Computation of skeptical outputs in P-DeLP satisfying indirect consistency: a level-based approach

Teresa Alsinet
Dept. of Computer Science
University of Lleida
Lleida, Spain
tracy@diei.udl.cat

Carlos I. Chesñevar
Dept. of Computer Science
Universidad Nacional del Sur
Bahía Blanca, Argentina
cic@cs.uns.edu.ar

Lluís Godo
Artificial Intelligence
Research Institute, CSIC
Bellaterra, Spain
godo@iiia.csic.es

Abstract

Recent research has identified the notion of *indirect consistency* as a rationality postulate that every rule-based argumentation frameworks should satisfy. Possibilistic Defeasible Logic Programming (P-DeLP) is an argumentation framework based on logic programming which incorporates a treatment of possibilistic uncertainty at object-language level, in which indirect consistency does not hold. In this paper we consider a novel approach to computing warranted arguments in P-DeLP which ensures the above rationality postulate and we describe a procedure to effectively compute them.

Keywords: argumentation, uncertainty handling, possibilistic logic

1 Introduction and motivation

Over the last few years, argumentation has been gaining increasing importance in several AI-related areas, mainly as a vehicle for facilitating rationally justifiable decision making when handling incomplete and potentially inconsistent information. Recently Caminada & Amgoud have defined several *rationality postulates* [6] which every rule-based argumentation system should satisfy. One of such postulates (called *Indirect Consistency*) involves ensuring that the closure of warranted conclusions be guaranteed to be consistent. Failing to satisfy this postulate implies some anomalies and unintuitive results (e.g. the modus ponens rule cannot be applied based on justified conclusions). A number of rule-based argumentation systems are identified in which such postulate does not hold (including DeLP [10] and Prakken & Sartor's [12], among others). As an alternative to solve this problem, the use of *transposed rules* is proposed to extend the representation of strict rules. For grounded semantics, the use of a transposition operator ensures that all rationality postulates are satisfied.

Possibilistic Defeasible Logic Programming (P-DeLP) [2] is an argumentation framework based on logic programming which incorporates the treatment of possibilistic uncertainty at the object-language level. Indeed, P-DeLP is an extension of Defeasible Logic Programming (DeLP) [10], a logic programming approach to argumentation which has been successfully used to solve real-world problems in several contexts such as knowledge distribution [4] and recommendations systems [9], among others. As in the case of DeLP, the P-DeLP semantics is skeptical, based on a query-driven proof procedure which computes warranted (justified) arguments. Following the terminology used in [6], P-DeLP can be seen as a member of the family of *rule-based argumentation systems*, as it is based on a logical language defined over a set of (weigthed) literals and the notions of *strict* and *defeasible* rules, which are used to characterize a P-DeLP program.

L. Magdalena, M. Ojeda-Aciego, J.L. Verdegay (eds): Proceedings of IPMU'08, pp. 497–504
Torremolinos (Málaga), June 22–27, 2008

In [1] the authors have presented a novel level-based approach to computing warranted arguments in P-DeLP which ensures the above rationality postulate without requiring the use of transposed rules. In this paper, after summarizing in Sections 2, 3 and 4 the main elements of P-DeLP, the role of Caminada and Amgoud's rationality postulate of indirect consistency and the new approach introduced in [1] respectively, we further build on this new approach in Section 5 by identifying situations in which a given program may yield multiple outputs, and considering for such a case a skeptical output which is the intersection of the possible outputs. We also provide an effective procedure to compute the skeptical set of warranted arguments for a given P-DeLP program.

2 Argumentation in P-DeLP: an overview

In order to make this paper self-contained, we will present next the main definitions that characterize the P-DeLP framework. For details the reader is referred to [2]. The language of P-DeLP is inherited from the language of logic programming, including the usual notions of atom, literal, rule and fact, but over an extended set of atoms where a new atom "$\sim p$" is added for each original atom p. Therefore, a *literal* in P-DeLP is either an atom p or a (negated) atom of the form $\sim p$, and a *goal* is any literal.

A *weighted* clause is a pair of the form (φ, α), where φ is a rule $Q \leftarrow P_1 \wedge \ldots \wedge P_k$ or a fact Q (i.e., a rule with empty antecedent), where $Q, P_1, \ldots, P_k$ are literals, and $\alpha \in [0, 1]$ expresses a lower bound for the necessity degree of φ. We distinguish between *certain* and *uncertain* clauses. A clause (φ, α) is referred as certain if $\alpha = 1$ and uncertain, otherwise. A set of P-DeLP clauses Γ will be deemed as *contradictory*, denoted $\Gamma \vdash \bot$, if , for some atom q, $\Gamma \vdash (q, \alpha)$ and $\Gamma \vdash (\sim q, \beta)$, with $\alpha > 0$ and $\beta > 0$, where $\vdash$ stands for deduction by means of the following particular instance of the *generalized modus ponens rule*:

$$\frac{(Q \leftarrow P_1 \wedge \cdots \wedge P_k, \alpha) \quad (P_1, \beta_1), \ldots, (P_k, \beta_k)}{(Q, \min(\alpha, \beta_1, \ldots, \beta_k))} \; [GMP]$$

