauses cannot invalidate already derivable arguments, but it can enable new ones that were not present before, thus modifying the dialectical relationships among arguments. Arguments that were warranted may therefore no longer keep that epistemic status. Consider a variant of the previous counterexample: let $\mathcal{P} = \{(q, 1), (p \leftarrow q, 0.9)\}$. Then $(p, 0.9) \in C_w(\mathcal{P})$, as there is an argument $\langle \mathcal{A}, p, 0.9\rangle$ on the basis of $\mathcal{P}$. However, $(p, 0.9) \notin C_w(\mathcal{P} \cup \{(\sim p, 0.95)\})$, as $\langle \mathcal{A}, p, 0.9\rangle$ is defeated by $\langle \mathcal{B}, \sim p, 0.95\rangle$, with $\mathcal{B} = \{(\sim p, 0.95)\}$. There are no more arguments to consider, and hence $\langle \mathcal{A}, p, 0.9\rangle$ is not warranted. □

Moreover, cummulativity, idempotence and right-weakening do not hold for C_w, as shown in the following counterexamples.

Example 24. Operator C_w does not satisfy idempotence. Consider program $\mathcal{P}_{\text{idem}}$ given in Fig. 3. Note that $q \notin C_w(\mathcal{P}_{\text{idem}})$: there is an argument $\langle \mathcal{A}, q, 0.7\rangle$, with $\mathcal{A} = \{(q \leftarrow z, 0.7), (z \leftarrow p, 0.7), (p, 0.7)\}$ supporting $(q, 0.7)$. In this case, argument $\langle \mathcal{A}, q, 0.7\rangle$ is defeated by $\langle \mathcal{B}, \sim q, 0.8\rangle$, with $\mathcal{B} = \{(\sim q \leftarrow r, 0.8), (r, 0.8)\}$. There is a third argument $\langle \mathcal{C}, \sim r, 0.9\rangle$, with $\mathcal{C} = \{(\sim r, 0.9)\}$. Even though this argument defeats $\langle \mathcal{B}, \sim q, 0.8\rangle$, it cannot be introduced as a defeater in the above analysis, as it would be in conflict with argument $\langle \mathcal{A}, q, 0.7\rangle$, violating the non-contradiction consistency constraint in argumentation lines (since $(\sim y, 1)$ and $(y, 0.7)$ would follow from $\mathcal{P}^{II}_{\text{idem}} \cup \mathcal{A} \cup \mathcal{B}$, where $\mathcal{P}^{II}_{\text{idem}}$ stands for the certain knowledge in $\mathcal{P}_{\text{idem}}$). The set of all warranted literals supported by $\mathcal{P}_{\text{idem}}$ is $W = \{(p, 0.7), (z, 0.7), (\sim r, 0.9)\}$. Consider now the program $\mathcal{P}' = \mathcal{P}_{\text{idem}} \cup W$. Let us analyze whether q is warranted or not w.r.t. $\mathcal{P}'$. There is an argument $\langle \mathcal{A}', q, 0.7\rangle$, with $\mathcal{A}' = \{(q \leftarrow z, 0.7)\}$, which is defeated by $\langle \mathcal{B}, \sim q, 0.8\rangle$ (as before). This defeater is defeated by $\langle \mathcal{C}', \sim r, 0.9\rangle$, with $\mathcal{C}' = \emptyset$. There are no more arguments to consider, and therefore $(q, 0.7)$ is warranted. Hence $q \in C_w(\mathcal{P}') = C_w(C_w(\mathcal{P}_{\text{idem}}))$, and as shown above $q \notin C_w(\mathcal{P}_{\text{idem}})$. Therefore C_w does not satisfy idempotence.

