Modeling Defeasible Argumentation within a Possibilistic Logic Framework with Fuzzy Unification

Teresa Alsinet  Carlos I. Chesñevar  Lluís Godo  Sandra Sandri
Dept. of Computer Science  University of Lleida, Spain  AI Research Institute (IIIA)  Bellaterra, Spain
{tracy,cic}@eps.udl.es  {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: Possibilistic 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 C of object constants (crisp and fuzzy constants), each having its sort; a set Pred of unary regular predicates, each one having a type (a type is a tuple of sorts); and connectives ∧, →. 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 C and the sort of A corresponds to the type of p. Formulas are Horn-rules of the form p₁ ∧ ⋯ ∧ pₖ → q with k ≥ 0, where p₁, ⋯, pₖ, q are atomic formulas. A (weighted) clause is a pair of the form (φ, α), where φ is a Horn-rule and α ∈ [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 x₀ euros, as the degree in which the salary x₀ is considered to be low, i.e. μₜₐₖₐ(x₀). 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 Uσ; a primitive proposition q into a value i(q) ∈ [0, 1]; a predicate p of type (σ) into a value i(p) ∈ Uσ; and an object constant A (crisp or fuzzy constant) of sort σ into a normalized fuzzy set m(A) with membership function μₘ(A) : Uσ → [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(φ), is just i(q) if φ is a primitive proposition q, and it is computed as μₘ(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(p₁ ∧ ⋯ ∧ pₖ → q) = 1 if min(w(p₁), ⋯, w(pₖ)) ≤ w(q), and is w(q) otherwise.

Notice that w(φ) may take any intermediate value between 0 and 1 as soon as φ contains some fuzzy constant and w(φ) 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 Iₓₜₐₖₐₐₘ is just the set {w interpretation | w = (U, i, m)} and, once fixed the context, [φ] denotes the fuzzy set of models for a formula φ defining μₜₐₖₐ(φ) = w(φ), for all w ∈ Iₓₜₐₖₐₐₘ.

Now, in a fixed context Iₓₜₐₖₐₐₘ, a belief state (or possibilistic model) is determined by a normalized possibility distribution on Iₓₜₐₖₐₐₘ, π : Iₓₜₐₖₐₐₘ → [0, 1]. Then, we say that π satisfies a clause (φ, α), written π |= (φ, α), iff the (suitable) necessity measure of the fuzzy set of models of φ with respect to π, denoted N([φ] | π), is indeed at least α. Here, for the sake of soundness preservation, we take

N([φ] | π) = inf w∈Iₓₜₐₖₐₐₘ π(w) ⇒ μₜₐₖₐ(φ)(w),

where ⇒ is the reciprocal of Gödel’s many-valued implication, defined as x ⇒ y = 1 if x ≤ y and x ⇒ 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...
For each context $I_{U,m}$, 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