A P-DeLP *program* $\mathcal{P}$ (or just program $\mathcal{P}$) is a pair (Π, Δ), where Π is a non-contradictory finite set of certain clauses, and Δ is a finite set of uncertain clauses. Formally, given a program $\mathcal{P} = (\Pi, \Delta)$, we say that a set $\mathcal{A} \subseteq \Delta$ of uncertain clauses is an *argument* for a goal Q with necessity degree $\alpha > 0$, denoted $\langle \mathcal{A}, Q, \alpha \rangle$, iff:

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

Moreover, if $\langle \mathcal{A}, Q, \alpha \rangle$ and $\langle \mathcal{S}, R, \beta \rangle$ are two arguments wrt a program $\mathcal{P} = (\Pi, \Delta)$, we say that $\langle \mathcal{S}, R, \beta \rangle$ is a *subargument* of $\langle \mathcal{A}, Q, \alpha \rangle$, denoted $\langle \mathcal{S}, R, \beta \rangle \sqsubseteq \langle \mathcal{A}, Q, \alpha \rangle$, whenever $\mathcal{S} \subseteq \mathcal{A}$. From the definition of argument, it follows that if $\langle \mathcal{S}, R, \beta \rangle \sqsubseteq \langle \mathcal{A}, Q, \alpha \rangle$ then (i) $\beta \geq \alpha$, and (ii) if $\beta = \alpha$, then $\mathcal{S} = \mathcal{A}$ iff $R = Q$.

Let $\mathcal{P}$ be a P-DeLP program, and let $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ and $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$ be two arguments wrt $\mathcal{P}$. We say that $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ counterargues $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$[1] iff there exists a subargument (called *disagreement subargument*) $\langle \mathcal{S}, Q, \beta \rangle$ of $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$ such that $Q_1 = \sim Q$. In such a case, we say that $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ is a *proper* (resp. *blocking*) *defeater* for $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$ when $\alpha_1 > \beta$ (resp. $\alpha_1 = \beta$).

In P-DeLP, as in other argumentation systems [8, 13], argument-based inference involves a dialectical process in which arguments are compared in order to determine which beliefs or goals are ultimately accepted (or *justified* or *warranted*) on the basis of a given program. This is formalized in terms of an exhaustive dialectical analysis of all possible argumentation lines rooted in a given argument. An *argumentation line*

[1] In what follows, for a given goal Q, we will write $\sim Q$ as an abbreviation to denote "$\sim q$", if $Q \equiv q$, and "q", if $Q \equiv \sim q$.

starting in an argument $\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle$ is a sequence of arguments $\lambda = [\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle, \langle \mathcal{A}_1, Q_1, \alpha_1 \rangle, \ldots, \langle \mathcal{A}_n, Q_n, \alpha_n \rangle, \ldots]$ such that each $\langle \mathcal{A}_i, Q_i, \alpha_i \rangle$ is a defeater for the previous argument $\langle \mathcal{A}_{i-1}, Q_{i-1}, \alpha_{i-1} \rangle$ in the sequence, $i > 0$. In order to avoid *fallacious* reasoning additional constraints are imposed, namely:

1. **Non-contradiction:** given an argumentation line λ, the set of arguments of the proponent (respectively opponent) should be *non-contradictory* wrt $\mathcal{P}$. [2]
2. **Progressive argumentation:** (i) every blocking defeater $\langle \mathcal{A}_i, Q_i, \alpha_i \rangle$ in λ with $i > 0$ is defeated by a proper defeater[3] $\langle \mathcal{A}_{i+1}, Q_{i+1}, \alpha_{i+1} \rangle$ in λ; and (ii) each argument $\langle \mathcal{A}_i, Q_i, \alpha_i \rangle$ in λ, with $i \geq 2$, is such that $Q_i \neq \sim Q_{i-1}$.

An argumentation line satisfying these constraints is called *acceptable*, and can be proven to be finite. The set of all possible acceptable argumentation lines forms a structure called *dialectical tree*. Given a program $\mathcal{P} = (\Pi, \Delta)$, we say that a goal Q is *warranted* wrt $\mathcal{P}$ with a *maximum necessity degree* α, written $\mathcal{P} \mid\sim^w \langle \mathcal{A}, Q, \alpha \rangle$, whenever there exists an argument $\langle \mathcal{A}, Q, \alpha \rangle$ such that: (i) every acceptable argumentation line starting with $\langle \mathcal{A}, Q, \alpha \rangle$ has an odd number of arguments; and (ii) there is no other argument of the form $\langle \mathcal{A}_1, Q, \beta \rangle$, with $\beta > \alpha$, satisfying (i).

