Modeling Defeasible Argumentation within a Possibilistic Logic Framework with Fuzzy Unification

Teresa Alsinet **Carlos I. Chesñevar**
Dept. of Computer Science
University of Lleida, Spain
{tracy,cic}@eps.udl.es

Lluís Godo **Sandra Sandri**
AI Research Institute (IIIA)
Bellaterra, Spain
{godo, sandri}@iiia.csic.es

Abstract

Possibilistic Defeasible Logic Programming (P-DeLP) is a logic programming language which combines features from argumentation theory and logic programming, incorporating the treatment of possibilistic uncertainty at object-language level. This paper presents a first approach towards extending P-DeLP to incorporate fuzzy constants and fuzzy propositional variables. We focus on how to characterize the resulting, extended language, and how to deal with conflicting arguments in the context of the proposed framework.

Keywords: Possiblilistic logic, fuzzy constants, fuzzy unification, defeasible argumentation.

1 Introduction

In the last decade, defeasible argumentation has emerged as a very powerful paradigm to model commonsense reasoning in the presence of incomplete and potentially inconsistent information [6]. Recent developments have been oriented towards integrating argumentation as part of logic programming languages. In this context, Possibilistic Defeasible Logic Programming (P-DeLP) [7] is a logic programming language which combines features from argumentation theory and logic programming, incorporating the treatment of possibilistic uncertainty at object-language level See also [4, 5] for other argumentation formalisms using possibilistic logic to handle merging of prioritized information.

In spite of its expressive power, an important limitation in P-DeLP (as defined in [7]) is that the treatment of fuzzy constants was not formalized. One interesting alternative for such formalization is the use of PGL$^+$, a Possibilistic Gödel logic extended with fuzzy constants. This paper presents a first approach towards extending P-DeLP through the use of PGL$^+$ in order to incorporate fuzzy constants and fuzzy propositional variables. We focus on how to characterize the resulting, extended language, and how to deal with conflicting arguments in the context of the proposed framework. The rest of the paper is structured as follows: first, in Section 2 we present the fundamentals of PGL$^+$. Then in Section 3 we define the DePGL$^+$ programming language. Section 4 focuses on the characterization of arguments in DePGL$^+$, and Section 5 analyzes the notions of conflict among arguments in the context of our proposal. Finally Section 6 concludes and discusses future work.

2 PGL$^+$: fundamentals

Possibilistic logic is a logic of uncertainty where a certainty degree between 0 and 1, interpreted as a lower bound of a necessity measure, is attached to each classical formula. In [1, 2] we defined a logical system for reasoning under possibilistic uncertainty and disjunctive vague knowledge with an efficient and complete proof procedure for atomic de-

duction when clauses fulfill two kinds of constraints. The formalism –called Possibilistic Gödel logic extended with fuzzy constants or PGL^+– is an implication-based extension of Possibilistic logic defined on top of (the positive fragment of) Gödel infinitely-valued logic, capable of dealing with fuzzy constants, allowing thus also fuzzy unification.

The *basic components* of PGL^+ formulas are: a set of primitive propositions (fuzzy propositional variables) *Var*; *sorts* of constants; a set $\mathcal{C}$ of object *constants* (crisp and fuzzy constants), each having its sort; a set *Pred* of *unary*[1] *regular* predicates, each one having a type (a *type* is a tuple of sorts); and *connectives* $\wedge$, $\rightarrow$. An *atomic formula* is either a primitive proposition from *Var* or of the form $p(A)$, where p is a predicate symbol from *Pred*, A is an object constant from $\mathcal{C}$ and the sort of A corresponds to the type of p. *Formulas* are Horn-rules of the form $p_1 \wedge \cdots \wedge p_k \rightarrow q$ with $k \geq 0$, where $p_1, \ldots, p_k, q$ are atomic formulas. A (weighted) *clause* is a pair of the form (φ, α), where φ is a Horn-rule and $\alpha \in [0,1]$.

Fuzzy constants are seen as (flexible) restrictions on an existential quantifier. Moreover, it is natural to take the truth-value of formulas, for instance, $salary(low)$, under a given interpretation in which the salary is x_0 euros, as the degree in which the salary x_0 is considered to be *low*, i.e. $\mu_{low}(x_0)$. This leads to treat formulas as many-valued, with the unit interval $[0,1]$ as set of truth-values.

A many-valued *interpretation* for the language is a structure $w = (U, i, m)$ which maps each basic sort σ into a non-empty domain U_σ; a primitive proposition q into a value $i(q) \in [0,1]$; a predicate p of type (σ) into a value $i(p) \in U_\sigma$; and an object constant A (crisp or fuzzy constant) of sort σ into a normalized fuzzy set $m(A)$ with membership function $\mu_{m(A)} : U_\sigma \rightarrow [0,1]$. Remark that for each predicate symbol p, $i(p)$ is the one and only value of domain which satisfies p in that interpretation. Indeed, the intended meaning of a fuzzy constant in PGL^+ is to express *disjunctive* knowledge about the possibly unknown value of the corresponding predicate. The *truth value* of an atomic formula φ under an interpretation $w = (U, i, m)$, denoted by $w(\varphi)$, is just $i(q)$ if φ is a primitive proposition q, and it is computed as $\mu_{m(A)}(i(p))$ if φ is of the form $p(A)$. This truth value extends to rules by means of the min-conjunction and Gödel's many-valued implication: $w(p_1 \wedge \cdots \wedge p_k \rightarrow q)$ is 1 if $\min(w(p_1), \ldots, w(p_k)) \leq w(q)$, and is $w(q)$ otherwise.

