Formalizing argumentative reasoning in a possibilistic logic programming setting with fuzzy unification

Teresa Alsinet a,*, Carlos I. Chesñevar b,d, Lluís Godo c, Sandra Sandri c, Guillermo Simari d

a Department of Computer Science, Universitat de Lleida, C/Jaume II 69, 25001 Lleida, Spain
b CONICET (National Council of Scientific and Technical Research), Argentina
c Artificial Intelligence Research Institute (IIIA), CSIC, Campus UAB s/n, 08193 Bellaterra, Spain
d Department of Computer Science & Engineering, Universidad Nacional del Sur, Av. Alem 1253, 8000 Bahia Blanca, Argentina

Available online 27 July 2007

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 the object-language level. In spite of its expressive power, an important limitation in P-DeLP is that imprecise, fuzzy information cannot be expressed in the object language. One interesting alternative for solving this limitation is the use of PGL+, a possibilistic logic over Gödel logic extended with fuzzy constants. Fuzzy constants in PGL+ allow expressing disjunctive information about the unknown value of a variable, in the sense of a magnitude, modelled as a (unary) predicate. The aim of this article is twofold: firstly, we formalize DePGL+, a possibilistic defeasible logic programming language that extends P-DeLP through the use of PGL+ in order to incorporate fuzzy constants and a fuzzy unification mechanism for them. Secondly, we propose a way to handle conflicting arguments in the context of the extended framework.

Keywords: Possibilistic logic; Fuzzy constants; Fuzzy unification; Defeasible argumentation; Warrant computation

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 [14]. Recent developments have been oriented towards integrating argumentation as part of logic programming languages. In this


* Corresponding author. Tel.: +34 973 702 734; fax: +34 973 702 702.

E-mail addresses: tracy@diei.udl.cat (T. Alsinet), cic@cs.uns.edu.ar (C.I. Chesñevar), godo@iiia.csic.es (L. Godo), sandri@iiia.csic.es (S. Sandri), grs@cs.uns.edu.ar (G. Simari).
context, Possibilistic Defeasible Logic Programming (P-DeLP) [18] is a logic programming language which combines features from argumentation theory and logic programming, incorporating the treatment of possibilistic uncertainty at object-language level. Roughly speaking, in P-DeLP degrees of uncertainty help in determining which arguments prevail in case of conflict.

In spite of its expressive power, an important limitation in P-DeLP (as defined in [18]) is that the explicit treatment of imprecise, fuzzy information was not actually performed. Such a possibility is indeed very important to properly represent qualitative, symbolic information about continuous numerical magnitudes. To remedy this problem, in this paper we propose the use of PGL+, a possibilistic logic over Gödel fuzzy logic extended with fuzzy constants. Fuzzy constants in PGL+ provide a suitable means for expressing such a symbolic/numerical interface between (finite) scales of labels and continuous scales of magnitudes represented by (unary) predicates. Indeed, a fuzzy constant is mapped, under a given PGL+ interpretation, to a fuzzy subset of a (possibly continuous) domain of elements, in contradistinction to single elements in the case of usual object constants in predicate logics. For instance, an imprecise statement like “John’s salary is low” can be expressed PGL+ by the formula \( \text{John_salary}(\text{low}) \) where \( \text{John_salary} \) is a predicate and \( \text{low} \) a fuzzy object constant, which will be mapped to a fuzzy set of the (numerical) domain of the variable John’s salary. Notice that this kind of states expresses disjunctive knowledge (mutually exclusive), in the sense that in each interpretation it is natural to require that the predicate \( \text{John_salary}(x) \) be true for one and only one variable assignment to \( x \), say \( u_0 \). Then, in such an interpretation it is also natural to evaluate to what extent \( \text{John_salary}(\text{low}) \) is true as the degree in which the salary \( u_0 \) is considered to be \( \text{low} \). Hence, allowing fuzzy constants in the language leads to treat formulas in a many-valued logical setting (that of Gödel many-valued logic in our framework), as opposed to the bivalued setting within classical possibilistic logic, with the unit interval \([0,1]\) as a set of truth-values.

The aim of this paper is twofold: first to define DePGL+, a possibilistic defeasible logic programming language that extends P-DeLP through the use of PGL+, instead of (classical) possibilistic logic, in order to incorporate fuzzy constants and fuzzy unification, and second to propose a way to handle conflicting arguments in the context of the extended framework. The rest of the article is structured as follows. First, in Section 2 we present an overview of PGL+ and discuss the fundamentals of defeasible argumentation. Then in Section 3 we define the DePGL+ programming language. Sections 4 and 5 focus on the characterization of arguments in DePGL+ and the analysis of the notion of conflict among arguments in the context of our proposal. In Section 6 we discuss some problematic situations that may arise when trying to define the notion of warranted arguments in DePGL+, and propose some solutions. Finally in Sections 7 and 8 we discuss some related work and present the main conclusions we have obtained.

2. Possibilistic logic and argumentation: an overview

In order to make this article self-contained, this section discusses the fundamentals of possibilistic logic and defeasible argumentation, with special emphasis on PGL+ and P-DeLP.

2.1. Possibilistic logic and PGL+

Possibilistic logic [19] 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 the propositional version, possibilistic formulas are pairs \((\varphi, x)\) where \(\varphi\) is a proposition of classical logic and interpreted as specifying a constraint \(N(\varphi) \geq x\) on the necessity measure of \(\varphi\). Possibilistic models are possibility distributions \(\pi : \Omega \rightarrow [0,1]\) on the set of classical (bivalued) interpretations \(\Omega\) which rank them in terms of plausibility: \(w\) is at least as plausible as \(w'\) when \(\pi(w) \geq \pi(w')\). If \(\pi(w) = 1\) then \(w\) is considered as fully plausible, while if \(\pi(w) = 0\) then \(w\) is considered as totally impossible. A possibilistic formula \((\varphi, x)\) is satisfied by \(\pi\), written \(\pi \models (\varphi, x)\) whenever \(N_\pi(\varphi) \geq x\), where \(N_\pi(\varphi) = \inf\{1 - \pi(w)|w(\varphi) = 0\}\).

In [3,4] the authors introduce PGL+, an extension of possibilistic logic allowing to deal with some form of fuzzy knowledge and with an efficient and complete proof procedure for atomic deduction when clauses fulfill two kinds of constraints. Technically speaking, PGL+ is a possibilistic logic defined on top of (a fragment of) Gödel infinitely-valued logic, allowing uncertainty qualification of predicates with imprecise, fuzzy constants, and allowing as well a form of graded unification between them. Next we provide some details.
The basic components of PGL\(^+\) formulas are: a set of primitive propositions (fuzzy propositional variables) \(\text{Var}\); a set \(\mathcal{S}\) of sorts of constants; a set \(\mathcal{C}\) of object constants, each having its sort; a set \(\text{Pred}\) of unary regular predicates, each one having a type; and connectives \(\land, \rightarrow\). An atomic formula is either a primitive proposition from \(\text{Var}\) or of the form \(p(A)\), where \(p\) is a predicate symbol from \(\text{Pred}\). \(A\) 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 \(p_1 \land \cdots \land p_k \rightarrow q\) with \(k \geq 0\), where \(p_1, \ldots, p_k, q\) are atomic formulas. A (weighted) clause is a pair of the form \((\phi, z)\), where \(\phi\) is a Horn-rule and \(z \in [0, 1]\).

Remark. Since variables, quantifiers and function symbols are not allowed, the language of PGL\(^+\) so defined remains in fact propositional. This allows us to consider only unary predicates since statements involving multiple (fuzzy) properties can be always represented in PGL\(^+\) as a conjunction of atomic formulas. For instance, the statement “Mary is young and tall” can be represented in PGL\(^+\) as \(\text{age}_\text{Mary}(\text{young}) \land \text{height}_\text{Mary}(\text{tall})\) instead of using a binary predicate involving two fuzzy constants like \(\text{age}_\& \text{height}_\text{Mary}(\text{young}, \text{tall})\).

A many-valued interpretation for the language is a structure \(w = (U, i, m)\), where \(U = \bigcup_{\sigma \in \mathcal{S}} U_\sigma\) is a collection of non-empty domains \(U_\sigma\), one for each basic sort \(\sigma \in \mathcal{S}\); \(i = (i_{\text{prop}}, i_{\text{pred}})\), where \(i_{\text{prop}}: \text{Var} \rightarrow [0, 1]\) maps each primitive proposition \(q\) into a value \(i_{\text{prop}}(q) \in [0, 1]\) and \(i_{\text{pred}}: \text{Pred} \rightarrow U\) maps a predicate \(p\) of type \(\sigma\) into a value \(i_{\text{pred}}(p) \in U_\sigma\); and \(m: \mathcal{C} \rightarrow [0, 1]^U\) maps an object constant \(A\) of sort \(\sigma\) into a normalized fuzzy set \(m(A)\) on \(U_\sigma\), with membership function \(\mu_{m(A)}: U_\sigma \rightarrow [0, 1]\). Note that for each predicate symbol \(p\), \(i_{\text{pred}}(p)\) is the one and only value of the domain which satisfies \(p\) in that interpretation and that \(m\) prescribes for each constant \(A\) at least one value \(u_0\) of the domain \(U_\sigma\) as fully compatible, i.e. such that \(\mu_{m(A)}(u_0) = 1\).