3 Indirect consistency and transposition of strict rules

In a recent paper Caminada and Amgoud [6] have characterized three *rationality postulates* that, according to the authors, any rule-based argumentation system should satisfy in order to avoid anomalies and unintuitive results. Their formalization is intentionally generic, based on a *defeasible theory* $\mathcal{T} = \langle \mathcal{S}, \mathcal{D} \rangle$, where $\mathcal{S}$ is a set of strict rules and $\mathcal{D}$ is a set of defeasible rules. The notion of negation is modelled in the standard way by means of a function "$-$". An *argumentation system* is a pair $\langle Args, Def \rangle$, where $Args$ is a set of arguments (based on a defeasible theory) and $Def \subseteq Args \times Args$ is a defeat relation. The *closure* of a set of literals $\mathcal{L}$ under the set $\mathcal{S}$, denoted $CL_{\mathcal{S}}(\mathcal{L})$ is the smallest set such that $\mathcal{L} \subseteq CL_{\mathcal{S}}(\mathcal{L})$, and if $\phi_1, \ldots, \phi_n \rightarrow \psi \in \mathcal{S}$, and $\phi_1, \ldots, \phi_n \in CL_{\mathcal{S}}(\mathcal{L})$, then $\psi \in CL_{\mathcal{S}}(\mathcal{L})$. A set of literals $\mathcal{L}$ is *consistent* iff there not exist $\psi, \phi \in \mathcal{L}$ such that $\psi = -\phi$, otherwise it is said to be *inconsistent*. An argumentation system $\langle Args, Def \rangle$ can have different *extensions* $E_1, E_2, \ldots, E_n$ $(n \geq 1)$ according to the adopted semantics. The conclusions associated with those arguments belonging to a given extension E_i are defined as $\mathsf{Concs}(E_i)$, and the *output* of the argumentation system is defined skeptically as $\mathsf{Output} = \bigcap_{i=1\ldots n} \mathsf{Concs}(E_i)$.

On the basis of the above concepts, Caminada and Amgoud [6] present three important postulates: *direct consistency*, *indirect consistency* and *closure*. Let $\mathcal{T}$ be a defeasible theory, $\langle Args, Def \rangle$ an argumentation system built from $\mathcal{T}$, Output the set of justified (warranted) conclusions, and $E_1, \ldots, E_n$ its extensions under a given semantics. Then these three postulates are defined as follows:

- $\langle Args, Def \rangle$ satisfies **closure** iff (1) for each i, $\mathsf{Concs}(E_i) = CL_{\mathcal{S}}(\mathsf{Concs}(E_i))$, and (2) $\mathsf{Output} = CL_{\mathcal{S}}(\mathsf{Output})$.
- $\langle Args, Def \rangle$ satisfies **direct consistency** iff (1) for each i, $\mathsf{Concs}(E_i)$ is consistent, and (2) Output is consistent.
- $\langle Args, Def \rangle$ satisfies **indirect consistency** iff (1) for each i, $CL_{\mathcal{S}}(\mathsf{Concs}(E_i))$ is consistent, and (2) $CL_{\mathcal{S}}(\mathsf{Output})$ is consistent.

They show that many rule-based argumentation systems fail to satisfy indirect consistency, in particular DeLP. The same applies for P-DeLP, as illustrated next. Consider the program $\mathcal{P} = (\Pi, \Delta)$, where $\Pi = \{(y, 1), (\sim y \leftarrow a \wedge b, 1)\}$ and $\Delta = \{ (a, 0.9), (b, 0.9) \}$. It is easy to see that arguments $\langle \{(a, 0.9)\}, a, 0.9 \rangle$ and $\langle \{(b, 0.9)\}, b, 0.9 \rangle$ have no defeaters wrt $\mathcal{P}$. Thus $\{y, a, b\} = \mathsf{Output}$ turns out to be warranted, but $y, \sim y \in$

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

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

$CL_\Pi(\{y, a, b\})$, so that indirect consistency does not hold.

Caminada and Amgoud propose as a solution the definition of a special *transposition operator* Cl_{tp} for computing the closure of strict rules. This accounts for taking every strict rule $r = \phi_1, \phi_2, \ldots, \phi_n \rightarrow \psi$ as a material implication in propositional logic which is equivalent to the disjunction $\phi_1 \vee \phi_2 \vee \ldots, \phi_n \vee \neg\psi$. From that disjunction different rules of the form $\phi_1, \ldots, \phi_{i-1}, \neg\psi, \phi_{i+1}, \ldots, \phi_n \rightarrow \neg\phi_i$ can be obtained (*transpositions* of r). If $\mathcal{S}$ is a set of strict rules, Cl_{tp} is the minimal set such that (1) $\mathcal{S} \subseteq Cl_{tp}(\mathcal{S})$ and (2) if $s \in Cl_{tp}(\mathcal{S})$ and t is a transposition of s, then $t \in Cl_{tp}(\mathcal{S})$. The use of such an operator allows the three rationality postulates to be satisfied in the case of the grounded extension [6, pp. 294] (which corresponds to the one associated with systems like DeLP or P-DeLP).