Notice that $w(\varphi)$ may take any intermediate value between 0 and 1 as soon as φ contains some fuzzy constant and $w(\varphi)$ depends not only on the crisp relations assigned to predicate symbols, but on the fuzzy sets assigned to fuzzy constants. Then, in order to define the possibilistic semantics, we need to fix a meaning for the fuzzy constants and to consider some extension of the standard notion of necessity measure for fuzzy events. The first is achieved by fixing a *context*. Basically a context is the set of interpretations sharing a common domain U and an interpretation of object constants m. So, given U and m, its associated context $I_{U,m}$ is just the set $\{w \text{ interpretation} \mid w = (U, i, m)\}$ and, once fixed the context, $[\varphi]$ denotes the fuzzy set of models for a formula φ defining $\mu_{[\varphi]}(w) = w(\varphi)$, for all $w \in \mathcal{I}_{U,m}$.

Now, in a fixed context $\mathcal{I}_{U,m}$, a belief state (or *possibilistic model*) is determined by a normalized possibility distribution on $\mathcal{I}_{U,m}$, $\pi : \mathcal{I}_{U,m} \rightarrow [0,1]$. Then, we say that π *satisfies* a clause (φ, α), written $\pi \models (\varphi, \alpha)$, iff the (suitable) necessity measure of the fuzzy set of models of φ with respect to π, denoted $N([\varphi] \mid \pi)$, is indeed at least α. Here, for the sake of soundness preservation, we take

$$N([\varphi] \mid \pi) = \inf_{w \in \mathcal{I}_{U,m}} \pi(w) \Rightarrow \mu_{[\varphi]}(w),$$

where $\Rightarrow$ is the reciprocal of Gödel's many-valued implication, defined as $x \Rightarrow y = 1$ if $x \leq y$ and $x \Rightarrow y = 1 - x$, otherwise. This necessity measure for fuzzy sets was proposed and discussed by Dubois and Prade (cf. [8]). For example, according to this semantics, the

[1] We restrict ourselves to unary predicates for the sake of simplicity. However, since variables and function symbols are not allowed, the language still remains propositional.

formula

$$(age\_Peter(about\_35), 0.9)$$

is to be interpreted in PGL$^+$ as the following set of clauses with imprecise but non-fuzzy constants

$$\{(age\_Peter([about\_35]_\beta), \min(0.9, 1-\beta) : \beta \in [0,1]\}$$

As usual, a set of clauses P is said to *entail* another clause (φ, α), written $P \models (\varphi, \alpha)$, iff every possibilistic model π satisfying all the clauses in P also satisfies (φ, α), and we say that a set of clauses P is *satisfiable* in the context determined by U and m if there exists a normalized possibility distribution $\pi : \mathcal{I}_{U,m} \rightarrow [0,1]$ that satisfies all the clauses in P. Satisfiable clauses enjoy the following result [3]: If P is satisfiable and $P \models (\varphi, \alpha)$, with $\alpha > 0$, there exists at least an interpretation $w \in \mathcal{I}_{U,m}$ such that $w(\varphi) = 1$.

Finally, still in a context $\mathcal{I}_{U,m}$, the *degree of possibilistic entailment* of an atomic formula (or goal) φ by a set of clauses P, denoted by $\|\varphi\|_P$, is the greatest $\alpha \in [0,1]$ such that $P \models (\varphi, \alpha)$. In [3], it is proved that $\|\varphi\|_P = \inf\{N([\varphi] \mid \pi) \mid \pi \models P\}$.

The calculus for PGL$^+$ in a given context $\mathcal{I}_{U,m}$ is defined by the following set of inference rules:

Generalized resolution:

$$\frac{(p \wedge s \rightarrow q(A), \alpha),\ (q(B) \wedge t \rightarrow r, \beta)}{(p \wedge s \wedge t \rightarrow r, \min(\alpha, \beta))}\ [\text{GR}],\ \text{if } A \subseteq B$$

Fusion:

$$\frac{(p(A) \wedge s \rightarrow q(D), \alpha),\ (p(B) \wedge t \rightarrow q(E), \beta)}{(p(A \cup B) \wedge s \wedge t \rightarrow q(D \cup E), \min(\alpha, \beta))}\ [\text{FU}]$$

Intersection:

$$\frac{(p(A), \alpha), (p(B), \beta)}{(p(A \cap B), \min(\alpha, \beta))}\ [\text{IN}]$$