Example 25. Operator C_w does not satisfy cummulativity. We must show that there exists a weighed literal for some program $\mathcal{P}$ such that if $(Q, \alpha) \in C_w(\mathcal{P})$, then $(R, \beta) \in C_w(\mathcal{P} \cup \{(Q, \alpha)\})$ does not imply $(R, \beta) \in C_w(\mathcal{P})$. Consider program $\mathcal{P}_{\text{idem}}$ in Fig. 3. As shown in Example 24, $(z, 0.7) \in C_w(\mathcal{P}_{\text{idem}})$, and $(q, 0.7) \in C_w(\mathcal{P}_{\text{idem}} \cup \{(z, 0.7)\})$. However, $(q, 0.7) \notin C_w(\mathcal{P}_{\text{idem}})$. Hence cummulativity does not hold for C_w.

Example 26. Operator C_w does not satisfy right weakening. Consider program $\mathcal{P}_{\text{idem}}$ in Fig. 3. Note that $(p, 0.7) \in C_w(\mathcal{P}_{\text{idem}})$ and $(\sim r, 0.9) \in C_w(\mathcal{P}_{\text{idem}})$. Besides, $(\sim y \leftarrow p, \sim r, 1) \in \mathcal{P}^{\Pi}_{\text{idem}}$, the set of strict knowledge in $\mathcal{P}_{\text{idem}}$. However, the conclusion of this certain rule is *not* warranted, i.e. $(\sim y, 0.7) \notin C_w(\mathcal{P}_{\text{idem}})$, since $(y, 1) \in \mathcal{P}^{\Pi}_{\text{idem}}$, and thus there exists no argument with conclusion $(\sim y, 0.7)$ (as it would violate condition (2) in Definition 8).

Finally, as for $C_\triangle$, most non-pure logical properties for C_w do not hold. In particular, C_w does not satisfy the properties of LL, CC, LA, RA, RN, RM and DR. In all cases this is based on the existence of counterexamples following the same line of reasoning as for $C_\triangle$.

6.3. Discussion

Let us briefly discuss the most relevant results provided by the properties presented before. When analyzing argumentative inference under the operator $C_\triangle$, idempotence shows us that adding argument conclusions as new facts to a given program does not add any new inference capabilities. Cummulativity shows us that any argument obtained from a program $\mathcal{P}$ can be kept as an intermediate proof (lemma) to be used in building more complex arguments. (Horn) supraclassicality indicates that every conclusion that follows via traditional SLD inference (involving only certain clauses) can be considered as a special form of argument (namely, an empty argument), whereas Horn right weakening tells us that certain rules in P-DeLP preserve the usual semantics for Horn rules (the existence of a certain rule $X \leftarrow Y$ causes that every argument concluding Y is also an argument for X).

Computing warrant also can be better understood in the light of the logical properties for C_w. There are only two properties (supraclassicality and subclassical cummulativity) which hold for this operator. Next we will briefly discuss some of the relevant properties which do not hold for C_w and their relationship with argument-based inference. In [35] some examples are informally presented to show that argumentation systems should assign facts a special status, and therefore should *not* be cumulative. In the particular case of cummulativity (traditionally the most defended property associated with non-monotonic inference), we have shown that it does not hold for C_w even when warranted conclusions are assigned the epistemic status of uncertain facts of the form (Q, α), $\alpha < 1$, which provides an even stronger result than the one suggested originally in [35]. From Horn supraclassicality it follows that every conclusion obtained from certain clauses is a particular case of warranted literal.

The failure of Horn right weakening indicates that a certain rule of the form $(Y \leftarrow X, 1)$ does *not* ensure that every warranted argument for (X, α) (with $\alpha < 1$) implies that (Y, α) is also warranted. In fact, it can be the case that the certain fact $(\sim Y, 1)$ is present in a given program, so that an argument for the goal Y cannot be even computed (as shown in Example 26). In a recent paper [13], Caminada and Amgoud identify this situation as a particular anomaly in several argumentation formalisms (e.g. [34,26]) and provide an interesting solution in terms of *rationality postulates* which—the authors claim—should hold in any well-defined argumentative system. In the case of P-DeLP the problem seems to require a different conceptualization, as the necessity degree 1 of the rule $(Y \leftarrow X, 1)$ is attached to the rule itself, and the necessity degree of the conclusion Y *depends* on the necessity degree α of the antecedent X. As an example, consider the program $\mathcal{P} = \{(\sim g \leftarrow a, 1), (a, 0.7), (g \leftarrow b, 1), (b, 0.4)\}$. In this case, $(a, 0.7)$ and $(b, 0.4)$ are warranted conclusions. However, we cannot warrant g and $\sim g$ with necessity degree 1. In fact, only $(\sim g, 0.7)$ can be warranted. In this respect, we have started a preliminary analysis for this problem in the context of P-DeLP, and currently part of our research is focused on this issue. In particular, we are formalizing a new conceptualization of what warranted and blocked goals with respect to a program should be. This new approach, where warranted and blocked goals are attached with degrees in a similar way of [33], addresses all rationality postulates proposed in [13] without the need of extending the original program with the transposed versions of all strict rules.