4 A level-based approach to computing warranted arguments

Although Caminada and Amgoud's proposal of using transposed rules is indeed valuable for rule-based argumentation systems to overcome the problem of indirect inconsistency, we have provided in [1] a new formal definition of warranted goal with maximum necessity degree which takes into account direct and indirect conflicts between arguments without explicitly transposing strict rules. This new approach distinguishes between *warranted* and *blocked* goals, which allows for a more refined criterion and has some other advantages (see [1] for a discussion), in particular it will allow us to define in next section an efficient top-down procedure for their computation.

The idea is the following. While direct conflicts between arguments refer to the case of both proper and blocking defeaters, as described in Section 2, indirect conflicts between arguments refer to the case when there exists an inconsistency emerging from the set of certain (strict) clauses of a program and arguments with no defeaters. For instance, consider the program $\mathcal{P} = (\Pi, \Delta)$ with $\Pi = \{(\sim y \leftarrow a \wedge b, 1), (y, 1), (\sim x \leftarrow c \wedge d, 1), (x, 1)\}$ and $\Delta = \{(a, 0.7), (b, 0.7), (c, 0.7), (d, 0.6)\}$. In the original P-DeLP, $\langle\{(a, 0.7)\}, a, 0.7\rangle$ and $\langle\{(b, 0.7)\}, b, 0.7\rangle$ are arguments with no defeaters and therefore their conclusions would be warranted. However, since $\Pi \cup \{(a, 0.7), (b, 0.7)\} \vdash \bot$, arguments $\langle\{(a, 0.7)\}, a, 0.7\rangle$ and $\langle\{(b, 0.7)\}, b, 0.7\rangle$ express (indirect) contradictory information. Moreover, as both goals are supported by arguments with the same necessity degree 0.7, none of them should be neither warranted nor rejected: we will refer to them as (indirect) *blocked* goals. On the other hand, a similar situation appears with $\langle\{(c, 0.7)\}, c, 0.7\rangle$ and $\langle\{(d, 0.6)\}, d, 0.6\rangle$. As before, $\Pi \cup \{(c, 0.7), (d, 0.6)\} \vdash \bot$, but in this case the necessity degree of goal c is greater than the necessity degree of goal d. In such a case, c can be indeed considered as a warranted goal (to the degree 0.7).

According to [1], an *output* for a P-DeLP program $\mathcal{P}$ is a pair $(Warr, Block)$, where $Warr$ and $Block$, denote respectively a set of warranted and blocked goals (together with their degrees) fulfilling a set of conditions, formalized in the next definition, that ensure a proper handling of the problem of global inconsistency. The intended construction of the sets $Warr, Block$ is done level-wise, starting from the highest level and iteratively going down from one level to next level below. If $1 \geq \alpha_1 > \ldots > \alpha_p > 0$ are the weights appearing in the set of arguments $ARG(\mathcal{P}) = \{\langle\mathcal{A}, Q, \alpha\rangle \mid \mathcal{A}$ is an argument for Q with necessity α wrt $\mathcal{P}\}$, one can write $Warr = Warr(\alpha_1) \cup \ldots \cup Warr(\alpha_p)$ and $Block = Block(\alpha_1) \cup \ldots \cup Block(\alpha_p)$, where $Warr(\alpha_i)$ and $Block(\alpha_i)$ are respectively the sets of the warranted and blocked goals to the (maximum) degree α_i. We will also write $Warr(> \alpha_i)$ to denote $\cup_{\beta > \alpha_i} Warr(\beta)$, and analogously for $Block(> \alpha_i)$, assuming $Warr(> \alpha_1) = Block(> \alpha_1) = \emptyset$. In what follows, given a program $\mathcal{P} = (\Pi, \Delta)$ we will denote by $rules(\Pi)$ and $facts(\Pi)$ the set of strict rules and strict facts of $\mathcal{P}$ respectively.

Definition 1 (Output for P-DeLP program) *An output for a program* $\mathcal{P} = (\Pi, \Delta)$*is a*

pair $(Warr, Block)$ *where the sets* $Warr(\alpha_i)$ *and* $Block(\alpha_i)$*, for* $i = 1 \ldots p$ *are required to satisfy the following recursive constraints:*

1. An argument $\langle \mathcal{A}, Q, \alpha_i \rangle \in ARG(\mathcal{P})$ *is called* acceptable *if it satisfies the following three conditions:*

(i) for any $\beta > \alpha_i$*, neither* $(\sim Q, \beta)$ *nor* (Q, β) *are in* $Warr(> \alpha_i) \cup Block(> \alpha_i)$

(ii) if $\langle \mathcal{B}, R, \beta \rangle \sqsubseteq \langle \mathcal{A}, Q, \alpha_i \rangle$ *with* $R \neq Q$*, then* $(R, \beta) \in Warr(\beta)$

(iii) $rules(\Pi) \cup Warr(> \alpha_i) \cup \{(R, \alpha_i) \mid \langle \mathcal{B}, R, \alpha_i \rangle \sqsubseteq \langle \mathcal{A}, Q, \alpha_i \rangle\} \nvdash \bot$.