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

Finally, the notion of proof in $\text{PGL}^+$, denoted by $\vdash$, is deduction by means of the triviality axiom and the $\text{PGL}^+$ inference rules. Then, given a context $I_{U,m}$, the degree of deduction of a goal $\varphi$ 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 $\text{PGL}^+$ 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 $\text{PGL}^+$ 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 $\text{PGL}^+$ programs satisfying what we call context constraint. The idea is that in a $\text{PGL}^+$ 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 $\text{PGL}^+$ 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, \land, \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 $Q \leftarrow L_1 \land \ldots \land L_n$, where $L_1, L_2, \ldots, L_n$ 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 $\text{speed(low)}$. In each interpretation $I = (U, i, m)$, the predicate $\text{speed}$ is assigned a unique element $i(\text{speed})$ of the corresponding domain. If $\text{low}$ denotes a crisp interval of rpm's, say $[0,2000]$, then $\text{speed(low)}$ will be true iff such element belongs to this interval, i.e. iff $i(\text{speed}) \in [0,2000]$. Now, the negated formula $\sim \text{speed(low)}$ is interpreted as $\sim \left[ \exists x \in \text{low} \text{ such that the engine speed is } x \right]$, or equivalently, $\forall x \in \text{low}, x \text{ does not correspond with the engine speed}$. But due to the disjunctive interpretation, $\sim \text{speed(low)}$ is false iff $\text{speed(\neg \text{low})}$ is true, where $\neg \text{low}$ 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 $\mu_{\neg A}(u) = \mu_{\neg A}(u)$, 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 $\tau$ of DePGL$^+$ clauses into PGL$^+$ we shall write $\Gamma \vdash_{\tau} (\varphi, \alpha)$ to denote $\tau(\Gamma) \vdash \tau((\varphi, \alpha))$, being $\Gamma$ and $(\varphi, \alpha)$ 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 \((\varphi, \alpha)\) will be referred as certain when \(\alpha = 1\) and uncertain, otherwise. Given a context \(\mathcal{I}_{U,m}\), a set of DePGL\(^+\) clauses \(\Gamma\) 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 \(\Gamma_1\) and \(\Gamma_2\) 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 \(\Gamma = \{ (p(A) \leftarrow q, 0.5), (p(B) \leftarrow q \land r, 0.3), (q, 0.8), (r, 1) \} \). Then, \(\Gamma \vdash_{\tau} (p(A), 0.5)\) and \(\Gamma \vdash_{\tau} (p(B), 0.3)\), and, by the IN inference rule, \(\Gamma \vdash_{\tau} (p(A \land B), 0.3)\). Hence, in a particular context \(\mathcal{I}_{U,m}\), \(\Gamma\) is contradictory as soon as \(m(A) \cap m(B)\) is a non-normalized fuzzy set whereas, for instance, \(\Gamma \setminus \{(r, 1)\}\) 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 \((\Pi, \Delta)\), where \(\Pi\) is a non-contradictory finite set of certain clauses, \(\Delta\) 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 \(p(a_1) \lor \cdots \lor p(a_n)\). Then, when parts of this (hidden) disjunctive information occur in the body of several program formulas 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 \(su1, su2\) and \(su3\). 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 \(su1\) is on, apparently fuel is pumped properly.
- When fuel is pumped, fuel seems to work ok.
- When \(su2\) 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 \(su2\) is on, usually speed is low.
- When \(su2\) and \(su3\) are on, usually speed is not low.
- When \(su3\) is on, normally fuel is ok.

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

- \(su1, su2\) and \(su3\) are on, and
- the temperature is around 3\(^\circ\) C.

This knowledge can be modelled by the program \(\mathcal{P}_{\text{engine}}\), 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 A is a
tentative proof (as it relies to some extent on
certain, possibilistic information) supporting
given literal (goal) Q with a necessity
degree α.

Definition 3 Given a context I_U,m and a
DePGL⁺ program P = (Π, Δ), a set A ⊆ Δ of un-
certain clauses is an argument for a goal Q with
necessity degree α > 0, denoted ⟨A, Q, α⟩, if:
(1) Π ∪ A ⊢^α (Q, α);
(2) Π ∪ A is non contradictory; and
(3) A is minimal wrt set inclusion, i.e. there is
no A₁ ⊆ A satisfying (1) and (2).

Let ⟨A, Q, α⟩ and ⟨S, R, β⟩ be two arguments.
We will say that ⟨S, R, β⟩ is a subargument of
⟨A, Q, α⟩ iff S ⊆ A. Notice that the goal R
can be a subgoal associated with the goal Q
in the argument A.

Given a context I_U,m, the set of arguments
for a DePGL⁺ program P = (Π, Δ) can be
found by iterative application of the following
construction rules:

1) Building arguments from facts (INTF):

⟨Q, 1⟩
Σ
(Q, α), Π ∪ {Q, α} ⊢ α, α < 1

for any (Q, 1) ∈ Π and any (Q, α) ∈ Δ.

2) Building Arguments by SU (SUA):

⟨A, p(A), α⟩

⟨A, p(B), min(α, N(m(B) | m(A)))⟩

if N(m(B) | m(A)) ≠ 0.

3) Building Arguments by UN (UNA):

⟨A, p(A), α⟩

⟨A, p(A'), 1⟩

where m(A') = max(1 - α, m(A)).