The truth value of an atomic formula \(\phi\) under an interpretation \(w = (U, i, m)\), denoted by \(w(\phi) \in [0, 1]\), is defined as \(w(\phi) = i_{\text{prop}}(\phi)\) for primitive propositions, and \(w(p(A)) = \mu_{m(A)}(i_{\text{pred}}(p))\) for atomic predicates. The truth evaluation is extended to rules by means of interpreting the \(\land\) connective by the min-conjunction and the \(\rightarrow\) connective by the so-called Gödel’s many-valued implication: \(w(p_1 \land \cdots \land p_k \rightarrow q) = 1\) if \(\min(w(p_1), \ldots, w(p_k)) \leq w(q)\), and \(w(p_1 \land \cdots \land p_k \rightarrow q) = w(q)\) otherwise.

Note that the truth value \(w(\phi)\) will depend not only on the interpretation \(i_{\text{pred}}\) of predicate symbols that \(\phi\) may contain, but also on the fuzzy sets assigned to fuzzy constants by \(m\). 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 is just the set of interpretations \(\mathcal{I}_{U,m} = \{w | w = (U, i, m)\}\) and, once fixed the context, \([\phi]\) denotes the fuzzy set of models for a formula \(\phi\) defining \(\mu_{[\phi]}(w) = w(\phi)\), for all \(w \in \mathcal{I}_{U,m}\).

Now, in a fixed context \(\mathcal{I}_{U,m}\), a belief state (or possibilistic model) is modelled by a normalized possibility distribution on \(\mathcal{I}_{U,m}\). Let \(\pi : \mathcal{I}_{U,m} \rightarrow [0, 1]\) which provides a ranking of interpretations according to their possibility degree. Then, we say that \(\pi\) satisfies a clause \((\phi, z)\), written \(\pi \models (\phi, z)\, \iff\, \text{the (suitable) necessity measure of the fuzzy set of models of } \phi \text{ with respect to } \pi\), denoted \(N(\overline{\phi} | \pi)\), is indeed at least \(z\). Due to different technical reasons (see e.g. [1,5]), the necessity measure adopted for PGL\(^+\) is defined as follows:

\[
N(\overline{\phi} | \pi) = \inf_{w \in \mathcal{I}_{U,m}} \pi(w) = \mu_{[\phi]}(w),
\]

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. [19]). According to this semantics, given a context \(\mathcal{I}_{U,m}\) a formula like

\[
(age_{\text{Peter}(about_{35})}, 0.9)
\]

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

\[
\{(age_{\text{Peter}(about_{35})}), \min(0.9, 1 - \beta) : \beta \in [0, 1]\},
\]

where \([\text{about}_{35}]_\beta\) denotes the \(\beta\)-cut of the fuzzy set \(m(\text{about}_{35})\).

As usual, a set of clauses \(P\) is said to entail another clause \((\phi, z)\), written \(P \models (\phi, z)\), if for every possibilistic model \(\pi\) satisfying all the clauses in \(P\) also satisfies \((\phi, z)\), 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 [1]: 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, always in a given context $\mathcal{I}_{U,m}$, the degree of possibilistic entailment of an atomic formula (or goal) $\varphi$ by a set of clauses $P$, denoted by $\|\varphi\|_\alpha$, is the greatest $\alpha \in [0,1]$ such that $P \models (\varphi, \alpha)$. In [1], it is proved that $\|\varphi\|_\alpha = \inf \{N(\{\varphi\})\alpha \mid P \models \varphi\}$.

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

**Generalized resolution:**

\[
(s \rightarrow q(A), \alpha), \\
\left(\frac{(q(B) \land t \rightarrow r, \beta)}{(s \land t \rightarrow r, \min(\alpha, \beta))}\right) \text{[GR]}, \quad \text{if } m(A) \subseteq m(B).
\]

**Fusion:**

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

**Intersection:**

\[
\left(\frac{(p(A), \alpha), (p(B), \beta)}{(p(A \land B), \min(\alpha, \beta))}\right) \text{[IN]}.
\]

**Resolving uncertainty:**

\[
\left(\frac{(p(A), \alpha)}{(p(A'), \alpha)}\right) \text{[UN], where } m(A') = \max(1 - \alpha, m(A)).
\]

**Semantic unification:**

\[
\left(\frac{(p(A), \alpha)}{(p(B), \min(\alpha, \beta))}\right) \text{[SU], where } \beta = N(m(B)|m(A)).
\]

In the description of the GR and FU rules, we have used $s$ and $t$ to denote an arbitrary conjunction of literals, possibly empty. We have also used above several notation conventions regarding fuzzy constants. Namely, $A \cup B$ denotes a fuzzy constant such that $m(A \cup B) = \max(m(A), m(B))$, $A \cap B$ denotes a fuzzy constant such that $m(A \cap B) = \min(m(A), m(B))$, where min and max are applied point-wisely (also the max in the UN inference rule), and the necessity measure $N(m(A)|m(A))$ is defined as above, i.e. $N(m(B)|m(A)) = \inf_{u \in \mathcal{U}} m(A)(u) \Rightarrow m(B)(u)$, where $A$ and $B$ are fuzzy constants of sort $\sigma$ and $\Rightarrow$ is the reciprocal of Gödel implication function. Remarkable properties of this measure are $N(m(A)|m(A)) = 1$ and $N(m(B)|m(B)) = \max(1 - \alpha, m(B)) = \min(\alpha, N(m(A)|m(B)))$. In the rest of the paper we will also write all these expressions without the explicit reference to the context mapping $m$ when no confusion is possible.

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

\[
\left(\frac{p_1 \land \cdots \land p_n \rightarrow q, \alpha}{(p_1, \beta_1), \ldots, (p_n, \beta_n)}\right) \text{[MP].}
\]

The notion of *proof* in $\mathcal{PGL}^+$, denoted by $\vdash$, is that of deduction by means of the triviality axiom, $(\varphi, 0)$, and the $\mathcal{PGL}^+$ inference rules. Given a context $\mathcal{I}_{U,m}$, the degree of deduction of a goal $\varphi$ from a set of clauses $P$, denoted $|\varphi|_\alpha$, is the greatest $\alpha \in [0,1]$ for which $P \vdash (\varphi, \alpha)$. In [4,1] it is shown that this notion of proof is complete for determining the degree of possibilistic entailment of a goal, i.e. $|\varphi|_\alpha = \|\varphi\|_\alpha$, for non-recursive and satisfiable programs $P$, called $\mathcal{PGL}^+$ 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. For instance, if the original program contains clauses like $(p(A) \rightarrow q, x)$ and $(p(B) \rightarrow q, \beta)$, then the new clause $(p(A \lor B) \rightarrow q, \min(x, \beta))$ would be added in this pre-processing step. 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 satisfies 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.

2.2. Defeasible argumentation and P-DeLP

Defeasible argumentation [14,30] has evolved in the last decade as a successful approach to formalize commonsense reasoning. When a rule supporting a conclusion may be defeated by new information, it is said that such reasoning is defeasible [26,27]. When defeasible reasons or rules are chained to reach a conclusion, we have arguments instead of proofs. Arguments may compete, rebutting each other, so a process of argumentation is a natural result of the search for arguments. Adjudication of competing arguments must be performed, comparing arguments in order to determine what beliefs are ultimately accepted as warranted or justified. Preference among conflicting arguments is defined in terms of a preference criterion which establishes a partial order $\preceq$ among possible arguments; thus, for two arguments $A$ and $B$ in conflict, it may be the case that $A$ is strictly preferred over $B$ ($A > B$), that $A$ and $B$ are equally preferable ($A \succeq B$ and $A \preceq B$) or that $A$ and $B$ are not comparable with each other. Arguments may be defeated by other arguments, which on their turn may be defeated by other arguments, and so on. This prompts a recursive analysis, which is usually modelled by means of a tree structure called dialectical tree or argument tree. When an argument is ultimately accepted after considering all possible defeaters, the argument is said to be warranted or justified.

In the last few years the argumentation community has given particular attention to several extensions of logic programming which have turned out to be computationally manageable for formalizing knowledge representation and argumentative inference. Several approaches have been developed, some of them based on normal logic programming [24], extended logic programming [29], and defeasible logic programming or DeLP [21], among others. The DeLP approach has been particularly attractive in the context of real-world applications, such as recommender systems [17], knowledge management [12] and natural language processing [15]. Possibilistic Defeasible Logic Programming (P-DeLP) is a logic programming framework based on DeLP, and hence combining features from argumentation theory and logic programming which incorporates a treatment of possibilistic uncertainty at the object-language level (see [18] for a full description of P-DeLP).

The language $\mathcal{L}$ of P-DeLP is inherited from the language of logic programming, including the usual notions of atom, literal, rule and fact. In particular, the symbol $\not\in$ stands for (strong) negation. A literal $L \in \mathcal{L}$ is a ground (fuzzy) atom $q$ or a negated ground (fuzzy) atom $\neg q$, where $q$ is a ground (fuzzy) propositional variable. A goal in P-DeLP is any literal $L \in \mathcal{L}$. A program $\mathcal{P}$ in P-DeLP is a set of weighted clauses, where every weighted clause is a pair of the form $(\varphi, x)$, where $\varphi$ is a rule $p \leftarrow q_1, q_2, \ldots, q_k$ or fact $p \leftarrow$ (i.e., a rule with empty antecedent), where $p, q_1, q_2, \ldots, q_k$ are literals, and $x \in [0, 1]$ expresses a lower bound for the necessity degree of $\varphi$. The subset $\Pi_\varphi$ of weighted clauses in $\mathcal{P}$ whose necessity degree is 1 corresponds to certain clauses, and is assumed to be non-contradictory. A set of clauses $\Gamma$ will be deemed as contradictory, denoted $\Gamma \vdash \bot$, if $\Gamma \vdash (q, x)$ and $\Gamma \vdash (\neg q, \beta)$, with $x > 0$ and $\beta > 0$, for some atom $q$ in $\mathcal{L}$.