2. For each acceptable $\langle \mathcal{A}, Q, \alpha_i \rangle \in ARG(\mathcal{P})$*,* $(Q, \alpha_i) \in Block(\alpha_i)$ *whenever*

(i) either there exists an acceptable argument $\langle \mathcal{B}, \sim Q, \alpha_i \rangle \in ARG(\mathcal{P})$*; or*

(ii) there exists $G \subseteq \{(P, \alpha_i) \mid \langle \mathcal{C}, P, \alpha_i \rangle \in ARG(\mathcal{P})$ *is acceptable with* $\sim P \notin Block(\alpha_i)\}$ *such that* $G \cup Warr(> \alpha_i) \cup rules(\Pi) \nvdash \bot$ *but* $G \cup Warr(> \alpha_i) \cup rules(\Pi) \cup \{(Q, \alpha_i)\} \vdash \bot$*;*

otherwise, $(Q, \alpha_i) \in Warr(\alpha_i)$.

The intuition underlying this definition is as follows: an argument $\langle \mathcal{A}, Q, \alpha \rangle$ is either warranted or blocked whenever each subargument $\langle \mathcal{B}, R, \beta \rangle \sqsubseteq \langle \mathcal{A}, Q, \alpha \rangle$, with $Q \neq R$, is warranted; then it is finally warranted if it induces neither direct nor indirect conflicts, otherwise it is blocked.

It is shown in [1] that if $(Warr, Block)$ is an output of a P-DeLP program, the set $Warr$ of warranted goals is indeed non-contradictory and satisfies indirect consistency with respect to the set of strict rules.

Proposition 2 (Indirect consistency) *Let* $\mathcal{P} = (\Pi, \Delta)$ *be a P-DeLP program and let* $(Warr, Block)$ *be an output for* $\mathcal{P}$*. Then:*

(i) $facts(\Pi) \subseteq Warr$

(ii) $Warr \nvdash \bot$*, and*

(iii) if $rules(\Pi) \cup Warr \vdash (Q, \alpha)$ *then* $(Q, \beta) \in Warr$ *for some* $\beta \geq \alpha$

5 Skeptical level-based approach

In this section we will come to the open question formulated in [1] of whether a program $\mathcal{P}$ always has a *unique* output $(Warr, Block)$ according to Def. 1. In general, the answer is yes, although there are some recursive situations that might lead to different outputs. These recursive situations can be produced by both direct and indirect conflicts between arguments. For instance, consider the program $\mathcal{P}_1 = (\Pi_1, \Delta_1)$, with

$$\begin{array}{rcl} \Pi_1 & = & \{(y, 1)\} \text{ and} \\ \Delta_1 & = & \{(p, 0.9), (q, 0.9) \\ & & (\sim p \leftarrow q, 0.9), (\sim q \leftarrow p, 0.9)\}. \end{array}$$

Then, according to Def. 1, p is a warranted goal iff q and $\sim q$ are a pair of blocked goals and viceversa, q is a warranted goal iff p and $\sim p$ are a pair of blocked goals. Hence, in that case we have two possible outputs: $(Warr_1, Block_1)$ and $(Warr_2, Block_2)$ where
$Warr_1 = \{(y, 1), (p, 0.9)\}$,
$Block_1 = \{(q, 0.9), (\sim q, 0.9)\}$,
$Warr_2 = \{(y, 1), (q, 0.9)\}$ and
$Block_2 = \{(p, 0.9), (\sim p, 0.9)\}$.

In such a case, either p or q can be warranted goals (but just one of them) and the indeterminacy is due to a direct conflict between arguments in $\mathcal{P}$ since there exists an argument for $\sim p$ which depends on q and an argument for $\sim q$ which depends on p. A different recursive situation is due to indirect conflicts between arguments. For instance, consider the program $\mathcal{P}_2 = (\Pi_2, \Delta_2)$, with

$$\begin{array}{rcl} \Pi_2 & = & \{(y, 1), \\ & & (\sim y \leftarrow p \wedge r, 1), (\sim y \leftarrow q \wedge s, 1)\} \\ \Delta_2 & = & \{(p, 0.9), (q, 0.9), \\ & & (r \leftarrow q, 0.9), (s \leftarrow p, 0.9)\}. \end{array}$$

According to Def. 1, p is a warranted goal iff q and s are a pair of blocked goals and viceversa, q is a warranted goal iff p and r are a pair of blocked goals. Then, in this case we also have two possible outputs: $(Warr_1, Block_1)$ and $(Warr_2, Block_2)$ where
$Warr_1 = \{(y, 1), (p, 0.9)\}$,
$Block_1 = \{(q, 0.9), (s, 0.9)\}$,
$Warr_2 = \{(y, 1), (q, 0.9)\}$ and
$Block_2 = \{(p, 0.9), (r, 0.9)\}$.