Resolving uncertainty:

$$\frac{(p(A), \alpha)}{(p(A'), 1)}\ [\text{UN}],\quad \text{where } A' = \max(1-\alpha, A)$$

Semantical unification:

$$\frac{(p(A), \alpha)}{(p(B), \min(\alpha, N(B \mid A)))}\ [\text{SU}]$$

For each context $\mathcal{I}_{U,m}$, the above GR, FU, SU, IN and UN inference rules can be proved to be *sound* with respect to the possibilistic entailment of clauses. Moreover we shall also refer to the following weighted **modus ponens** rule, which can be seen as a particular case of the GR rule

$$\frac{(p_1 \wedge \ldots \wedge p_n \rightarrow q, \alpha),\ (p_1, \beta_1), \ldots, (p_n, \beta_n)}{(q, \min(\alpha, \beta_1, \ldots, \beta_n))}\ [\text{MP}]$$

Finally, the notion of *proof* in PGL$^+$, denoted by $\vdash$, is deduction by means of the triviality axiom and the PGL$^+$ inference rules. Then, given a context $\mathcal{I}_{U,m}$, the *degree of deduction* of a goal φ from a set of clauses P, denoted $|\varphi|_P$, is the greatest $\alpha \in [0,1]$ for which $P \vdash (\varphi, \alpha)$.

In [2, 3] it is shown that this notion of proof is complete for determining the degree of possibilistic entailment of a goal, i.e. $|\varphi|_P = \|\varphi\|_P$, for non-recursive and satisfiable programs P, called PGL$^+$ programs, that satisfy two further constraints, called *modularity* and *context* constraints. Actually, the modularity constraint can be achieved by a pre-processing of the program which extends the original PGL$^+$ program with valid clauses by means of the GR and FU inference rules. This is indeed the first step of an efficient and complete proof procedure for PGL$^+$ programs satisfying what we call *context* constraint. The idea is that in a PGL$^+$ program satisfying the context constraint, the use of the SU and MP inference rules is enough to attain a degree of deduction equal to the degree of possibilistic entailment. Then, the second step of the proof procedure is based on the MP, SU, UN and IN rules and translates a PGL$^+$ program satisfying the modularity constraint into a semantically equivalent set of 1-weighted facts, whenever the program satisfied the context constraint. The final step is a deduction step, based on the SU rule, which computes the maximum degree of possibilistic entailment of a goal from the equivalent set of 1-weighted facts.

3 The DePGL$^+$ programming language

As already pointed out our objective is to extend the P-DeLP programming language through the use of PGL$^+$ in order to incorporate fuzzy constants and fuzzy propositional variables; we will refer to this extension as *Defeasible* PGL$^+$, DePGL$^+$ for short. To this end, the base language of P-DeLP [7] will be extended with fuzzy constants and fuzzy propositional variables, and arguments will have an attached necessity measure associated with the supported conclusion.

The DePGL$^+$ language $\mathcal{L}$ is defined over PGL$^+$ atomic formulas together with the connectives $\{\sim, \wedge, \leftarrow\}$. The symbol $\sim$ stands for *negation*. A *literal* $L \in \mathcal{L}$ is a PGL$^+$ atomic formula or its negation. A *rule* in $\mathcal{L}$ is a formula of the form $Q \leftarrow L_1 \wedge \ldots \wedge L_n$, where $Q, L_1, \ldots, L_n$ are literals in $\mathcal{L}$. When $n = 0$, the formula $Q \leftarrow$ is called a *fact* and simply written as Q. In the following, capital and lower case letters will denote literals and atoms in $\mathcal{L}$, respectively.

In argumentation frameworks, the negation connective allows to represent conflicts among pieces of information. In the frame of DePGL$^+$, the handling of negation deserves some explanation. As for negated propositional variables $\sim p$, the negation connective $\sim$ will not be considered as a proper Gödel negation, rather $\sim p$ will be treated as another propositional variable p', with a particular status with respect to p, since it will be only used to detect contradictions at the syntactical level. On the other hand, negated literals of the form $\sim p(A)$, where A is a fuzzy constant, will be handled in the following way.

As previously mentioned, fuzzy constants are *disjunctively* interpreted in PGL$^+$. For instance, consider the formula *speed*(*low*). In each interpretation $I = (U, i, m)$, the predicate *speed* is assigned a *unique* element $i(speed)$ of the corresponding domain. If *low* denotes a crisp interval of rpm's, say $[0, 2000]$, then *speed*(*low*) will be true iff such element belongs to this interval, i.e. iff $i(speed) \in [0, 2000]$. Now, the negated formula $\sim speed(low)$ is interpreted as "$\neg[\exists x \in low$ such that the engine speed is $x]$", or equivalently, "$\forall x \in low$, x does not correspond with the engine speed". But due to the disjunctive interpretation, $\sim speed(low)$ is false iff $speed(\neg low)$ is true, where $\neg low$ denotes the complement of the interval $[0, 2000]$ in the corresponding domain. Then, given a context $\mathcal{I}_{U,m}$, this leads us to understand a negated literal $\sim p(A)$ as another positive literal $p(\neg A)$, where the fuzzy constant $\neg A$ denotes the (fuzzy) complement of A, that is, where $\mu_{m(\neg A)}(u) = n(\mu_{m(A)}(u))$, for some suitable negation function n (usually $n(x) = 1 - x$).