7. Related work

In the last years the development of combined approaches based on qualitative reasoning and uncertainty has deserved much research work [31]. Preference-based approaches to argumentation have been developed, many of them oriented towards formalizing conflicting desires in MAS [4–6]. In contrast with these preference-based approaches, the P-DeLP framework involves necessity measures explicitly attached to fuzzy formulas and the proof procedure of the underlying possibilistic fuzzy logic is used for computing the necessity measure for arguments.

There have also been generic approaches connecting defeasible reasoning and possibilistic logic (e.g. [10]), and recently a number of hybrid approaches connecting argumentation and uncertainty have been developed. Probabilistic argumentation systems [28,29] use probabilities to compute degrees of support and plausibility of goals, related to Dempster–Shafer belief and plausibility functions. However this is not a dialectics-based framework as opposed to the one presented in this paper. In [37] a fuzzy argumentation system based on extended logic programming is proposed. In contrast with our framework, this approach relies only on fuzzy values applied to literals and there is no explicit treatment of possibilistic uncertainty.

Research in logical properties for defeasible argumentation can be traced back to Benferhat et al. [8,9] and Vreeswijk [40,41]. In the context of his abstract argumentation systems, Vreeswijk showed that many logical properties for non-monotonic inference relationships turned out to be counterintuitive for argument-based systems. Benferhat et al. [8] were the first who studied argumentative inference in uncertain and inconsistent knowledge bases. They defined an argumentative consequence relationship $\vdash_{\mathcal{A}}$ taking into account the existence of arguments favoring a given conclusion against the absence of arguments in favor of its contrary. In contrast, the $|\!\sim_{\widetilde{w}}$ relationship proposed in this paper takes into account the *whole* dialectical analysis for arguments derivable from the program for any given goal.

In [8,9] the authors also extend the argumentative relation $\vdash_{\mathcal{A}}$ to prioritized knowledge bases, assessing weights to conclusions on the basis of the $\vdash_{\pi}$ entailment relationship from possibilistic logic [23]. A direct comparison to our $|\!\sim_{\widetilde{w}}$ relationship is not easy since we are using a logic programming framework and not general propositional logic, but roughly speaking while $\vdash_{\pi}$ takes into account the inconsistency degree associated with the whole knowledge base, our logic programming framework allows us to perform a dialectical analysis restricted only to conflicting arguments related with the goal being solved.

8. Conclusions and future work

In this paper we have described how to model an agent's beliefs and perceptions using P-DeLP. A salient feature of P-DeLP is that it is based on two logical frameworks that have already been implemented (namely PGL [3] and DeLP [26]). Several features leading to efficient implementations of the argumentative proof procedure described in this paper have been also recently studied, particularly those related to comparing conflicting arguments by specificity [39] and defining transformation properties for DeLP programs [15].

In this context, P-DeLP keeps all the original features of DeLP while incorporating more expressivity and representation capabilities by means of possibilistic uncertainty and fuzzy knowledge. One particularly interesting feature of P-DeLP is the possibility of defining aggregated preference criteria by combining the necessity measures associated with arguments with other syntax-based criteria (e.g. specificity [38,39]).