Again, either p or q can be warranted (but just one of them) but now the indeterminacy is due to an indirect conflict between arguments in $\mathcal{P}_2$: the warranty of p depends on r which

in turn depends on q, and the warranty of q depends on s which in turn depends on p.

The above examples show that, although our approach is skeptical, we can get alternative extensions for warranted beliefs whenever some recursive situation, due to direct or indirect conflicts, occurs between the literals of a P-DeLP program. A natural solution for this problem would be adopting the intersection of all possible outputs in order to define the set of those literals which are ultimately warranted.

Definition 3 (Skeptical ouput) *Let $\mathcal{P}$ be a P-DeLP program, and let $output_i(\mathcal{P}) = (Warr_i, Block_i)$ for $i = 1, \ldots, n$ denote all possible outputs for $\mathcal{P}$. Then, the* skeptical output *for $\mathcal{P}$ is defined as $output_{skep}(\mathcal{P}) = (Warr, Block)$ where $Warr = \bigcap_{i=1\ldots n} Warr_i$ and $Block = \bigcap_{i=1\ldots n} Block_i$.*

Given a program $\mathcal{P}$, it is always possible to consider another program $\mathcal{P}'$ whose output is *unique* and corresponds with the skeptical output of $\mathcal{P}$. Indeed, let $W' = \bigcup_{i=1\ldots n} Warr_i$, $B' = \bigcup_{i=1\ldots n} Block_i$, let Δ' be defined as

$$\Delta' = \Delta \backslash \{(\varphi, \alpha) \in \Delta \mid \text{ for some } Q \in (W' \backslash Warr) \bigcup (B' \backslash Block) \text{ either } Q \text{ or } \sim Q \text{ occurs in } \varphi\},$$

and let $\mathcal{P}' = (\Pi, \Delta')$ be called the *deterministic* P-DeLP program for $\mathcal{P}$. Then we have:

Proposition 4 *Let $\mathcal{P}$ be a P-DeLP program, let $(Warr, Block)$ be the skeptical output for $\mathcal{P}$, and let $\mathcal{P}'$ be the deterministic P-DeLP program for $\mathcal{P}$. Then, $(Warr, Block)$ is the unique output for $\mathcal{P}'$ and hence satisfies indirect inconsistency.*

Proof: By construction each acceptable argument in $\mathcal{P}'$ is an acceptable argument in $\mathcal{P}$. But since acceptable arguments in $\mathcal{P}'$ only involve literals in $Warr \cup Block$, $(Warr, Block)$ is an output for $\mathcal{P}'$ as well. Now, by Definition 1, each acceptable argument is either blocked or warranted, hence, by construction of $Warr$ and $Block$, $(Warr, Block)$ is the only output for $\mathcal{P}'$. □

For instance, the skeptical outputs for the above P-DeLP programs $\mathcal{P}_1$ and $\mathcal{P}_2$ are $output_{skep}(\mathcal{P}_1) = output_{skep}(\mathcal{P}_2) = (\{(y, 1)\}, \emptyset)$, and the corresponding deterministic P-DeLP programs are $\mathcal{P}'_1 = (\Pi_1, \Delta'_1)$ and $\mathcal{P}'_2 = (\Pi_2, \Delta'_2)$ with $\Delta'_1 = \Delta'_2 = \emptyset$.

Given a P-DeLP program $\mathcal{P}$ the following algorithm determines whether the output of $\mathcal{P}$ is unique and computes the skeptical output for $\mathcal{P}$ together with the deterministic P-DeLP program whenever some recursive situation leads to different outputs.