Therefore, given a context $\mathcal{I}_{U,m}$, using the above interpretations of the negation, and interpreting the DePGL$^+$ arrow $\leftarrow$ as the PGL$^+$ implication $\rightarrow$, we can actually transform a DePGL$^+$ program P into a PGL$^+$ program $\tau(P)$, and then apply the deduction machinery of PGL$^+$ on $\tau(P)$ for automated proof purposes. Hence, if the PGL$^+$ program $\tau(P)$ is non-recursive and satisfiable, and the context $\mathcal{I}_{U,m}$ together with the negation function n enjoys the PGL$^+$ context constraint, the PGL$^+$ proof procedure can be used to determine the maximum degree of possibilistic entailment, i.e. $\|\tau(L)\|_{\tau(P)} = |\tau(L)|_{\tau(P)}$ for all literal $L \in \mathcal{L}$.

From now on and for the sake of a simpler notation, given a context $\mathcal{I}_{U,m}$, a negation function n and the transformation τ of DePGL$^+$ clauses into PGL$^+$ we shall write $\Gamma \vdash_\tau (\varphi, \alpha)$ to denote $\tau(\Gamma) \vdash \tau((\varphi, \alpha))$, being Γ and (φ, α) DePGL$^+$ clauses. Moreover, we shall consider that the negation function n is implicitly determined by each context $\mathcal{I}_{U,m}$, i.e. the function m will map both fuzzy constants A and their complement (negation) $\neg A$.

4 Arguments in DePGL$^+$

In the last sections we formalized the many-valued and the possibilistic semantics of the underlying logic of DePGL$^+$. In this section we formalize the procedural mechanism for building arguments in DePGL$^+$.

We distinguish between *certain* and *uncertain*

DePGL$^+$ clauses. A DePGL$^+$ clause (φ, α) will be referred as certain when $\alpha = 1$ and uncertain, otherwise. Given a context $\mathcal{I}_{U,m}$, a set of DePGL$^+$ clauses Γ will be deemed as *contradictory*, denoted $\Gamma \vdash_\tau \bot$, if

(i) either $\Gamma \vdash_\tau (q, \alpha)$ and $\Gamma \vdash_\tau (\sim q, \beta)$, with $\alpha > 0$ and $\beta > 0$, for some atom q in $\mathcal{L}$,

(ii) or $\Gamma \vdash_\tau (p(A), \alpha)$ with $\alpha > 0$, for some predicate p and some fuzzy constant A such that $m(A)$ is non-normalized.

Notice that in the latter case, $\tau(\Gamma)$ is not satisfiable and there exist $\Gamma_1 \subset \tau(\Gamma)$ and $\Gamma_2 \subset \tau(\Gamma)$ such that Γ_1 and Γ_2 are satisfiable and $|p(B)|_{\Gamma_1} > 0$ and $|p(C)|_{\Gamma_2} > 0$, with $A = B \cap C$.

Example 1 *Consider the set of clauses* $\Gamma = \{ (p(A) \leftarrow q, 0.5),\ (p(B) \leftarrow q \wedge r, 0.3),\ (q, 0.8),\ (r, 1) \}$. *Then,* $\Gamma \vdash_\tau (p(A), 0.5)$ *and* $\Gamma \vdash_\tau (p(B), 0.3)$, *and, by the IN inference rule,* $\Gamma \vdash_\tau (p(A \cap B), 0.3)$. *Hence, in a particular context* $\mathcal{I}_{U,m}$, Γ *is contradictory as soon as* $m(A) \cap m(B)$ *is a non-normalized fuzzy set whereas, for instance,* $\Gamma \backslash \{(r, 1)\}$ *is satisfiable.*

A DePGL$^+$ program is a set of clauses in $\mathcal{L}$ in which we distinguish certain from uncertain information. As additional requirement, certain knowledge is required to be non-contradictory and the corresponding PGL$^+$ program [2] is required to satisfy the modularity constraint [2, 3]. Formally: Given a context $\mathcal{I}_{U,m}$, a *DePGL$^+$ program* $\mathcal{P}$ is a pair (Π, Δ), where Π is a non-contradictory finite set of certain clauses, Δ is a finite set of uncertain clauses, and $\tau(\Pi \cup \Delta)$ satisfies the modularity constraint.