Part of our current research work will be developed into different directions. First, we will extend the existing implementation of DeLP to incorporate the new features of P-DeLP. Second, we will apply the resulting implementation of P-DeLP to improve existing real-world applications of DeLP and to develop new ones. Finally, we will analyze extending P-DeLP to first order. It must be remarked that the Generalized Modus Ponens rule used in P-DeLP is syntactically the same as the one used in possibilistic logic [23]. As a consequence, to implement the machinery of P-DeLP the underlying possibilistic fuzzy logic PGL can be replaced by the possibilistic logic. The advantage of this approach is that the current logic programming system can be extended to first order, incorporating fuzzy unification between fuzzy constants [3]. In this respect, in [1] we have extended P-DeLP through the use of fuzzy constants and fuzzy unification at the object language and we have proposed a way to handle conflicting arguments in the context of the extended framework.

Acknowledgments

We thank anonymous reviewers for their comments and suggestions to improve the final version of this paper. This research was partially supported by CYCYT Projects MULOG (TIN2004-07933-C03-01/03) and IEA (TIN2006-15662-C02-01/02), by CONICET (Argentina), by the Secretaría General de Ciencia y Tecnología de la Universidad Nacional del Sur (Project 24/ZN10), and by Agencia Nacional de Promoción Científica y Tecnológica (PICT 2002 No. 13096).

Appendix A. Non-monotonic inference relationships and their properties

A.1. Fundamentals

In classical logic, inference rules allow us to determine whether a given well-formed formula γ follows via "$\vdash$" from a set Γ of well-formed formulas, where "$\vdash$" is a *consequence relationship* (satisfying idempotence, *cut* and monotonicity). As non-monotonic and defeasible logics evolved into a valid alternative to formalize commonsense reasoning a similar concept was needed to capture the notion of logical consequence without demanding some of these requirements (e.g. monotonicity). This led to the definition of a more generic notion of inference in terms of *inference relationships*. Given a set Γ of well-formed formulas in an arbitrary logical language $\mathcal{L}$, we write $\Gamma|\sim \gamma$ to denote an inference relationship "$|\sim$", where γ is a (non-monotonic) consequence of Γ. We define an *inference operator* $C_{|\sim}$ associated with an inference relationship, with $C_{|\sim}(\Gamma) = \{\gamma \mid \Gamma|\sim \gamma\}$. Given an inference relationship "$|\sim$" and a set Γ of sentences, the following are called *basic* (or *pure*) properties associated with the inference operator $C_{|\sim}(\Gamma)$:

(1) **Inclusion (IN)**: $\Gamma \subseteq C(\Gamma)$.
(2) **Idempotence (ID)**: $C(\Gamma) = C(C(\Gamma))$.
(3) **Cut (CT)**: $\Gamma \subseteq \Phi \subseteq C(\Gamma)$ implies $C(\Phi) \subseteq C(\Gamma)$.
(4) **Cautious monotonicity (CM)**: $\Gamma \subseteq \Phi \subseteq C(\Gamma)$ implies $C(\Gamma) \subseteq C(\Phi)$.
(5) **Cummulativity (CU)**: $\gamma \in C(\Gamma)$ implies $\phi \in C(\Gamma \cup \{\gamma\})$ iff $\phi \in C(\Gamma)$, for any well-formed formulas $\gamma, \phi \in \mathcal{L}$.
(6) **Monotonicity (MO)**: $\Gamma \subseteq \Phi$ implies $C(\Gamma) \subseteq C(\Phi)$.

These properties are called *pure*, since they can be applied to any language $\mathcal{L}$, and are abstractly defined for an arbitrary inference relationship "$|\sim$". Nevertheless, other properties which link a classical inference operator *Th* with an arbitrary inference relationship can be stated. In what follows we will assume that *Th* stands for an operator that characterizes classical inference, whereas *C* corresponds to some (non-monotonic) inference relationship "$|\sim$".

A.2. Horn and non-Horn logical properties

A common name for cataloging these non-pure properties is the distinction between *Horn properties* and *non-Horn properties*.[6] Next we summarize the most important non-pure properties. An in-depth discussion of these properties can be found elsewhere [30].