As in most argument-based logic programming frameworks, in P-DeLP solving a goal $Q$ accounts for finding an argument $\mathcal{A}$ supporting $Q$ which is ultimately accepted or warranted. Given a P-DeLP program $\mathcal{P}$, the notion of an argument $\mathcal{A}$ supporting a literal $Q$ with a necessity degree $x$ (denoted $(\mathcal{A}, Q, x)$) is based inferring $(Q, x)$ from $\mathcal{P}$ using Generalized Modus Ponens as a possibilistic resolution rule. The set $\mathcal{A}$ accounts for the set of weighted clauses from $\mathcal{P}$ with necessity degree in $[x, 1)$ used to derive $(Q, x)$. 

The set of uncertain clauses in a given P-DeLP program $\mathcal{P}$ account for tentative and incomplete information. Hence conflicting arguments may arise. An argument $\langle \mathcal{A}, Q, x \rangle$ may be defeated by another argument $\langle \mathcal{A}', Q', x' \rangle$. The notion of defeat in P-DeLP is associated with determining a sub-argument (sub-proof) $\langle \mathcal{A}', Q', x' \rangle$ in the attacked argument $\langle \mathcal{A}, Q, x \rangle$ such that $\Pi_p \cup \{Q', x'\}, \{R, \beta\}$ is contradictory and $\beta \geq x'$. In this case, the argument $\langle \mathcal{A}, Q, x \rangle$ is called a defeater for $\langle \mathcal{A}, Q, x \rangle$. As defeaters are arguments, they may be in turn defeated by other arguments. This prompts a recursive analysis, associated with solving a goal $Q$ in P-DeLP.

Given a P-DeLP program $\mathcal{P}$, solving a goal $Q_0$ accounts for first finding an argument $\langle \mathcal{A}_0, Q_0, x_0 \rangle$ supporting $(Q_0, x_0)$, and then performing an exhaustive analysis of possible defeaters for $\langle \mathcal{A}_0, Q_0, x_0 \rangle$, defeaters for such defeaters, and so on. Every one of such sequences $\lambda = [\langle \mathcal{A}_0, Q_0, x_0 \rangle, \langle \mathcal{A}_1, Q_1, x_1 \rangle, \ldots, \langle \mathcal{A}_n, Q_n, x_n \rangle, \ldots]$ is called an argumentation line, standing for a dialogue between two parties (a proponent who advances the even-level arguments, starting with the original argument at issue, and an opponent who attacks the proponent’s arguments, by advancing odd-level arguments). If all possible argumentation lines rooted in $\langle \mathcal{A}_0, Q_0, x_0 \rangle$ are of odd length, this implies that every possible dialogue on the basis of the program $\mathcal{P}$ was won by the proponent, and hence the original argument $\langle \mathcal{A}_0, Q_0, x_0 \rangle$ is warranted.

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 [18] will be extended with fuzzy constants and fuzzy propositional variables, while arguments will keep 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 $\sim L \in \mathcal{L}$ is a PGL+ atomic formula or its negation. A rule in $\mathcal{L}$ is a formula of the form $Q \leftarrow L_1 \land \cdots \land L_n$, 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. In what regards 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 $\text{speed}(\text{low})$. In each interpretation $I = (U, i, m)$, the predicate $\text{speed}$ is assigned a unique element $i(\text{speed})$ of the corresponding domain. If $\text{low}$ is interpreted by a crisp interval of rpm’s, say $[0, 2000]$, then $\text{speed}(\text{low})$ will be true in $I$ iff such element $i(\text{speed})$ belongs to this interval, i.e. iff $i(\text{speed}) \in [0, 2000]$. Now, the negated formula $\sim \text{speed}(\text{low})$ is to be interpreted as “not [\exists x \in \text{low} such that the engine speed is $x$]”. Since the elements in $\text{low}$ are disjunctive, under PGL+ interpretations, it amounts to “[\exists x \notin \text{low} such that the engine speed is $x$]”, and thus $\sim \text{speed}(\text{low})$ is true iff $\text{speed}(\text{not-low})$ is true, where $\text{not-low}$ 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 $p(\sim A)$, where the fuzzy constant $\sim A$ denotes the (fuzzy) complement of $A$, that is, where $m(\sim A)(u) = n(p_m(A)(u))$, for some suitable negation function $n$. One usually takes $n(x) = 1 - x$, but any other is also allowed. Indeed, we shall consider that the negation function $n$ is implicitly determined by each context $\mathcal{I}_{U, m}$, i.e. the function $m$ will interpret both fuzzy constants $A$ and their complement (negation) $\sim A$.

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, denoted as $\tau(P)$, and then, we can apply the deduction machinery of PGL+ on $\tau(P)$ for automated proof purposes. From now on and for the sake of a simpler notation, we shall write $\Gamma_{\tau,-}(\varphi, z)$ to denote $\tau(\Gamma) \vdash \tau((\varphi, z))$, where the elements in $\Gamma$ and $(\varphi, z)$ are DePGL+ clauses.
4. Arguments in DePGL⁺

In Section 2.1 we have formalized the many-valued and the possibilistic semantics of PGL⁺, 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 \((\varphi, x)\) will be referred as certain when \(x = 1\) and uncertain, otherwise. Given a context \(\mathcal{I}_{U,m}\), a set of DePGL⁺ clauses \(\Gamma\) will be deemed as contradictory, denoted \(\Gamma \vdash \bot\), when

(i) either \(\Gamma_1 \vdash (q, x)\) and \(\Gamma_2 \vdash (\sim q, \beta)\), with \(x > 0\) and \(\beta > 0\), for some atom \(q\) in \(L\),

(ii) or \(\Gamma_1 \vdash (p(A), x)\) with \(x > 0\), 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 \(|p(C)|_{\Gamma_2} > 0\), for some fuzzy constants \(B\) and \(C\) such that \(A = B \cap C\).

**Example 1.** Consider the set of clauses \(\Gamma = \{(q, 0.8), (r, 1), (p(A) \leftarrow q, 0.5), (p(B) \leftarrow q \land r, 0.3)\}\). Then, \(\Gamma_1 \vdash (p(A), 0.5)\) and \(\Gamma_2 \vdash (p(B), 0.3)\), and, by the IN inference rule, \(\Gamma_1 \vdash (p(A \cap B), 0.3)\). 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 \(L\) in which we distinguish certain from uncertain information. As additional requirement, certain knowledge is required to be non-contradictory and the corresponding PGL⁺ program (by means of the transformation \(\tau\)) is required to satisfy the modularity constraint [4,1]. This is formally stated as follows.

**Definition 2 (DePGL⁺ program).** Given a context \(\mathcal{I}_{U,m}\), a DePGL⁺ program \(\mathcal{P}\) is a pair \((\Pi, A)\), where \(\Pi\) is a non-contradictory finite set of certain clauses, \(A\) is a finite set of uncertain clauses, and \(\tau(\Pi \cup A)\) satisfies the modularity constraint.

The requirement of the modularity constraint of a DePGL⁺ program ensures that all (explicit and hidden) program clauses are considered. Indeed, since fuzzy constants are interpreted as (flexible) restrictions on an existential quantifier, atomic formulas clearly express disjunctive information. For instance, within a context \(\mathcal{I}_{U,m}\), when \(m(A) = \{a_1, \ldots, a_n\}\), \(p(A)\) is semantically 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 clauses we also have to consider all those new clauses that can be obtained through a completion process of the program which is based on the GR and FU inference rules.

**Example 3 (Adapted from [18]).** Consider an intelligent agent controlling an engine with three switches \(sw_1, sw_2\) and \(sw_3\). 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 \(sw_1\) is on, apparently fuel is pumped properly.
- When fuel is pumped, fuel seems to work ok.
- When \(sw_2\) 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 very low, there are reasons to believe that the pump is clogged.
- When \(sw_2\) is on, usually speed is low.
- When \(sw_2\) and \(sw_3\) are on, usually speed is not low.
- When \(sw_3\) is on, normally fuel is ok.
Suppose also that the agent knows some particular facts about the current state of the engine:

- \(sw_1\), \(sw_2\) and \(sw_3\) are on,
- the temperature is in the interval \([25,31]^{\circ}C\), and
- the temperature seems to be around \(31^{\circ}C\).

This knowledge can be modelled by the program \(P_{eng}\) shown in Fig. 1. Note that uncertainty is assessed in terms of different necessity degrees while imprecise knowledge is represented by means of fuzzy object constants like \(high\), \(low\), \(very\_low\), \(around\_31\) and \(interval\_25\_31\).

Next we introduce the notion of argument in DePGL\(^+\). Informally, an argument for a literal (goal) \(Q\) with necessity degree \(\alpha\) is a tentative (as it relies to some extent on uncertain, possibilistic information) proof for \(\langle \mathcal{A}, Q, \alpha \rangle\).