The requirement of the modularity constraint of a DePGL$^+$ program ensures that all (explicit and hidden) clauses of programs are considered. Indeed, since fuzzy constants are interpreted as (flexible) restrictions on an existential quantifier, atomic formulas clearly express disjunctive information. For instance, when $A = \{a_1, \ldots, a_n\}$, $p(A)$ is equivalent to the disjunction $p(a_1) \vee \cdots \vee p(a_n)$. Then, when parts of this (hidden) disjunctive information occur in the body of several program formulas

[2] We consider a set of DePGL$^+$ clauses P as a set of PGL$^+$ clauses $\tau(P)$, where τ stands for transformations discussed in the previous section.

(1) $(\sim fuel\_ok \leftarrow pump\_clog, 1)$
(2) $(pump\_fuel \leftarrow sw1, 0.6)$
(3) $(fuel\_ok \leftarrow pump\_fuel, 0.85)$
(4) $(pump\_oil \leftarrow sw2, 0.8)$
(5) $(oil\_ok \leftarrow pump\_oil, 0.8)$
(6) $(engine\_ok \leftarrow fuel\_ok \wedge oil\_ok, 0.6)$
(7) $(\sim engine\_ok \leftarrow temp(high), 0.95)$
(8) $(\sim oil\_ok \leftarrow temp(high), 0.9)$
(9) $(pump\_clog \leftarrow pump\_fuel \wedge speed(low), 0.7)$
(10) $(speed(low) \leftarrow sw2, 0.8)$
(11) $(\sim speed(low) \leftarrow sw2, sw3, 0.8)$
(12) $(fuel\_ok \leftarrow sw3, 0.9)$
(13) $(sw1, 1)$
(14) $(sw2, 1)$
(15) $(sw3, 1)$
(16) $(temp(around\_31), 0.85)$

Figure 1: DePGL$^+$ program $\mathcal{P}_{eng}$ (example 2)

we also have to consider all those new formulas that can be obtained through a completion process of the program which is based on the RE and FU inference rules.

Example 2 *(Adapted from [7]) 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. The agent's generic (and incomplete) knowledge about how this engine works is the following:*

- *If the pump is clogged, then the engine gets no fuel.*
- *When sw1 is on, apparently fuel is pumped properly.*
- *When fuel is pumped, fuel seems to work ok.*
- *When sw2 is on, usually oil is pumped.*
- *When oil is pumped, usually it works ok.*
- *When there is oil and fuel, normally the engine is ok.*
- *When there is heat, the engine is almost sure not ok.*
- *When there is heat, normally there are oil problems.*
- *When fuel is pumped and speed is low, there are reasons to believe that the pump is clogged.*
- *When sw2 is on, usually speed is low.*
- *When sw2 and sw3 are on, usually speed is not low.*
- *When sw3 is on, normally fuel is ok.*

Suppose also that the agent knows some particular facts about the current state of the engine:

- *sw1, sw2 and sw3 are on, and*
- *the temperature is around 31° C.*

This knowledge can be modelled by the program $\mathcal{P}_{engine}$ *shown in Fig. 1. Note that uncertainty is assessed in terms of different necessity degrees and vague knowledge is represented by means of fuzzy constants (low, around_31, high).*

Next we will introduce the notion of *argument*

in DePGL$^+$. Informally, a argument $\mathcal{A}$ is a tentative proof (as it relies to some extent on uncertain, possibilistic information) supporting a given literal (goal) Q with a necessity degree α.

Definition 3 *Given a context $\mathcal{I}_{U,m}$ and a DePGL$^+$ 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) $\Pi \cup \mathcal{A} \vdash_\tau^* (Q, \alpha)$;
(2) $\Pi \cup \mathcal{A}$ is non contradictory; and
(3) $\mathcal{A}$ is minimal wrt set inclusion, i.e. there is no $\mathcal{A}_1 \subset \mathcal{A}$ satisfying (1) and (2).

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 $\mathcal{A}$.

Given a context $\mathcal{I}_{U,m}$, the set of arguments for a DePGL$^+$ program $\mathcal{P} = (\Pi, \Delta)$ can be found by iterative application of the following construction rules:

1) Building arguments from facts (INTF):

$$\frac{(Q,1)}{\langle \emptyset, Q, 1\rangle} \qquad \frac{(Q,\alpha),\ \Pi \cup \{(Q,\alpha)\} \not\vdash_\tau \bot,\ \alpha < 1}{\langle \{(Q,\alpha)\}, Q, \alpha\rangle}$$

for any $(Q, 1) \in \Pi$ and any $(Q, \alpha) \in \Delta$.

2) Building Arguments by SU (SUA):

$$\frac{\langle \mathcal{A}, p(A), \alpha\rangle}{\langle \mathcal{A}, p(B), \min(\alpha, N(m(B) \mid m(A)))\rangle}$$

if $N(m(B) \mid m(A)) \neq 0$.

3) Building Arguments by UN (UNA):

$$\frac{\langle \mathcal{A}, p(A), \alpha\rangle}{\langle \mathcal{A}, p(A'), 1\rangle}$$

where $m(A') = \max(1 - \alpha, m(A))$.