Horn properties:

(1) **Supraclassicality**: $Th(A) \subseteq C(A)$.
(2) **Left logical equivalence (LL)**: $Th(A) = Th(B)$ implies $C(A) = C(B)$.
(3) **Right weakening (RW)**: If $x \supset y \in Th(A)$ and $x \in C(A)$ then $y \in C(A)$.[7]
(4) **Conjunction of conclusions (CC)**: If $x \in C(A)$ and $y \in C(A)$ then $x \wedge y \in C(A)$.
(5) **Subclassical cummulativity (SC)**: If $A \subseteq B \subseteq Th(A)$ then $C(A) = C(B)$.
(6) **Left absorption (LA)**: $Th(C(\Gamma)) = C(\Gamma)$.
(7) **Right absorption (RA)**: $C(Th(\Gamma)) = C(\Gamma)$.

Non-Horn properties:

(1) **Rationality of negation (RN)**: If $A|\sim z$ then either $A \cup \{x\}|\sim z$ or $A \cup \{\sim x\}|\sim z$.
(2) **Disjunctive rationality (DR)**: If $A \cup \{x \vee y\}|\sim z$ then $A \cup \{x\}|\sim z$ or $A \cup \{y\}|\sim z$.
(3) **Rational monotonicity (RM)**: If $A|\sim z$ then either $A \cup \{x\}|\sim z$ or $A|\sim \sim x$.

---

[6] Horn properties have the form "*from the presence of some particular inferences, the presence of some other inferences can be assured*". Non-Horn properties, on the other hand, have the form "*from the absence of some particular inferences, the absence of some other inferences can be assured*".

[7] It should be noted that "$\supset$" stands for material implication, to be distinguished from the symbol " $\leftarrow$ " used in a logic programming setting.

References

[1] T. Alsinet, C. Chesñevar, L. Godo, G. Simari, S. Sandri, Formalizing argumentative reasoning in a possibilistic logic programming setting with fuzzy unification, Internat. J. Approx. Reasoning, 2008, in press, doi:10.1016/j.ijar.2007.07.004.

[2] T. Alsinet, L. Godo, A complete calculus for possibilistic logic programming with fuzzy propositional variables, in: C. Boutilier, M. Goldszmidt (Eds.), Proc. 16th Conf. on Uncertainty in Artificial Intelligence (UAI-2000), Morgan Kaufmann, Stanford, CA, June 30–July 3, 2000, pp. 1–10.

[3] T. Alsinet, L. Godo, A proof procedure for possibilistic logic programming with fuzzy constants, in: S. Benferhat, P. Besnard (Eds.), Proc. 6th European Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2001), Lecture Notes in Computer Science, Vol. 2143, Springer, Berlin, 2001, pp. 760–771.

[4] L. Amgoud, A formal framework for handling conflicting desires, in: T.D. Nielsen, N.L. Zhang (Eds.), Proc. 7th European Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2001), Lecture Notes in Computer Science, Vol. 2711, Springer, Berlin, 2003, pp. 552–563.

[5] L. Amgoud, C. Cayrol, Inferring from inconsistency in preference-based argumentation frameworks, J. Automat, Reasoning 29 (2002) 125–169.

[6] L. Amgoud, H. Prade, Reaching agreement through argumentation: a possibilistic approach, in: D. Dubois, C.A. Welty, M.A. Williams (Eds.), Proc. 9th Internat. Conf. on Principles of Knowledge Representation and Reasoning (KR2004), Whistler, Canada, June 2–5, 2004, pp. 175–182.

[7] G. Antoniou, D. Billington, Relating defeasible and default logic, in: M. Stumptner, D. Corbett, M.J. Brooks (Eds.), Australian Joint Conf. on Artificial Intelligence, Lecture Notes in Computer Science, Vol. 2256, Springer, Berlin, 2001, pp. 13–24.

[8] S. Benferhat, D. Dubois, H. Prade, Argumentative inference in uncertain and inconsistent knowledge bases, in: D. Heckerman, E.H. Mamdani (Eds.), Proc. 9th Annu. Conf. on Uncertainty in Artificial Intelligence (UAI'93), Morgan Kaufmann, Providence, Washington, DC, USA, July 9–11, 1993, pp. 411–419.