**Definition 4 (Argument).** Given a context \(I_U, m\) and a DePGL\(^+\) program \(P = (I, A)\), a set \(\mathcal{A} \subseteq A\) of uncertain clauses is an argument for a goal \(Q\) with necessity degree \(\alpha > 0\), denoted \(\langle \mathcal{A}, Q, \alpha \rangle\), iff:

1. \(I \cup \mathcal{A}\) is non-contradictory;
2. \(\alpha = \sup \{\beta \in [0, 1] | I \cup A \vdash (Q, \beta)\}\), i.e. \(\alpha\) is the greatest degree of deduction of \(Q\) from \(\tau(I \cup \mathcal{A})\), denoted as \(|Q|_{\tau(I \cup \mathcal{A})}\); and
3. \(\mathcal{A}\) is minimal wrt set inclusion, i.e. there is no \(\mathcal{A}_1 \subset \mathcal{A}\) such that \(I \cup \mathcal{A}_1 \vdash (Q, \alpha)\).

**Definition 5 (Subargument).** 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 \(\mathcal{A}\).

Let \(\mathcal{I}_{U,m}\) be a context, let \(P\) be a DePGL\(^+\) program and let \(p\) be a predicate symbol of type \((\sigma)\) appearing in \(P\). Then, in [4,1] it is shown that

\[
|p(A)|_{\tau(P)} = |p(A)|_{\tau(C, 1)} = N(m(A)|m(C)),
\]

where \(A\) and \(C\) are object constants of sort \(\sigma\) and \(C\) is such that, for each \(u \in U_\sigma\), \(\mu_{m(C)}(u) = \inf \{\mu_{m(B)}(u) | B\) object constant of sort \(\sigma\) such that \(P \vdash (p, B, 1)\}\). Thus, the greatest degree of deduction of

\[
\begin{align*}
(1) & \quad \neg fuel\_ok \leftarrow pump\_clog, 1 \\
(2) & \quad pump\_fuel \leftarrow sw1, 0.6 \\
(3) & \quad fuel\_ok \leftarrow pump\_fuel, 0.85 \\
(4) & \quad pump\_oil \leftarrow sw2, 0.8 \\
(5) & \quad oil\_ok \leftarrow pump\_oil, 0.8 \\
(6) & \quad engine\_ok \leftarrow fuel\_ok \land oil\_ok, 0.6 \\
(7) & \quad \neg engine\_ok \leftarrow temp(high), 0.95 \\
(8) & \quad \neg oil\_ok \leftarrow temp(high), 0.9 \\
(9) & \quad pump\_clog \leftarrow pump\_fuel \land speed(very\_low), 0.7 \\
(10) & \quad speed(low) \leftarrow sw2, 0.8 \\
(11) & \quad \neg speed(low) \leftarrow sw2, sw3, 0.8 \\
(12) & \quad fuel\_ok \leftarrow sw3, 0.9 \\
(13) & \quad sw1, 1 \\
(14) & \quad sw2, 1 \\
(15) & \quad sw3, 1 \\
(16) & \quad temp(interval\_25\_31), 1 \\
(17) & \quad temp(around\_31), 0.85 \\
\end{align*}
\]

Fig. 1. DePGL\(^+\) program \(P_{eng}\) (Example 3).
p(A) from τ(P) corresponds to the unification degree between the fuzzy constant A and the most specific fuzzy constant that can be deduced from P with necessity degree 1. Then, in order to compute arguments with the greatest degree of deduction, we need to introduce the notion of canonical argument.

**Definition 6 (Canonical argument).** Let $\mathcal{P}$ be a DePGL$^+$ program and let $\mathcal{I}_{U,m}$ be a context. An argument wrt $\mathcal{P}$ and $\mathcal{I}_{U,m}$ of $\langle \mathcal{A}, Q, z \rangle$ is called canonical if either $Q$ is a propositional variable, or if $Q = p(A)$ then $z = 1$ and there is no fuzzy constant $C$ more specific than $A$ such that $\Pi \cup \mathcal{A} \vdash (p(C), 1)$.

It must be noted that given an argument of the form $\langle \mathcal{A}, (p(A), z) \rangle$, the canonical argument associated with the set calA and predicate $p$ is unique. As we will see later in Sections 5 and 6, the notion of canonical argument will turn to be very useful since it will allow us to restrict the search for conflicting arguments and simplify the process of deciding when an argument is ultimately acceptable or not. In [4,1] an efficient algorithm has been presented for computing the most specific fuzzy constant that can be deduced, for a given predicate symbol, from a set of clauses with necessity degree 1 which is based on the MP, SU, IN and UN inference rules. Consequently, this algorithm can be then used to compute canonical arguments.

The next procedure addresses the important issue of how to build arguments for a DePGL$^+$ program.

**Algorithm 7 (Argument construction procedure).** Given a context $\mathcal{I}_{U,m}$ and a DePGL$^+$ program $\mathcal{P} = (\Pi, A)$, a set $\mathcal{A} \subseteq A$ of uncertain clauses is an argument for a goal $Q$ with necessity degree $z > 0$ wrt $\mathcal{P}$ and $\mathcal{I}_{U,m}$ iff $\mathcal{A}$ and $z$ can be computed by (recursively) applying any of the following construction rules:

1. **Building arguments from facts (INTF):**
   - If $(Q, 1) \in \Pi$
     then $\mathcal{A} = \emptyset$ and $z = 1$
   - If $(Q, \beta) \in A$ and $\Pi \cup \{(Q, \beta)\} \not\vdash_\perp$ and $\Pi \not\vdash_\perp(Q, \gamma)$ for any $\gamma \geq \beta$
     then $\mathcal{A} = \{(Q, \beta)\}$ and $z = \beta$

2. **Building arguments from program rules by applying the modus ponens rule (MPA):**
   - If $(Q \leftarrow L_1 \land \cdots \land L_k, 1) \in \Pi$ and $\langle \mathcal{A}_1, L_1, \beta_1 \rangle, \ldots, \langle \mathcal{A}_k, L_k, \beta_k \rangle$ are arguments and $\Pi \cup \bigcup_{i=1}^k \mathcal{A}_i \not\vdash_\perp$ and there is no $\mathcal{B} \in \bigcup_{i=1}^k \mathcal{A}_i$, such that $\Pi \cup \mathcal{B} \vdash_\perp(Q, \gamma)$ with $\gamma \geq \min(\beta_1, \beta_2, \ldots, \beta_k)$
     then $\mathcal{A} = \bigcup_{i=1}^k \mathcal{A}_i$ and $z = \min(\beta_1, \beta_2, \ldots, \beta_k)$
   - If $(Q \leftarrow L_1 \land L_2 \land \cdots \land L_k, \beta) \in A$ and $\langle \mathcal{A}_1, L_1, \beta_1 \rangle, \ldots, \langle \mathcal{A}_k, L_k, \beta_k \rangle$ are arguments and $\Pi \cup \{(Q \leftarrow L_1 \land L_2 \land \cdots \land L_k, \beta)\} \cup \bigcup_{i=1}^k \mathcal{A}_i \not\vdash_\perp$ and there is no $\mathcal{B} \in \bigcup_{i=1}^k \mathcal{A}_i$, $(Q \leftarrow L_1 \land \cdots \land L_k, \beta)$ such that $\Pi \cup \mathcal{B} \vdash_\perp(Q, \gamma)$ with $\gamma \geq \min(\beta, \beta_1, \beta_2, \ldots, \beta_k)$
     then $\mathcal{A} = \bigcup_{i=1}^k \mathcal{A}_i \cup \{(Q \leftarrow L_1 \land L_2 \land \cdots \land L_k, \beta)\}$ and $z = \min(\beta, \beta_1, \beta_2, \ldots, \beta_k)$

3. **Building arguments from canonical arguments by applying the unification rule (SUa):**
   - If $Q = p(B)$ and $\langle \mathcal{A}_1, p(A), 1 \rangle$ is a canonical argument such that $N(m(B) \mid m(A)) \neq 0$ and there is no $\mathcal{A}_2 \subset \mathcal{A}_1$ such that $\Pi \cup \mathcal{A}_2 \vdash_\perp (p(B), \gamma)$ with $\gamma \geq N(m(B) \mid m(A))$
     then $\mathcal{A} = \mathcal{A}_1$ and $z = N(m(B) \mid m(A))$

4. **Building arguments from canonical arguments by applying the intersection rule (INA):**
   - If $Q = p(C)$ and $\langle \mathcal{A}_1, p(A), 1 \rangle$ and $\langle \mathcal{A}_2, p(B), 1 \rangle$ are a pair of canonical arguments such that $m(C) = m(A) \cap m(B)$ and $\Pi \cup \mathcal{A}_1 \cup \mathcal{A}_2 \not\vdash_\perp$ and there is no $\mathcal{B} \in \mathcal{A}_1 \cup \mathcal{A}_2$ such that $\Pi \cup \mathcal{B} \vdash_\perp (p(C), 1)$
     then $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$ and $z = 1$

The basic idea with the argument construction procedure is to keep a trace of the set $\mathcal{A} \subseteq A$ of all uncertain information in the program $\mathcal{P}$ used to derive a given goal $Q$ with necessity degree $z$ and to ensure that $z = |Q|_\tau(\Pi, \mathcal{I}_{U,m})$. On the one hand, appropriate preconditions ensure that the proof obtained satisfies the non-contradiction constraint of arguments wrt the certain knowledge $\Pi$ of the program and that computed arguments are minimal wrt set inclusion. On the other hand, the completeness results of the PGL$^+$ proof method (see [1]) ensure that necessity degrees computed by means of the MP, SU and IN inference rules after resolving uncertainty on both program facts and new derived facts, correspond to greatest degrees of deduction. Given a context $\mathcal{I}_{U,m}$ and a DePGL$^+$ program $\mathcal{P}$, rule INTF allows to construct arguments from facts. An empty argument