4) Building Arguments by IN (INA):

$$\frac{\langle \mathcal{A}_1, p(A), \alpha\rangle,\ \langle \mathcal{A}_2, p(B), \beta\rangle,\ \Pi \cup \mathcal{A}_1 \cup \mathcal{A}_2 \not\vdash_\tau \bot}{\langle \mathcal{A}_1 \cup \mathcal{A}_2, p(A \cap B), \min(\alpha, \beta)\rangle}$$

5) Building Arguments by MP (MPA):

$$\frac{\langle \mathcal{A}_1, L_1, \alpha_1\rangle\ \ \langle \mathcal{A}_2, L_2, \alpha_2\rangle\ \ \ldots\ \langle \mathcal{A}_k, L_k, \alpha_k\rangle\ \ (L_0 \leftarrow L_1 \wedge L_2 \wedge \ldots \wedge L_k, 1),\ \Pi \cup \bigcup_{i=1}^k \mathcal{A}_i \not\vdash_\tau \bot}{\langle \bigcup_{i=1}^k \mathcal{A}_i, L_0, \beta\rangle}$$

for any certain rule $(L_0 \leftarrow L_1 \wedge L_2 \wedge \ldots \wedge L_k, 1) \in \Pi$, with $\beta = min(\alpha_1, \ldots, \alpha_k)$.

$$\frac{\langle \mathcal{A}_1, L_1, \alpha_1\rangle\ \ \langle \mathcal{A}_2, L_2, \alpha_2\rangle\ \ \ldots\ \langle \mathcal{A}_k, L_k, \alpha_k\rangle\ \ (L_0 \leftarrow L_1 \wedge L_2 \wedge \ldots \wedge L_k, \gamma),\ \text{with } \gamma < 1,\ \Pi \cup \{(L_0 \leftarrow L_1 \wedge L_2 \wedge \ldots \wedge L_k, \gamma)\} \cup \bigcup_{i=1}^k \mathcal{A}_i \not\vdash_\tau \bot}{\langle \bigcup_{i=1}^k \mathcal{A}_i \cup \{(L_0 \leftarrow L_1 \wedge L_2 \wedge \ldots \wedge L_k, \gamma)\}, L_0, \beta\rangle}$$

for any weighted rule $(L_0 \leftarrow L_1 \wedge L_2 \wedge \ldots \wedge L_k, \gamma) \in \Delta$, with $\beta = min(\alpha_1, \ldots, \alpha_k, \gamma)$.

The basic idea with the argument construction procedure is to keep a trace of the set $\mathcal{A} \subseteq \Delta$ of all uncertain information in the program $\mathcal{P}$ used to derive a given goal Q with necessity degree α. Appropriate preconditions ensure that the proof obtained always ensures the non-contradictory constraint of arguments wrt the certain knowledge Π of the program. Given a context $\mathcal{I}_{U,m}$ and a DePGL$^+$ 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 wrt the set Π in $\mathcal{P}$. Rules SUA and UNA accounts for semantical unification and resolving uncertainty, respectively. As both rules do not combine new uncertain knowledge, we do not need to check the non-contradictory constraint. Rule INA applies intersection between previously argumented goals. Therefore, we must ensure that the resulting intersection enjoys the non-contradictory constraint wrt Π. Rules MPA account for the use of modus ponens, both with certain and defeasible rules. Note they assume the existence of an argument for every literal in the antecedent of the rule. Then, in a 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 $(L_0 \leftarrow L_1 \wedge L_2 \wedge \ldots \wedge L_k, \gamma)$ when $\gamma < 1$.

Example 4 *Consider the program $\mathcal{P}_{eng}$ in Example 2, where $temp(\cdot)$ is a unary predicate of type* degrees*, $speed(\cdot)$ is a unary predicate of type* rpm*,* heat *and* around_31 *are two object constants of type* degrees*, and* low *is an object constant of type* rpm*. Further, consider the context $\mathcal{I}_{U,m}$ such that:*

- $U = \{U_{\text{degrees}} = [-100, 100]\ ^oC,$
  $U_{\text{rpm}} = [0, 200]\}$;
- $m(high) = [28, 30, 100, 100]$[3],
  $m(around\_31) = [26, 31, 31, 36]$,
  $m(low) = [10, 15, 25, 30]$, *and*
  $m(\neg low) = 1 - m(low)$.

The following arguments can be derived from $\mathcal{P}_{eng}$*:*

1. *The argument* $\langle \mathcal{B}, fuel\_ok, 0.6\rangle$ *can be derived as follows:*
   i) $\langle \emptyset, sw1, 1\rangle$ *from (13) via INTF.*
   ii) $\langle \mathcal{B}', pump\_fuel, 0.6\rangle$ *from (2) and i) via MPA.*
   iii) $\langle \mathcal{B}, fuel\_ok, 0.6\rangle$ *from (3) and ii) via MPA.*