[9] S. Benferhat, D. Dubois, H. Prade, Some syntactic approaches to the handling of inconsistent knowledge bases: a comparative study. Part ii: the prioritized case, in: E. Orlowska (Ed.), Logic at Work, Vol. 24, Physica-Verlag, Heidelberg, 1998, pp. 473–511.

[10] S. Benferhat, D. Dubois, H. Prade, The possibilistic handling of irrelevance in exception-tolerant reasoning, Ann. Math. Artificial Intelligence 35 (2002) 29–61.

[11] P. Besnard, A. Hunter, A logic-based theory of deductive arguments, Artificial Intelligence 128 (2001) 203–235.

[12] R. Brena, J. Aguirre, C. Chesñevar, E. Ramirez, L. Garrido, Knowledge and information distribution leveraged by intelligent agents, Knowledge Inform. Systems 12 (2) (2007) 203–227.

[13] M. Caminada, L. Amgoud, On the evaluation of argumentation formalisms, Artificial Intelligence 171 (2007) 286–310.

[14] M. Capobianco, C. Chesñevar, G. Simari, An argument-based framework to model an agent's beliefs in a dynamic environment, in: I. Rahwan, P. Moraitis, C. Reed (Eds.), Lecture Notes in Computer Science (First Internat. Workshop on Argumentation in Multi-Agent Systems, New York, NY, USA, July 19, 2004.), Vol. 3366, Springer, Berlin, 2005, pp. 95–110.

[15] C. Chesñevar, J. Dix, F. Stolzenburg, G. Simari, Relating defeasible and normal logic programming through transformation properties, Theoret. Comput. Sci. 290 (1) (2003) 499–529.

[16] C. Chesñevar, A. Maguitman, An argumentative approach to assessing natural language usage based on the web corpus, in: R. López de Mántaras, L. Saitta (Eds.), Proc. 16th European Conf. on Artificial Intelligence (ECAI'2004), IOS Press, Valencia, Spain, August 22–27, 2004, pp. 581–585.

[17] C. Chesñevar, A. Maguitman, ARGUENET: an argument-based recommender system for solving web search queries, in: Proc. 2004 2nd Internat. IEEE Conf. on Intelligent Systems (IS-2004), Vol. 1, 22–24 June 2004, Varna, Bulgaria, pp. 282–287.

[18] C. Chesñevar, A. Maguitman, R. Loui, Logical models of argument, ACM Comput. Surveys 32 (4) (2000) 337–383.

[19] C. Chesñevar, A. Maguitman, G. Simari, A first approach to argument-based recommender systems based on defeasible logic programming, in: J.P. Delgrande, T. Schaub (Eds.), Proc. 10th Internat. Workshop on Non-Monotonic Reasoning (NMR-2004), Whistler, Canada, June 6–8, 2004, pp. 109–117.

[20] C. Chesñevar, G. Simari, T. Alsinet, L. Godo, A logic programming framework for possibilistic argumentation with vague knowledge, in: D.M. Chickering, J.Y. Halpern (Eds.), Proc. 20th Conf. on Uncertainty in Artificial Intelligence (UAI'04), AUAI Press, Banff, Canada, July 7–11, 2004, pp. 76–84.

[21] C. Chesñevar, G. Simari, L. Godo, Computing dialectical trees efficiently in possibilistic defeasible logic programming, in: C. Baral et al. (Eds.), LNAI/LNCS Springer Series, Vol. 3662 (Proc. 8th Internat. Conf. on Logic Programming and Nonmonotonic Reasoning LPNMR 2005), Springer, Berlin, 2005, pp. 158–171.

[22] C. Chesñevar, G. Simari, L. Godo, T. Alsinet, Argument-based expansion operators in possibilistic defeasible logic programming: characterization and logical properties, in: L. Godo (Ed.), Proc. 8th European Conf. on Qualitative Aspects of Reasoning under Uncertainty (ECSQARU), Barcelona, Spain, Lecture Notes in Computer Science, Vol. 3571, Springer, Berlin, 2005, pp. 353–365.