Algorithm 5 *Skeptical output*

Input $\mathcal{P} = (\Pi, \Delta)$: *A P-DeLP program*
Output
  unicity: Boolean
  $\mathcal{P}' = (\Pi, \Delta')$: *Deterministic P-DeLP program*
  (W_s, B_s): *Skeptical output for* $\mathcal{P}$
Auxiliary variables
  α: *Necessity degree* $\in [0, 1]$
  C: *Set of arguments*
  A: *Set of goals*
  G: *Set of goals*
  (W_e, B_e): *Union of outputs for level* α
Method
  unicity := *true*
  $\Delta' := \Delta$
  $W_s := \{(Q, 1) \mid \Pi \vdash (Q, 1)\}$
  $B_s := \emptyset$
  $\alpha :=$ `maximum_level_degree`(Δ')
  while $(\alpha \neq 0)$ **do**
    $C :=$ $\{\langle \mathcal{A}, Q, \alpha\rangle \mid \mathcal{A} \subseteq \Delta'$ *is an argument for* Q *with necessity* α *and it is acceptable untill the* α*-level* $\}$
    $A := \{Q \mid \langle \mathcal{A}, Q, \alpha\rangle \in C$ *is acceptable* $\}$
    $(W_s, B_s, W_e, B_e) :=$ `level_warrant`$(\alpha, C, A, W_s, B_s, \emptyset)$
      $\Delta' :=$ $\Delta' \backslash \{(\varphi, \beta) \in \Delta' \mid$ *for some* $P \in ((W_e \backslash W_s) \cup (B_e \backslash B_s))$ *either* P *or* $\sim P$ *occurs in* φ $\}$
    $\alpha :=$ `next_level_degree`(Δ', α)
  end while
  return$(unicity, W_s, B_s, \Delta')$

First the algorithm computes the set of warrants form the set of certain clauses Π. Then, for each level $\alpha < 1$ of uncertain clauses in Δ, it determines the set C of "almost" acceptable arguments to the α-level[4] and the set A of goals with ultimately accepted arguments in C. Finally, the algorithm computes, by means of the recursive function `level_warrant`, the skeptical output (W_s, B_s) for each level α. In case of indeterminacy the function `level_warrant` updates the *unicity* variable and computes the extended set of warranted and blocked arguments (W_e, B_e), i.e. the union of all outputs of level α. Then, from (W_s, B_s) and (W_e, B_e), the algorithm

[4] An argument $\langle \mathcal{A}, Q, \alpha\rangle$ is called "almost" acceptable to the α-level if it satisfies the three conditions of acceptability described in Def. 1 relative to the sets W_s and B_s, that is, whenever: (i) $(Q, \beta) \notin W_s \cup B_s$ and $(\sim Q, \beta) \notin W_s \cup B_s$ for all $\beta > \alpha$, (ii) if $\langle \mathcal{B}, R, \beta\rangle \sqsubseteq \langle \mathcal{A}, Q, \alpha\rangle$ with $\beta > \alpha$ then $(R, \beta) \in W_s$, and (iii) $rules(\Pi) \cup W_s \cup \{(Q, \alpha)\} \nvdash \bot$.

updates the deterministic set of uncertain clauses Δ'. Note that for each level α the skeptical output is computed form the updated set of uncertain clauses Δ', and thus, the indeterminacy at level α is propagated to the rest of levels lower than α.

function `level_warrant`

Input
α: Necessity degree $\in [0, 1]$
C: Set of arguments
A: Set of acceptable goals
W: Set of warranted goals
B: Set of blocked goals
G: Set of goals

Output
(W_s, B_s): Skeptical output
(W_e, B_e): Union of outputs

Auxiliary variables
I, D, S: Set of goals
$C_{Q_1}, \ldots, C_{Q_n}$: Set of arguments
$A_{Q_1}, \ldots, A_{Q_n}$: Set of goals
$W_s^{Q_1}, \ldots, W_s^{Q_n}$: Set of skeptical warranted goals
$B_s^{Q_1}, \ldots, B_s^{Q_n}$: Set of skeptical blocked goals
$W_e^{Q_1}, \ldots, W_e^{Q_n}$: Union of warranted goals
$B_e^{Q_1}, \ldots, B_e^{Q_n}$: Union of blocked goals