\footnote{That is, the smallest, as membership function, with respect to the point-wise order.}
can be obtained for any certain fact in \( \mathcal{P} \). An argument concluding an uncertain fact \( (Q, z) \) in \( \mathcal{P} \) can be derived whenever assuming \( (Q, z) \) is not contradictory wrt the set \( \mathcal{I} \) in \( \mathcal{P} \) and that \( Q \) cannot be proved from \( \mathcal{I} \) with a necessity degree greater or equal than \( z \). Rule MPA accounts for the use of modus ponens, both with certain and defeasible rules. Note 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 \( \mathcal{I} \), the sets \( \mathcal{A}_1, \ldots, \mathcal{A}_k \) corresponding to the arguments for the antecedents of the rule and the rule \( (Q \leftarrow L_1 \land \cdots \land L_k, \beta) \) when \( \beta < 1 \), and whenever it is strictly necessary to consider all these clauses in order to prove \( Q \) with a greater necessity degree. Rule SUA accounts for semantic unification from canonical arguments; i.e., corresponds to the unification between the fuzzy constant \( B \) and the more specific fuzzy constant that can be deduced from \( \mathcal{A}_1 \) with necessity degree 1. As the rule does not deal with new uncertain knowledge, we do not need to check the non-contradictory constraint. However, it is necessary to ensure that all defeasible information is strictly necessary to derive the goal. In a similar way, rule INA applies intersection between canonical arguments provided that the resulting intersection is non-contradictory wrt \( \mathcal{I} \) and minimal wrt set inclusion.

Note that we cannot ensure that arguments with necessity degree 1 are canonical arguments. The following proposition establishes the relationship between arguments and canonical arguments.

**Proposition 8.** Let \( \mathcal{I}_{U,m} \) be a context, let \( \mathcal{P} = (\mathcal{I}, \Delta) \) be a DePGL* program and let \( \mathcal{A} \subseteq \Delta \) be a set of uncertain clauses. If \( \langle \mathcal{A}, p(A), z \rangle \) is an argument then there exists one, and only one, fuzzy constant \( C \) such that \( \langle \mathcal{A}, p(C), 1 \rangle \) is a canonical argument.

**Proof.** On the one hand, if \( \langle \mathcal{A}, p(A), z \rangle \) is an argument then \( z = \sup \{ \beta \in [0, 1] | \mathcal{I} \cup \mathcal{A} \vdash_R \beta \} \). Therefore, as we proved in [1], \( \mathcal{I} \cup \mathcal{A} \vdash_R \beta \) and, by the soundness of the UN inference rule, \( \mathcal{I} \cup \mathcal{A} \vdash_R (p(B), 1) \) where \( B \) is a fuzzy constant such that \( m(B) = \max(1 - z, m(A)) \). Hence, we can ensure that \( \mathcal{B} = \{ B \} \) object constant \( \mathcal{I} \cup \mathcal{A} \vdash_R (p(B), 1) \) is a non-empty set, and thus, we can safely define \( C \) as the most specific fuzzy constant that can be deduced from \( \mathcal{I} \cup \mathcal{A} \) with necessity degree 1.

On the other hand, if \( \langle \mathcal{A}, p(A), z \rangle \) is an argument then \( \mathcal{A} \) is minimal wrt set inclusion, and thus, for all \( \mathcal{A}_1 \subseteq \mathcal{A} \), \( p(A) \mid_{\mathcal{I} \cup \mathcal{A}_1} < z \) and, by the completeness of PGL⁺ semantics, \( p(B) \mid_{\mathcal{I} \cup \mathcal{A}_1} < z \). Therefore, by the PGL⁺ semantics, \( p(B) \mid_{\mathcal{I} \cup \mathcal{A}_1} < 1 \) where \( B \) is a fuzzy constant such that \( m(B) = \max(1 - z, m(A)) \). Hence, as \( C \) is either \( B \) or is more specific than \( B \), \( p(C) \mid_{\mathcal{I} \cup \mathcal{A}_1} < 1 \), and thus, \( p(C) \mid_{\mathcal{I} \cup \mathcal{A}_1} < 1 \) for all \( \mathcal{A}_1 \subseteq \mathcal{A} \). \( \square \)

**Example 9.** Consider the program \( \mathcal{P}_{\text{eng}} \) in Example 3, where \( \text{temp}(\cdot) \) is a unary predicate of type \( (\text{degrees}) \), \( \text{speed}(\cdot) \) is a unary predicate of type \( (\text{rpm}) \), “high”, “interval_25_31” and “around_31” are object constants of sort \( \text{degrees} \), and “very_low” and “low” are object constants of sort \( \text{rpm} \). Further, consider the context \( \mathcal{I}_{U,m} \) such that:

\[
U = \{U_{\text{degree}} = [-100, 100], U_{\text{rpm}} = [0, 200]\};
\]

\[
m(\text{high}) = [28, 30, 100, 100],^2
\]

\[
m(\text{interval}_{25,31}) = [25, 25, 31, 31],
\]

\[
m(\text{around}_{31}) = [26, 31, 31, 36],
\]

\[
m(\text{very_low}) = [5, 15, 15, 25],
\]

\[
m(\text{low}) = [10, 15, 25, 30],
\]

\[
m(-\text{low}) = 1 - m(\text{low}).
\]

Remark that, for this particular context, the corresponding PGL⁺ program satisfies the modularity constraint. Then, the following arguments can be derived from \( \mathcal{P}_{\text{eng}} \):

1. The argument \( \langle \mathcal{A}_1, \text{fuel ok}, 0.6 \rangle \) can be derived as follows:
   (i) \( \langle \emptyset, \text{sw1}, 1 \rangle \) from rule (13) via INTF.
   (ii) \( \langle \mathcal{A}, \text{pump fuel}, 0.6 \rangle \) from rule (2) and (i) via MPA.
   (iii) \( \langle \mathcal{A}_1, \text{fuel ok}, 0.6 \rangle \) from rule (3) and (ii) via MPA.

^2 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.
where
\[ D = \{ (\text{pump\_fuel} \leftarrow \text{sw1}, 0.6) \}, \]
\[ B_1 = D \cup \{ (\text{fuel\_ok} \leftarrow \text{pump\_fuel}, 0.85) \}. \]

(2) Similarly, the argument \( \langle \mathcal{A}_1, \text{oil\_ok}, 0.8 \rangle \) can be derived using the rules (15), (4) and (5) via INTF, MPA, and MPA respectively, with \( \mathcal{A}_1 = \{ (\text{pump\_oil} \leftarrow \text{sw2}, 0.8); (\text{oil\_ok} \leftarrow \text{pump\_oil}, 0.8) \}. \)

(3) The argument \( \langle \mathcal{A}_1, \text{engine\_ok}, 0.6 \rangle \) can be derived as follows:
(i) \( \langle B_1, \text{fuel\_ok}, 0.6 \rangle \) as shown above.
(ii) \( \langle \mathcal{A}_1, \text{oil\_ok}, 0.8 \rangle \) as shown above.
(iii) \( \langle \mathcal{A}_1, \text{engine\_ok}, 0.6 \rangle \) from (i), (ii) and the rule (6) via MPA, with \( \mathcal{A}_1 = \{ (\text{engine\_ok} \leftarrow \text{fuel\_ok} \land \text{oil\_ok}, 0.6) \} \cup B_1 \cup \mathcal{A}_1. \)

Note that \( \langle \mathcal{A}_1, \text{oil\_ok}, 0.8 \rangle \) and \( \langle B_1, \text{fuel\_ok}, 0.6 \rangle \) are subarguments of \( \langle \mathcal{A}_1, \text{engine\_ok}, 0.6 \rangle \).

(4) One can also derive the argument \( \langle \mathcal{A}_2, \sim \text{oil\_ok}, 0.8 \rangle \), where \( \mathcal{A}_2 = \{ (\text{temp(around}, 31), 0.85), (\sim \text{oil\_ok} \leftarrow \text{temp(high)}, 0.9) \} \), as follows:
(i) As \( N(m(\text{around}, 31)|m(\text{interval}, 25, 31)) = 0 \), it is not possible to derive an argument for \( \text{temp(around}, 31) \) from the set of certain clauses of program \( \mathcal{P}_{\text{eng}} \). Then,
\[ \langle \{ (\text{temp(around}, 31), 0.85) \}, \text{temp(around}, 31), 0.85 \rangle \]
can be derived from rule (17) via INTF.

(ii) Consider one new fuzzy constant “specific\_around\_31” interpreted in the context \( I_{U,m} \) as
\[ m(\text{specific\_around}, 31) = \min(m(\text{interval}, 25, 31), \max(1 - 0.85, m(\text{around}, 31))). \]

The canonical argument for (i) is
\[ \langle \{ (\text{temp(around}, 31), 0.85) \}, \text{temp(specific\_around}, 31), 1 \rangle. \]

Now, as it is not possible to derive an argument for \( \text{temp(high)} \) from the set of certain clauses of program \( \mathcal{P}_{\text{eng}} \) and \( N(m(\text{high})|m(\text{specific\_around}, 31)) = 0.8 \), from the canonical argument via SUA we get
\[ \langle \{ (\text{temp(around}, 31), 0.85) \}, \text{temp(high), 0.8} \rangle. \]

(iii) \( \langle \mathcal{A}_2, \sim \text{oil\_ok}, 0.8 \rangle \) from (ii) and the rule (8) via MPA.