   where $\mathcal{B}' = \{(pump\_fuel \leftarrow sw1, 0.6)\}$ *and* $\mathcal{B} = \mathcal{B}' \cup \{(fuel\_ok \leftarrow pump\_fuel, 0.85)\}$.
2. *Similarly, the argument* $\langle \mathcal{C}_1, oil\_ok, 0.8\rangle$ *can be derived using the rules (15), (4) and (5) via INTC, MPA, and MPA respectively, with:* $\mathcal{C}_1 = \{(pump\_oil \leftarrow sw2, 0.8);$ $(oil\_ok \leftarrow pump\_oil, 0.8)\}$.
3. *The argument* $\langle \mathcal{A}_1, engine\_ok, 0.6\rangle$ *can be derived as follows:*
   i) $\langle \mathcal{B}, fuel\_ok, 0.6\rangle$ *as shown above.*
   ii) $\langle \mathcal{C}_1, oil\_ok, 0.8\rangle$ *as shown above.*
   iii) $\langle \mathcal{A}_1, engine\_ok, 0.6\rangle$ *from i), ii), (6) via MPA.*

   with $\mathcal{A}_1 = \{(engine\_ok \leftarrow fuel\_ok \wedge oil\_ok, 0.6)\}$ $\cup\ \mathcal{B} \cup \mathcal{C}_1$. *Note that* $\langle \mathcal{C}_1, oil\_ok, 0.8\rangle$ *and* $\langle \mathcal{B}, fuel\_ok, 0.6\rangle$ *are subarguments of* $\langle \mathcal{A}_1, engine\_ok, 0.6\rangle$.
4. *Also the argument* $\langle \mathcal{C}_2, \sim oil\_ok, 0.8\rangle$, *where* $\mathcal{C}_2 = \{(temp(around\_31), 0.85), (\sim oil\_ok \leftarrow temp(high), 0.9)\}$, *can be derived as follows, where* $temp31$ *denotes* $temp(around\_31)$*:*
   i) $\langle \{(temp31,\ 0.85)\}, temp31,\ 0.85\rangle$, *from (16) via INTF.*
   ii) $\langle \{(temp31,\ 0.85)\}, temp(high), 0.8\rangle$, *from i) via SUA, taking into account that* $N(high \mid around\_31) = 0.8$.
   iii) $\langle \mathcal{C}_2, \sim oil\_ok, 0.8\rangle$, *from i), ii), (6) via MPA.*
5. *Similarly,* $\langle \mathcal{A}_2, \sim engine\_ok, 0.8\rangle$ *can be derived using the rules (16) and (7) via INTF, SUA, and MPA, with*
   $\mathcal{A}_2 = \{(temp(around\_31),\ 0.85);$
   $(\sim engine\_ok \leftarrow temp(high), 0.95)\}$.

5 Counter-argumentation and defeat in DePGL$^+$

Given a program and a particular context, it can be the case that there exist conflicting arguments for one literal and its negation. For instance, in the above example, $\langle \mathcal{A}_1, engine\_ok, 0.6\rangle$ and $\langle \mathcal{A}_2, \sim engine\_ok, 0.8\rangle$, and $\langle \mathcal{C}_1, oil\_ok, 0.8\rangle$ and $\langle \mathcal{C}_2, \sim oil\_ok, 0.8\rangle$, and thus, the program $\mathcal{P}_{eng}$ considering the context $\mathcal{I}_{U,m}$ is contradictory. Therefore, it is necessary to define a formal framework for solving conflicts among arguments in DePGL$^+$. This is formalized next by the notions of counterargument and defeat, based on the same ideas used in P-DeLP [7] but incorporating the treatment of fuzzy constants.

Definition 5 (Counterargument) *Let* $\mathcal{P}$ *be a DePGL$^+$ 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 wrt* $\mathcal{P}$. *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*)* $\langle S, Q, \beta\rangle$ *of* $\langle \mathcal{A}_2, Q_2, \alpha_2\rangle$ *such that* $Q = \sim Q_1$.

Since arguments rely on uncertain and hence defeasible information, conflicts among arguments may be resolved by comparing their strength and deciding which argument is defeated by which one. Therefore, a notion of defeat amounts to establish a *preference criterion* on conflicting arguments. In our framework, following [7], it seems natural to define it on the basis of necessity degrees associated with arguments.

Definition 6 (Defeat) *Let* $\mathcal{P}$ *be a DePGL$^+$ 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 in* $\mathcal{P}$. *We will say that* $\langle \mathcal{A}_1, Q_1, \alpha_1\rangle$ defeats $\langle \mathcal{A}_2, Q_2, \alpha_2\rangle$ *(or equivalently* $\langle \mathcal{A}_1, Q_1, \alpha_1\rangle$ *is a* defeater *for* $\langle \mathcal{A}_2, Q_2, \alpha_2\rangle$*) iff*

(1) Argument $\langle \mathcal{A}_1, Q_1, \alpha_1\rangle$ *counterargues argument* $\langle \mathcal{A}_2, Q_2, \alpha_2\rangle$ *with disagreement subargument* $\langle \mathcal{A}, Q, \alpha\rangle$*; and*

(2) Either it holds that $\alpha_1 > \alpha$, *in which case* $\langle \mathcal{A}_1, Q_1, \alpha_1\rangle$ *will be called a* proper defeater *for* $\langle \mathcal{A}_2, Q_2, \alpha_2\rangle$, *or* $\alpha_1 = \alpha$, *in which case* $\langle \mathcal{A}_1, Q_1, \alpha_1\rangle$ *will be called a* blocking defeater *for* $\langle \mathcal{A}_2, Q_2, \alpha_2\rangle$.