[23] D. Dubois, J. Lang, H. Prade, Possibilistic logic, in: D. Gabbay, C. Hogger, J. Robinson (Eds.), Handbook of Logic in Artificial Intelligence and Logic Programming, Nonmonotonic Reasoning and Uncertain Reasoning, Vol. 3, Claredon Press, Oxford, 1994, pp. 439–513.

[24] D. Dubois, H. Prade, Fuzzy sets in approximate reasoning—Part 1: inference with possibility distributions, Fuzzy Sets and Systems 40 (1) (1991) 143–202.

[25] P. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning and logic programming and n-person games, Artificial Intelligence 77 (1995) 321–357.

[26] A. García, G. Simari, Defeasible logic programming: an argumentative approach, Theory and Practice of Logic Programming 4 (1) (2004) 95–138.

[27] S. Gómez, C. Chesñevar, A hybrid approach to pattern classification using neural networks and defeasible argumentation, in: Proc. 17th Internat. FLAIRS Conf. American Association for Artificial Intelligence, Miami, FL, USA, 2004, pp. 393–398.

[28] R. Haenni, J. Kohlas, N. Lehmann, Probabilistic argumentation systems, in: J. Kohlas, S. Moral (Eds.), Handbook of Defeasible Reasoning and Uncertainty Management Systems, Algorithms for Uncertainty and Defeasible Reasoning, Vol. 5, Kluwer Academic Publishers, Dordrecht, 2000, pp. 221–288.
[29] R. Haenni, N. Lehmann, Probabilistic argumentation systems: a new perspective on dempster-shafer theory, Internat. J. Intelligent Systems 1 (18) (2003) 93–106.
[30] D. Makinson, General patterns in nonmonotonic reasoning, in: D. Gabbay, C. Hogger, J. Robinson (Eds.), Handbook of Logic in Artificial Intelligence and Logic Programming, Nonmonotonic and Uncertain Reasoning, Oxford University Press, Oxford, 1994, pp. 35–110.
[31] S. Parsons, Qualitative Methods for Reasoning under Uncertainty, MIT Press, Cambridge, MA, 2001.
[32] S. Parsons, C. Sierra, N. Jennings, Agents that reason and negotiate by arguing, J. Logic Comput. 8 (3) (1998) 261–292.
[33] J.L. Pollock, Defeasible reasoning with variable degrees of justification, Artificial Intelligence 133 (1–2) (2001) 233–282.
[34] H. Prakken, G. Sartor, Argument-based extended logic programming with defeasible priorities, J. Appl. Non-classical Logics 7 (1997) 25–75.
[35] H. Prakken, G. Vreeswijk, Logical systems for defeasible argumentation, in: D. Gabbay, F. Guenther (Eds.), Handbook of Philosophical Logic, Kluwer Academic Publishers, Dordrecht, 2002, pp. 219–318.
[36] I. Rahwan, S. Ramchurn, N. Jennings, P. McBurney, S. Parsons, L. Sonenberg, Argumentation-based negotiation, Knowledge Eng. Rev. 18 (4) (2003) 343–375.
[37] R. Schweimeier, M. Schroeder, Fuzzy argumentation and extended logic programming, in: Proc. ECSQARU 2001 Workshop "Adventures in Argumentation", Toulouse, France, 2001.
[38] G. Simari, R. Loui, A mathematical treatment of defeasible reasoning and its implementation, Artificial Intelligence 53 (1992) 125–157.
[39] F. Stolzenburg, A. García, C. Chesñevar, G. Simari, Computing generalized specificity, J. Non-Classical Logics 13 (1) (2003) 87–113.
[40] G. Vreeswijk, Studies in defeasible argumentation, Ph.D. Thesis, Vrije University, Amsterdam, Holanda, 1993.
[41] G. Vreeswijk, Abstract argumentation systems, Artificial Intelligence 90 (1–2) (1997) 225–279.
[42] M.J. Wooldridge, Introduction to Multiagent Systems, Wiley, New York, 2002.