4) Building Arguments by IN (INA):

⟨A₁, p(A₁), α₁⟩, ⟨A₂, p(B), β⟩,
Π ∪ A₁ ∪ A₂ ⊢ α₁

⟨A₁ ∪ A₂, p(A₁ ∩ B), min(α₁, β)⟩

5) Building Arguments by MP (MPA):

⟨A₁, L₁, α₁⟩, ⟨A₂, L₂, α₂⟩, ..., ⟨Aₖ, Lₖ, αₖ⟩
(L₀ ← L₁ ∧ L₂ ∧ ... ∧ Lₖ, 1)
Π ∪ {L₀ ← L₁ ∧ L₂ ∧ ... ∧ Lₖ, γ, 1}

⟨∪ₖ₌₁ Aᵢ, L₀, β⟩

for any certain rule (L₀ ← L₁ ∧ L₂ ∧ ... ∧ Lₖ, 1) ∈ Π,
with β = min(α₁, ..., αₖ).

The basic idea with the argument construction
procedure is to keep a trace of the set
A ⊆ Δ of all uncertain information in the
program P used to derive a given goal Q
with necessity degree α. Appropriate pre-
conditions ensure that the proof obtained al-
ways ensures the non-contradictory constraint
of arguments wrt the certain knowledge Π
of the program. Given a context I_U,m and a
DePGL⁺ program P, rule INTF allows to
construct arguments from facts. An empty argu-
ment can be obtained for any certain fact in
P. An argument concluding an uncertain fact
(Q, α) in P can be derived whenever assuming
(Q, α) is not contradictory wrt the set Π in
P. Rules SUA and UNA accounts for seman-
tical 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 when-
ever no contradiction results when putting to-
gether Π, the sets A₁, ..., Aₖ corresponding to
the arguments for the antecedents of the rule
and the rule (L₀ ← L₁ ∧ L₂ ∧ ... ∧ Lₖ, γ)
when γ < 1.

Example 4 Consider the program P_eng in Ex-
ample 2, where temp(γ) is a unary predicate of
type (degrees), speed(γ) is a unary predicate of
type (rpm), heat and around .31 are two object
constants of type degrees, and low is an object con-
stant of type rpm. Further, consider the context
I_U,m such that:
The following arguments can be derived from $P_{eng}$:

1. The argument $(B, \text{fuel}.\text{ok}, 0.6)$ can be derived as follows:
   - $i)$ $(B, \text{sw}1, 1)$ from (13) via INTF.
   - $ii)$ $(B', \text{pump}.\text{fuel}, 0.6)$ from (2) and $i)$ via MPA.
   - $iii)$ $(B, \text{fuel}.\text{ok}, 0.6)$ from (3) and $ii)$ via MPA.

   where $B' = \{(\text{pump}.\text{fuel} \leftarrow \text{sw}1, 0.6)\}$ and $B = B' \cup \{(\text{fuel}.\text{ok} \leftarrow \text{pump}.\text{fuel}, 0.85)\}$.

2. Similarly, the argument $(C_1, \text{oil}.\text{ok}, 0.8)$ can be derived using the rules (15), (4) and (5) via INTC, MPA, and MPA respectively, with:

   $C_1 = \{(\text{pump}.\text{oil} \leftarrow \text{sw}2, 0.8); (\text{oil}.\text{ok} \leftarrow \text{pump}.\text{oil}, 0.8)\}$.

3. The argument $(A_1, \text{engine}.\text{ok}, 0.6)$ can be derived as follows:
   - $i)$ $(B, \text{fuel}.\text{ok}, 0.6)$ as shown above.
   - $ii)$ $(C_1, \text{oil}.\text{ok}, 0.8)$ as shown above.
   - $iii)$ $(A_1, \text{engine}.\text{ok}, 0.6)$ from $i)$, $ii)$, (6) via MPA.

   with $A_1 = \{(\text{engine}.\text{ok} \leftarrow \text{fuel}.\text{ok} \wedge \text{oil}.\text{ok}, 0.6)\}$
   $B \cup C_1$. Note that $(C_1, \text{oil}.\text{ok}, 0.8)$ and $(B, \text{fuel}.\text{ok}, 0.6)$ are subarguments of $(A_1, \text{engine}.\text{ok}, 0.6)$.