(5) Similarly, an argument \( \langle \mathcal{A}_2, \sim \text{engine\_ok}, 0.8 \rangle \) can be derived using the rules (17), (16) and (7) via INTF, SUA, and MPA, with \( \mathcal{A}_2 = \{ (\text{temp(around}, 31), 0.85); (\sim \text{engine\_ok} \leftarrow \text{temp(high)}, 0.95) \} \).

5. Counter-argumentation and defeat in DePGL+

Given a program and a particular context, it can be the case that there exist arguments for contradictory literals. For instance, in the above example, \( \langle \mathcal{A}_1, \text{engine\_ok}, 0.6 \rangle \) and \( \langle \mathcal{A}_2, \sim \text{engine\_ok}, 0.8 \rangle \), and \( \langle \mathcal{A}_1, \text{oil\_ok}, 0.8 \rangle \) and \( \langle \mathcal{A}_2, \sim \text{oil\_ok}, 0.8 \rangle \), and thus, the program \( \mathcal{P}_{\text{eng}} \) considering the context \( I_{U,m} \) 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 [18] but incorporating the treatment of fuzzy constants.

**Definition 10 (Counterargument).** Let \( \mathcal{P} \) be a DePGL+ program, let \( I_{U,m} \) be a context, and let \( \langle \mathcal{A}_1, Q_1, x_1 \rangle \) and \( \langle \mathcal{A}_2, Q_2, x_2 \rangle \) be two arguments wrt \( \mathcal{P} \) in the context \( I_{U,m} \). We will say that \( \langle \mathcal{A}_1, Q_1, x_1 \rangle \) counterargues \( \langle \mathcal{A}_2, Q_2, x_2 \rangle \) if there exists a subargument (called disagreement subargument) \( \langle \mathcal{I}, Q, \beta \rangle \) of \( \langle \mathcal{A}_2, Q_2, x_2 \rangle \) such that either

(i) \( Q_1 \) and \( Q \) are propositional variables and \( Q_1 = \sim Q \).

\footnote{For a given goal \( Q \), we write \( \sim Q \) as an abbreviation to denote “\( \sim q \)” if \( Q \equiv q \) and “\( q \)” if \( Q \equiv \sim q \).}
confusion arises. The following possibilities arise:

Note that our definition of counterargument accounts for the two usual conflict situations in argumentation systems [14,30]: direct attacks (also called rebutters), in which conflicting arguments have opposite conclusions, and indirect attacks (sometimes referred to as undercutters in the literature), in which a given argument is in conflict with some intermediate step or subargument of another argument.

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, when no fuzzy constants are involved, it seems natural to define it on the basis of necessity degrees associated with arguments, following [18]. When fuzzy constants are involved, due to the concept of contradiction we have adopted, the comparison of conflictive arguments becomes more involved.

To simplify, assume we have two arguments

\[ \text{Arg}_1 = \langle X, p(A), x \rangle, \quad \text{Arg}_2 = \langle Y, p(B), \beta \rangle \]

such that \( A \cap B \) is non-normalized,\(^4\) hence \( \text{Arg}_1 \) counterargues \( \text{Arg}_2 \) and viceversa. In order to compare these arguments what we do is to analyze how much each of them supports the negated conclusion of the other. In fact, from \( \text{Arg}_1 \) we can build an argument for \( \sim p(B) \) by applying the SUA inference rule to its corresponding canonical argument \( \langle X, p(A'), 1 \rangle \), where \( A' = \max(1 - x, A) \), which yields the argument

\[ \text{Arg}_1' = \langle X, \sim p(B), \min(x, N(-B|A)) \rangle \]

taking into account that, by definition, \( \sim p(B) = p(\neg B) \) and that \( N(\neg B|\max(1 - x, A)) = \min(x, N(-B|A)) \). Analogously, from \( \text{Arg}_2 \) we can build the following argument for \( \sim p(A) \):

\[ \text{Arg}_2' = \langle Y, \sim p(A), |(\beta, N(-A|B))| \rangle. \]

Therefore, we need to actually compare the strengths of \( \text{Arg}_1' \) and \( \text{Arg}_2' \) on the one hand, and of \( \text{Arg}_2' \) and \( \text{Arg}_1' \) on the other hand. The following possibilities arise:

1. if \( \min(x, N(-B|A)) > \beta \), then it follows that \( x > \min(\beta, N(-A|B)) \) as well. In this case \( \text{Arg}_1' \) is stronger than \( \text{Arg}_2' \) and \( \text{Arg}_1' \) stronger than \( \text{Arg}_2' \). Then it is clear that \( \text{Arg}_1' \) is strictly stronger than \( \text{Arg}_2' \). Conversely, if \( \min(\beta, N(-A|B)) > x \), then \( \text{Arg}_2' \) is strictly stronger than \( \text{Arg}_1' \);
2. if \( \min(x, N(-B|A)) = \beta \) and \( \min(\beta, N(-A|B)) = \alpha \), then \( \text{Arg}_1' \) and \( \text{Arg}_2' \) are equally strong, as well as \( \text{Arg}_1 \) and \( \text{Arg}_2 \). In this case, we have that \( \min(N(-B|A), N(-A|B)) = \alpha = \beta \), and we can compare the values of \( N(-B|A) \) and \( N(-A|B) \) to decide whether \( \text{Arg}_1 \) or \( \text{Arg}_2 \) finally wins;
3. if \( \min(x, N(-B|A)) = \beta \) and \( \min(\beta, N(-A|B)) < x \), we have that \( \text{Arg}_1' \) is equally strong to \( \text{Arg}_2' \) but \( \text{Arg}_1' \) is weaker than \( \text{Arg}_1 \). In this case we consider \( \text{Arg}_1 \) as the winner. Conversely, when \( \min(x, N(-B|A)) < \beta \) and \( \min(\beta, N(-A|B)) = x \), \( \text{Arg}_2 \) is considered as winner;
4. finally, if \( \min(x, N(-B|A)) < \beta \) and \( \min(\beta, N(-A|B)) < x \) then there is no winner argument, we have a tie.

According to the above considerations we define the following notions of proper and blocking defeaters.

**Definition 11 (Defeat).** Let \( \mathcal{P} \) be a DePGL\(^+\) program, let \( \mathcal{I}_{U,m} \) be a context, and let the argument \( \langle \mathcal{A}_1, Q_1, x_1 \rangle \) counterargue the argument \( \langle \mathcal{A}_2, Q_2, x_2 \rangle \) with disagreement subargument \( \langle \mathcal{A}, Q, \beta \rangle \). We distinguish two cases:

- **Case (1):** \( Q_1 \) and \( Q \) are propositional variables
  - We say that \( \langle \mathcal{A}_1, Q_1, x_1 \rangle \) is a proper (resp. blocking) defeater for \( \langle \mathcal{A}_2, Q_2, x_2 \rangle \) when \( x_1 > \beta \) (resp. \( x_1 = \beta \)).

- **Case (2):** \( Q_1 = p(A) \) and \( Q = p(B) \)
  - We say that \( \langle \mathcal{A}_1, Q_1, x_1 \rangle \) is a proper defeater for \( \langle \mathcal{A}_2, Q_2, x_2 \rangle \) when either

\[^4\] Note that we write \( A \cap B \) for \( m(A) \cap m(B) \), and similarly in other expressions which follow, dropping the \( m \) function symbol when no confusion arises.
following Examples 3 and 9, it is the case that the argument

Example 12. Following Examples 3 and 9, it is the case that the argument \( \langle A_2, \sim \text{ engine\_ok}, 0.8 \rangle \) is a proper
defeater for the argument \( \langle A_1, \text{ engine\_ok}, 0.6 \rangle \) while \( \langle A_2, \sim \text{ oil\_ok}, 0.8 \rangle \) is a blocking defeater for
\( \langle A_1, \text{ oil\_ok}, 0.8 \rangle \).

Example 13. Consider the DePGL+ program

\( \mathcal{P} = \{(\text{temp}(\text{around\_31}), 0.45), (\text{temp}(\text{between\_25\_30}), 0.7)\} \),

where \( \text{temp}(\cdot) \) is a unary predicate of type \( \text{degrees} \), and the context \( \mathcal{A}_{U,m} \) with \( U = \{U_{\text{degrees}} = [-100, 100]^\circ C\} \) and

\[
\begin{align*}
m(\text{around\_31}) &= [26, 31, 36], \\
m(\text{between\_25\_30}) &= [20, 25, 30, 35], \\
m(\sim \text{around\_31}) &= 1 - m(\text{around\_31}), \text{ and} \\
m(\sim \text{between\_25\_30}) &= 1 - m(\text{between\_25\_30}).
\end{align*}
\]

Consider the following sets of clauses:

\( \mathcal{A}_1 = \{(\text{temp}(\text{around\_31}), 0.45)\} \),
\( \mathcal{A}_2 = \{(\text{temp}(\text{between\_25\_30}), 0.7)\} \).