Following Examples 2 and 4, we have that argument $\langle \mathcal{A}_2, \sim engine\_ok, 0.8\rangle$ is a defeater of argument $\langle \mathcal{A}_1, engine\_ok, 0.6\rangle$ while $\langle \mathcal{C}_2, \sim oil\_ok, 0.8\rangle$ is a blocking defeater of $\langle \mathcal{C}_1, oil\_ok, 0.8\rangle$.

[3] We represent a trapezoidal fuzzy set as $[t_1; t_2; t_3; t_4]$, where the interval $[t_1, t_4]$ is the support and the interval $[t_2, t_3]$ is the core.

As in most argumentation systems, the main goal in DePGL$^+$ involves a procedure to determine if a given argument $\langle \mathcal{A}, Q, \alpha \rangle$ is *warranted* (or ultimately accepted) wrt a program $\mathcal{P}$. Intuitively, an argument $\langle \mathcal{A}, Q, \alpha \rangle$ is warranted if

1. it has no defeaters, or
2. every defeater for $\langle \mathcal{A}, Q, \alpha \rangle$ is on its turn defeated by another argument which is warranted.

In P-DeLP this is done by an exhaustive dialectical analysis of all argumentation lines rooted in a given argument (see [7] for details) which can be efficiently performed by means of a top-down algorithm.

However, the situation DePGL$^+$ gets more involved. Indeed, due to the disjunctive interpretation of fuzzy constants and their associated fuzzy unification mechanism, a complete analogous procedure for DePGL$^+$ cannot be applied since new blocking situations between arguments have to be considered. Defining such a procedure is part of our current research work.

6 Conclusions. Future work

As we have shown in this paper, PGL$^+$ constitutes a powerful formalism that can be integrated into an argument-based framework like P-DeLP, allowing to combine uncertainty expressed in possibilistic logic and fuzziness characterized in terms of fuzzy constants and fuzzy propositional variables.

In this paper we have focused on characterizing DePGL$^+$, a formal language that combines features from PGL$^+$ along with elements which are present in most argumentative frameworks (like the notions of argument, counterargument, and defeat). As stated in Section 5, part of our current work is focused on providing a formal characterization of warrant in the context of the proposed framework. We are also analyzing how to characterize an alternative conceptualization of warrant in which different warrant degrees can be attached to formulas on the basis of necessity degrees, extending some concepts suggested in [9]. Research in these directions is currently being pursued.

Acknowledgements

This work was supported by Spanish Projects TIC2003-00950, TIN2004-07933-C03-01/03, TIN2004-07933-C03-03, by Ramón y Cajal Program (MCyT, Spain) and by CONICET (Argentina).

References

[1] T. Alsinet and L. Godo. A complete calculus for possibilistic logic programming with fuzzy propositional variables. In *Proc. of UAI-2000 Conf.*, pp. 1–10, 2000.

[2] T. Alsinet and L. Godo. A proof procedure for possibilistic logic programming with fuzzy constants. In *Proc. of ECSQARU-2001*, LNAI 243, pp. 760–771, 2001.

[3] T. Alsinet. *Logic Programming with Fuzzy Unification and Imprecise Constants: Possibilistic Semantics and Automated Deduction.* Monografies of the IIIA, Num. 15, CSIC, 2003.

[4] L . Amgoud and C. Cayrol. Inferring from inconsistency in preference-based argumentation frameworks. *J. Autom. Reasoning* 29(2):125–169, 2002.

[5] Amgoud, L., and Kaci, S. An argumentation framework for merging conflicting knowledge bases: The prioritized case. In *Proc. of ECSQARU-2005* , LNAI 3571, pp. 527–538, 2005.

[6] C. Chesñevar, A. Maguitman, and R. Loui. Logical Models of Argument. *ACM Computing Surveys*, 32(4):337–383, 2000.

[7] C. Chesñevar, G. Simari, T. Alsinet and L. Godo. A Logic Programming Framework for Possibilistic Argumentation with Vague Knowledge. In *Proc. of UAI 2004*. Banff (Canada), pp. 76–84, July 2004.

[8] D. Dubois, J. Lang, and H. Prade. Possibilistic logic. In (D.Gabbay et al. eds.) *Handbook of Logic in Art. Int. and Logic Prog. (Nonmonotonic Reasoning and Uncertain Reasoning)*, pp. 439–513. Oxford Univ. Press, 1994.

[9] J. L. Pollock. Defeasible reasoning with variable degrees of justification. *Artif. Intell.*, 133(1-2):233–282, 2001.