4. Also the argument $(C_2, \text{~oil}.\text{ok}, 0.8)$, where $C_2 = \{(\text{temp}.\text{around}.\text{31}, 0.85); (\text{~oil}.\text{ok} \leftarrow \text{temp}.\text{high}, 0.9)\}$, can be derived as follows, where temp31 denotes temp.(around.31):

   - $i)$ $(\text{temp}.\text{31}, 0.85)$, temp31, 0.85, from (16) via INTF.
   - $ii)$ $(\text{temp}.\text{31}, 0.85); \text{temp}.\text{high}(0.8)$, from $i)$ via SUA, taking into account that
     $\text{N(high \mid around.31) = 0.8}$.
   - $iii)$ $(C_2, \text{~oil}.\text{ok}, 0.8)$, from $i)$, $ii)$, (6) via MPA.

5. Similarly, $(A_2, \text{~engine}.\text{ok}, 0.8)$ can be derived using the rules (16) and (7) via INTF, SUA, and MPA, with $A_2 = \{(\text{temp}.\text{around}.\text{31}, 0.85); (\text{~engine}.\text{ok} \leftarrow \text{temp}.\text{high}, 0.95)\}$.

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, $(A_1, \text{engine}.\text{ok}, 0.6)$ and $(A_2, \text{~engine}.\text{ok}, 0.8)$, and $(C_1, \text{oil}.\text{ok}, 0.8)$ and $(C_2, \text{~oil}.\text{ok}, 0.8)$, and thus, the program $P_{eng}$ considering the context $I_{\text{ur,m}}$ 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 $P$ be a DePGL$^+$ program, and let $(A_1, Q_1, \alpha_1)$ and $(A_2, Q_2, \alpha_2)$ be two arguments wrt $P$. We will say that $(A_1, Q_1, \alpha_1)$ counterargues $(A_2, Q_2, \alpha_2)$ iff there exists a subargument (called disagreement subargument) $(S, Q, \beta)$ of $(A_2, Q_2, \alpha_2)$ such that $Q = \text{~}Q_1$.

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 $P$ be a DePGL$^+$ program, and let $(A_1, Q_1, \alpha_1)$ and $(A_2, Q_2, \alpha_2)$ be two arguments in $P$. We will say that $(A_1, Q_1, \alpha_1)$ defeats $(A_2, Q_2, \alpha_2)$ (or equivalently $(A_1, Q_1, \alpha_1)$ is a defeater for $(A_2, Q_2, \alpha_2)$) iff

1. Argument $(A_1, Q_1, \alpha_1)$ counterargues argument $(A_2, Q_2, \alpha_2)$ with disagreement subargument $(A, Q, \alpha)$; and

2. Either it holds that $\alpha_1 > \alpha$, in which case $(A_1, Q_1, \alpha_1)$ will be called a proper defeater for $(A_2, Q_2, \alpha_2)$, or $\alpha_1 = \alpha$, in which case $(A_1, Q_1, \alpha_1)$ will be called a blocking defeater for $(A_2, Q_2, \alpha_2)$.

Following Examples 2 and 4, we have that argument $(A_2, \text{~engine}.\text{ok}, 0.8)$ is a defeater of argument $(A_1, \text{engine}.\text{ok}, 0.6)$ while $(C_2, \text{~oil}.\text{ok}, 0.8)$ is a blocking defeater of $(C_1, \text{oil}.\text{ok}, 0.8)$.
As in most argumentation systems, the main goal in DePGL* involves a procedure to determine if a given argument \( \langle A, Q, \alpha \rangle \) is warranted (or ultimately accepted) wrt a program \( \mathcal{P} \). Intuitively, an argument \( \langle A, Q, \alpha \rangle \) is warranted if

1. it has no defeaters, or
2. every defeater for \( \langle A, Q, \alpha \rangle \) 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