Within the context \( \mathcal{A}_{U,m} \), the arguments

\( A_1 = \langle \mathcal{A}_1, \text{temp}(\text{around\_31}), 0.45 \rangle \),
\( A_2 = \langle \mathcal{A}_2, \text{temp}(\text{between\_25\_30}), 0.7 \rangle \),

can be derived from \( \mathcal{P} \), but notice that \( m(\text{around\_31}) \cap m(\text{between\_25\_30}) \) is a non-normalized fuzzy set, and
thus, \( A_1 \) counterargues \( A_2 \), and viceversa. However, since we have

\[
\begin{align*}
N(m(\sim \text{around\_31})|m(\text{between\_25\_30})) &= 0 \quad \text{and} \\
N(\sim m(\text{between\_25\_30})|m(\sim \text{around\_31})) &= 0,
\end{align*}
\]

one can only derive arguments for the negated literals \( \sim \text{temp}(\text{around\_31}) \) and \( \sim \text{temp}(\text{between\_25\_30}) \) with

\[
\begin{align*}
m(\sim A)(t) &= \begin{cases} 
1, & \text{if } m(A)(t) = 0, \\
0, & \text{otherwise}
\end{cases} \\
N(m(\sim(\text{around\_31})|m(\text{between\_25\_30})) &= 0.2 \quad \text{and} \\
N(m(\text{between\_25\_30})|m(\sim\text{around\_31})) &= 0.2.
\end{align*}
\]
However, as $0.2 < 0.45$ and $0.2 < 0.7$, in this new particular context we would still have that $A_1$ is blocking defeater for $A_2$ and vice versa.

6. Computing warranted arguments in DePGL$^+$

As already explained in Section 2, 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 DePGL$^+$ program).

Skeptical argument-based semantics [20,30] are commonly used for computing warranted arguments. The intuition behind such skeptical approaches to the notion of warrant (using the object language of DePGL$^+$) can be defined as follows:

1. An argument $(A, Q, x)$ is warranted if $(A, Q, x)$ has no defeaters;
2. An argument $(A, Q, x)$ is warranted if it has defeaters $(A_1, Q_1, \beta_1), \ldots, (A_k, Q_k, \beta_k)$, such that every defeater $(A_i, Q_i, \beta_i)$ ($1 \leq i \leq k$) is in turn defeated by a warranted argument.

In DeLP and in P-DeLP the above intuition is formalized in terms of an exhaustive dialectical analysis of all possible argumentation lines rooted in a given argument (see [18] for details) which can be efficiently performed by means of a top-down algorithm, as described in [16].

**Example 14.** Given the following simple P-DeLP program $P = \{(p, 0.45), (p, 0.7)\}$, we can see that $A = \{\{\langle p, 0.7 \rangle\}, p, 0.7\}$ is warranted, as there is no argument defeating $A$ from the program $P$. Similarly, we can conclude that the argument $A' = \{\{\langle p, 0.45 \rangle\}, p, 0.45\}$ is not warranted, as argument $A$ is a proper defeater for the argument $A'$. Argument $A'$ is therefore not warranted as it is defeated by a warranted argument.

In DePGL$^+$, one can perform a similar dialectical analysis provided some care is taken with the management of fuzzy constants and their associated fuzzy unification mechanism as we show in the following example.

**Example 15.** Consider the DePGL$^+$ program $P$ and the context $I_{U,m}$ of Example 13. Let $A_3 = \{\langle temp(about_{25}), 0.9 \rangle\}$, and let $P' = P \cup A_3$ be a new program. Further, consider two new fuzzy constants “between_{31,32}” and “about_{25,ext}”. The three new fuzzy constants are interpreted in the context $I_{U,m}$ as

- $m(about_{25}) = [24, 25, 26]$, 
- $m(about_{25}) = 1 - m(about_{25})$, 
- $m(between_{31,32}) = [26, 31, 32, 37]$, and 
- $m(between_{25,ext}) = [24, 25, 25, 32]$.

Notice that arguments $A_1$ and $A_2$ from Example 13 are still arguments with respect to the new program $P'$. Now, in the frame of the program $P_1$, from the canonical argument associated with $A_1$ and by applying the SUA procedural rule, we can build the argument

$A_3 = \langle A_1, temp(between_{31,32}), 0.45 \rangle$,

since $N(m(between_{31,32}) | max(1 - 0.45, m(around_{31}))) = min(0.45, 1) = 1$. One can easily check that, as in the case of $A_1$, $A_3$ is a blocking defeater for $A_2$, and vice versa. Moreover, as $m(around_{31}) \leq m(between_{31,32})$, i.e. “around_{31}” is more specific than “between_{31,32}”, we have

$N(m(\sim between_{25,30}) | m(around_{31})) \geq N(m(\sim between_{25,30}) | m(between_{31,32}))$

and thus, the argument $A_3$ can be considered a spurious blocking defeater for the argument $A_2$. 
On the other hand, the argument
\[ A_4 = (\mathcal{A}_3, \text{temp(about}_{25}), 0.9), \]
can be derived from \( P \). Then, as \( m(\text{about}_{25}) \cap m(\text{around}_{31}) \) is a non-normalized fuzzy set, the argument \( A_4 \) counterargues the argument \( A_1 \), and vice versa. Moreover, as
\[ N(m(\text{around}_{31})|m(\text{about}_{25})) = N(m(\text{about}_{25})|m(\text{around}_{31})) = 1 \]
and \( 0.9 > 0.45 \), the argument \( A_4 \) is a proper defeater for the argument \( A_1 \). Now, from the canonical argument attached with \( A_4 \) and by applying the SUA procedural rule, we can build the argument
\[ A_5 = (\mathcal{A}_3, \text{temp(about}_{25}, \text{ext}), 0.9), \]
since \( N(m(\text{about}_{25}, \text{ext})|m(\text{about}_{25})) = \min(0.9, 1) = 0.9 \). As \( m(\text{about}_{25}, \text{ext}) \cap m(\text{around}_{31}) \) is a non-normalized fuzzy set, the argument \( A_5 \) counterargues the argument \( A_1 \), and vice versa. Moreover, as it holds that \( N(m(\text{around}_{31})|m(\text{about}_{25}, \text{ext})) = 0.5, N(m(\text{about}_{25}, \text{ext})|m(\text{around}_{31})) = 0 \) and \( \min(0.9, 0.5) > 0.45 \), the argument \( A_5 \) is a proper defeater for the argument \( A_1 \). However, as the fuzzy constant “\text{about}_{25}” is more specific than the fuzzy constant “\text{about}_{25, ext}”, the argument \( A_5 \) can be considered a spurious proper defeater for the argument \( A_1 \).

Considering suitable extensions (by adding ambiguity) of fuzzy constants one can find multiple spurious (proper and blocking) defeaters for arguments. Then, in order to provide DePGGL ± with an efficient procedure for computing warrants (based on an exhaustive dialectical analysis of all argumentation lines), we have to restrict ourselves to canonical defeaters. The formalization of the notion of argumentation line in the framework of DePGGL ± is done as follows. An argumentation line starting in an argument \( \langle \mathcal{A}_0, Q_0, x_0 \rangle \) is a sequence of arguments
\[ \lambda = \langle \mathcal{A}_0, Q_0, x_0 \rangle, \langle \mathcal{A}_1, Q_1, x_1 \rangle, \ldots, \langle \mathcal{A}_n, Q_n, x_n \rangle, \ldots \]
where each \( \langle \mathcal{A}_i, Q_i, x_i \rangle \) is a defeater for the previous argument \( \langle \mathcal{A}_{i-1}, Q_{i-1}, x_{i-1} \rangle \) in the sequence, \( i > 0 \).

In order to avoid fallacious reasoning, most argument-based approaches impose additional constraints on such an argument exchange to be rationally acceptable (see e.g. [24,11]). In particular, for DeGLP+ we impose the following constraints on the argumentation lines:

1. Non-contradiction: given an argumentation line \( \lambda \), the set of arguments of the proponent (resp. opponent) should be non-contradictory wrt \( P \) and \( \mathcal{I}_{\forall, \exists} \).
2. Progressive argumentation: (i) every blocking defeater \( \langle \mathcal{A}_i, Q_i, x_i \rangle \) in \( \lambda \) with \( i > 0 \) is defeated by a proper defeater \( \langle \mathcal{A}_{i+1}, Q_{i+1}, x_{i+1} \rangle \) in \( \lambda \); and (ii) each argument \( \langle \mathcal{A}_i, Q_i, x_i \rangle \) in \( \lambda \), with \( i > 2 \), is such that \( Q_i \neq Q_{i-1} \).
3. Canonicity: every argument \( \langle \mathcal{A}_i, Q_i, x_i \rangle \) in \( \lambda \) with \( i > 0 \) is canonical; i.e. \( \langle \mathcal{A}_i, Q_i, x_i \rangle \) is the best proper or blocking defeater one can consider from a given set of clauses.

An argumentation line satisfying these three conditions are called acceptable. The first 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
\[ \lambda = \langle \ldots, \mathcal{A}_i, Q_i, x_i \rangle, \langle \mathcal{A}_{i+1}, Q_{i+1} \sim Q, x_{i+1} \rangle, \langle \mathcal{A}_{i+2}, Q, x_{i+2} \rangle, \ldots \],

it would mean that \( \langle \mathcal{A}_i, Q, x_i \rangle \) could have been in fact replaced by a stronger argument taking into the information in \( \langle \mathcal{A}_{i+2}, Q, x_{i+2} \rangle \). The canonicity condition avoids the use of spurious defeaters, due to the application of the SUA inference rule, with weaker information than what it actually could carry, and thus able to be potentially defeated by stronger counter-arguments. The enforced use of canonical arguments in the process of exchange of arguments ensures that both the proponent and the opponent are arguing with the best

---

5 It must be noted that the last argument in an argumentation line is allowed to be a blocking defeater for the previous one.
arguments for a given goal at hand. Moreover, it also enforces that both the length of acceptable argumentation lines and the number of acceptable argumentation lines rooted in a given argument is finite, since for any subset of uncertain clauses $\mathcal{A} \subseteq \Delta$ and each predicate $p$ appearing in a given program (there are finitely-many such predicates), there can be at most one canonical argument of the kind $(\mathcal{A}, p(C), 1)$.

Given a program $\mathcal{P}$, a context $\mathcal{I}_{U,m}$ and an argument $(\mathcal{A}_0, Q_0, z_0)$, the set of all acceptable argumentation lines starting in $(\mathcal{A}_0, Q_0, z_0)$ accounts for a whole dialectical analysis for $(\mathcal{A}_0, Q_0, z_0)$.

**Definition 16** (Warrant). Given a program $\mathcal{P} = (\Pi, \Delta)$, a context $\mathcal{I}_{U,m}$, and a goal $Q$, we will say that $Q$ is warranted wrt $\mathcal{P}$ in the context $\mathcal{I}_{U,m}$ with a maximum necessity degree $z$ iff there exists an argument of the form $(\mathcal{A}, Q, z)$, for some $\mathcal{A} \subseteq \Delta$, such that:

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

Note that 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 $(\mathcal{A}, Q, z)$ as specified in Definition 16, we will also say that the argument $(\mathcal{A}, Q, z)$ is warranted. Continuing with Examples 13 and 15, we will next show how to determine, according to the above definition, whether some arguments appearing there (arguments $A_4, A_1$ and $A_2$) are warranted.

**Example 17.** Consider the DePGL$^+$ program $\mathcal{P}$ and the context $\mathcal{I}_{U,m}$ of Example 15. Further, consider two new fuzzy constants “between$_{25,30_{0.7}}$” and “about$_{25_{0.9}}$” interpreted in the context $\mathcal{I}_{U,m}$ as

$$m(\text{between}_{25,30_{0.7}}) = \max(1 - 0.7, m(\text{between}_{25,30})),$$

and

$$m(\text{about}_{25_{0.9}}) = \max(1 - 0.9, m(\text{about}_{25})).$$

Let us recall the following arguments:

$$A_1 = (\mathcal{A}_1, \text{temp(around}_{31}, 0.45),$$
$$A_2 = (\mathcal{A}_2, \text{temp(} \text{between}_{25,30}, 0.7),$$
and

$$A_4 = (\mathcal{A}_3, \text{temp(} \text{about}_{25}, 0.9).$$

Consider first the argument $A_4$. On the one hand, it has neither a proper defeater nor a blocking defeater, hence there exists an acceptable argumentation line starting with $A_4$ with just one argument. Indeed, the only possible argumentation line rooted in $A_4$ that can be obtained is $[A_4]$. Since this line has odd length, according to Definition 16, the goal “temp(about$_{25}$)” can be warranted wrt $\mathcal{P}$ in the context $\mathcal{I}_{U,m}$ with a maximum necessity degree of 0.9. On the other hand, the canonical argument attached with $A_4$ is

$$A_6 = (\mathcal{A}_3, \text{temp(} \text{about}_{25_{0.9}}, 1)$$

and, obviously, $A_6$ is also warranted wrt $\mathcal{P}$ in the context $\mathcal{I}_{U,m}$.

Consider now the case of argument $A_1$. On the one hand, the argument $A_6$ is a canonical proper defeater for $A_4$ and $A_6$ is a warranted argument. On the other hand, the canonical argument attached with $A_2$ is

$$A_7 = (\mathcal{A}_2, \text{temp(} \text{between}_{25,30_{0.7}}, 1)$$

and $A_7$ is a canonical blocking defeater for $A_1$. Therefore two acceptable argumentation lines rooted at $A_1$ can be built: $[A_1, A_6]$ and $[A_1, A_7]$. Since it is not the case that every argumentation line rooted in $A_1$ has odd length, the argument $A_1$ cannot be warranted.

Finally, following a similar discussion for $A_2$, we can conclude that the argument $A_2$ is not warranted either.

However, the goal $\text{temp(} \text{between}_{25,30}$ can be warranted from $\mathcal{A}_3$ with the maximum necessity degree of 0.9 as follows: From the canonical argument $A_6$, by applying the SUA procedural rule, we get the argument

$$A_8 = (\mathcal{A}_3, \text{temp(} \text{between}_{25,30}, 0.9)$$
since $N(m(\text{between}_{25,30})|m(\text{about}_{25,0.9})) = 0.9$, and obviously, $A_8$ is also warranted wrt $\mathcal{P}$ in the context $\mathcal{F}_{U,m}$.

It must be noted that to decide whether a given goal $Q$ is warranted (on the basis of a given argument $A_0$ for $Q$) it may be not necessary to compute every possible argumentation line rooted in $A_0$, e.g. in the case of $A_1$ in the previous example, it sufficed to detect just one even-length argumentation line to determine that is not warranted. Some aspects concerning computing warrant efficiently by means of a top-down procedure in P-DeLP can be found in [16].

7. Related work

To the best of our knowledge, in the literature there have been not many approaches that aim at combining argumentation and fuzziness, except for the work of Schroeder and Schweimeier [32,31,33]. Their argumentation framework is defined for a logic programming framework based on extended logic programming with well-founded semantics, and providing a declarative bottom-up fixpoint semantics along with an equivalent top-down proof procedure. In contrast with our approach, this argumentation framework defines fuzzy unification on the basis of the notion of edit distance, based on string comparison [33]. Their proposal, on the other hand, does not include an explicit treatment of possibilistic uncertainty as in our case.

There have been different approaches connecting argumentative inference, defeasible reasoning and possibilistic logic (e.g. [10,8,9]). Including possibilistic logic as part of an argumentation framework for modelling preference handling and information merging has recently been considered by Amgoud and Kaci [7] and Amgoud and Cayrol [6]. Such formulations are based on using a possibilistic logic framework to handle merging of prioritized information, obtaining an aggregated knowledge base. Arguments are then analyzed on the basis of the resulting aggregated knowledge base. An important difference of these proposals with our formulation is that our framework smoothly integrates an explicit representation of fuzziness together with a possibilistic uncertainty handling. Indeed, in the proposed framework we attach necessity degrees to object level formulas, which are propagated according to suitable inference rules and play an important role in determining the final status of arguments.

Besides of considering possibilistic logic and fuzziness, a number of hybrid approaches connecting argumentation and uncertainty have been developed, such as Probabilistic Argumentation Systems [22,23], which use probabilities to compute degrees of support and plausibility of goals, related to Dempster–Shafer belief and plausibility functions. However this approach is not based on a dialectical theory (with arguments, defeaters, etc.) nor includes fuzziness as presented in this paper. In a recent paper [25] a declarative language to handle arguments with modalities like possible, probable, plausible, etc. is proposed. The resulting framework is applied to modelling problems in the context of a medical domain. In contrast with our approach, no possibilistic logic semantics is associated with the framework, as modalities are categorized in terms of a declarative semantics formalized on the basis of a complete lattice. Besides, no representation of fuzziness at object level is provided in this framework, as in the case of our proposal.

8. Conclusions and future work

In this paper we have provided a formalization of DePGL +, a possibilistic defeasible logic programming language that integrates argumentation capabilities and the characterization of fuzziness at object level in terms of fuzzy constants and fuzzy propositional variables. Our extended framework is motivated on previous research which showed how to successfully integrate defeasible argumentation and possibilistic uncertainty [18]. We have shown how PGL + can be suitably adapted to be included in an argument-based setting. Fuzzy constants in PGL + allow expressing imprecise information about the possibly unknown value of a variable (in the sense of magnitude) modelled as a (unary) predicate. It must be remarked that the notions of argument, defeat and dialectical analysis – common to all argumentation frameworks – could be naturally borrowed into our formalization, and their expressivity was augmented by the incorporation of fuzziness, integrated in the argument-based inference process (rules INTF, MPA, SUA and INA). However, as discussed in Section 5, the notion of canonicity of an argument was an additional requirement in the new, extended framework,
needed to ensure the proper computation of argumentation lines (as discussed in Section 5) by enforcing that the number of argumentation lines rooted in any argument be finite.

Part of our current work is focused on studying complexity issues in the context of our proposal, as well as emerging logical properties which could help to speed up computation of warranted arguments. In that respect, we think that many of the results already available for $\text{PGL}^+$ can be used as a basis for exploring such possibilities in the context of $\text{DePGL}^+$. It must be also noted that we have not introduced default negation in $\text{DePGL}^+$, even though this form of negation is available in $\text{DeLP}$ [21] (where an extended literal $\text{not} \ p$ is proven iff the literal $\ p$ fails to be ultimately acceptable). We are currently exploring the inclusion of default negation into our formalism. On the other hand, Caminada and Amgoud [13] identify anomalies in several argumentation formalisms and provide an interesting solution in terms of rationality postulates which – the authors claim – should hold in any well-defined argumentative system. In [2] we have started a preliminary analysis for this problem in the context of P-DeLP [18], 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 a program should be. This new approach, where warranted and blocked goals are attached with degrees in a similar way of [28], addresses all rationality postulates proposed in [13] without the need of extending the original program with the transposed versions of all strict rules.

As for the knowledge representation capabilities of $\text{DePGL}^+$, the formalism proposed has some representational limitations due to the restriction of allowing only unary predicates. Clearly, having an underlying full predicate logic would make the framework more powerful. Indeed, $\text{PGL}^+\forall$, the first order extension of $\text{PGL}^+$, has been already developed in [1], so it remains as an interesting future work to extend the argumentative framework over $\text{PGL}^+\forall$. Given that this would considerably increase the technical complexity of the paper without providing new conceptual insights, we also leave it as a future task to develop.

Acknowledgements

We thank anonymous reviewers for their comments and suggestions to improve the final version of this paper. This research was partially supported by CICYT Projects MULONG (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).

References