Method
while ($A \neq \emptyset$ or $G \neq \emptyset$) **do**
repeat
for each $Q \in A$ such that $\sim Q \in A$ **do**
$B := B \cup \{(Q, \alpha), (\sim Q, \alpha)\}$
$A := A \backslash \{Q, \sim Q\}$
$C := C \backslash \{\langle \mathcal{A}, P, \alpha\rangle \in C \mid \langle \mathcal{B}, R, \alpha\rangle \sqsubseteq \langle \mathcal{A}, P, \alpha\rangle$ with $R = Q$ or $R = \sim Q\}$
end for
repeat
for each $Q \in A$ such that $\langle \mathcal{A}, \sim Q, \alpha\rangle \notin C$ **do**
$G := G \cup \{Q\}$
$A := A \backslash \{Q\}$
$C := C \backslash \{\langle \mathcal{A}, Q, \alpha\rangle\}$
end for
$I := \{Q \in G \mid$ `indirect_block`$(\alpha, Q, G, W, \emptyset)$ $\}$
$B := B \cup \{(Q, \alpha) \mid Q \in I\}$
$G := G \backslash I$
$C := C \backslash \{\langle \mathcal{A}, P, \alpha\rangle \in C \mid \langle \mathcal{B}, R, \alpha\rangle \sqsubseteq \langle \mathcal{A}, P, \alpha\rangle$ with $R \in I$ or $\sim R \in I\}$
until C does not vary
for each $Q \in G$ **do**
$D :=$ `compute_dependencies`(α, Q, C, A)
if ($\neg$ `indirect_block`(α, Q, G, W, D)) **then**
$W := W \cup \{(Q, \alpha)\}$
$G := G \backslash \{Q\}$
$C := C \backslash \{\langle \mathcal{A}, Q, \alpha\rangle\}$
end if
end for
$A := A \cup \{Q \mid \langle \mathcal{A}, Q, \alpha\rangle \in C$ is acceptable $\}$
until A does not vary
$S :=$ `direct_indeterminacy`(α, A, C)
if ($S \neq \emptyset$) **then**
$unicity :=$ false
for each $Q \in S$ **do**
$B_Q := B \cup \{(Q, \alpha), (\sim Q, \alpha)\}$
$A_Q := A \backslash \{Q\}$
$C_Q := C \backslash \{\langle \mathcal{A}, P, \alpha\rangle \in C \mid \langle \mathcal{B}, R, \alpha\rangle \sqsubseteq \langle \mathcal{A}, P, \alpha\rangle$ with $R = Q$ or $R = \sim Q\}$
$(W_s^Q, B_s^Q, W_e^Q, B_e^Q) :=$ `level_warrant`$(\alpha, C_Q, A_Q, W, B_Q, G)$
end for
$(W_s, B_s) := (\cap_{Q \in S} W_Q, \cap_{Q \in S} B_Q)$
$(W_e, B_e) := (\cup_{Q \in S}(W_s^Q \cup W_e^Q), \cup_{Q \in S}(B_s^Q \cup B_e^Q))$
return(W_s, B_s, W_e, B_e)
else
$S :=$ `indirect_indeterminacy`(α, A, C, W, G)
if ($S \neq \emptyset$) **then**
$unicity :=$ false
for each $Q \in S$ **do**
$W_Q := W \cup \{(Q, \alpha)\}$
$G_Q := G \backslash \{Q\}$
$C_Q := C \backslash \{\langle \mathcal{A}, Q, \alpha\rangle\}$
$A_Q := A \cup \{Q \mid \langle \mathcal{A}, Q, \alpha\rangle \in C_Q$ is acceptable $\}$
$(W_s^Q, B_s^Q, W_e^Q, B_e^Q) :=$ `level_warrant`$(\alpha, C_Q, A_Q, W_Q, B, G_Q)$
end for
$(W_s, B_s) := (\cap_{Q \in S} W_Q, \cap_{Q \in S} B_Q)$
$(W_e, B_e) := (\cup_{Q \in S}(W_s^Q \cup W_e^Q), \cup_{Q \in S}(B_s^Q \cup B_e^Q))$
return(W_s, B_s, W_e, B_e)
end if
end while
$(W_s, B_s) := (W, B)$
$(W_e, B_e) := (W, B)$
return(W_s, B_s, W_e, B_e)

The function `level_warrant` first checks direct conflicts between acceptable goals of A and computes the set G of acceptable goals which do not produce direct conflicts. Then the function `indirect_block` checks possible indirect conflicts between the goals of G and the set of warranted goals W and, for each goal $Q \in G$, Q is warranted iff Q does not produce indirect conflicts between the goals of G, the set of warranted goals W, and the set of goals D which do not depend on Q. For each $Q \in G$, the function `compute_dependencies` computes, from the arguments in C and the set of acceptable goals A, the set of goals D which do not depend on Q. Finally functions `direct_indeterminacy` and `indirect_indeterminacy` check possible recursive situations between the goals of A and G, respectively, and compute the set of goals $S = \{Q_1, \ldots, Q_n\}$ which lead to the indeterminacy. When the indeterminacy occurs between the goals of A, i.e. when $S \subseteq A$, the function `level_warrant` recursively computes the output obtained for each goal $Q_i \in S$ by setting Q_i as a blocked goal. Similarly, when the indeterminacy occurs between the goals of G, the function `level_warrant` computes the output obtained for each goal $Q_i \in S$ by setting Q_i as a *warranted* goal. Finally the skeptical output (W_s, B_s) and the extended output (W_e, B_e) are computed by taking the intersection and union, respectively, of the set of all possible outputs for each goal $Q_i \in S$. When the output is unique the function `level_warrant` processes all goals in A and G and the extended output corresponds with the skeptical output.

function `indirect_block`

Input
α: Necessity degree $\in [0, 1]$
Q: Goal
G: Set of goals
W: Set of warranted goals
D: Set of goals which do not depend on Q

Output $conflict$: Boolean

Method
$conflict :=$ $\exists\ S \subseteq (G \backslash \{Q\}) \cup D$ such that $rules(\Pi) \cup W \cup \{(P, \alpha) \mid P \in S\} \nvdash \bot$ and $rules(\Pi) \cup W \cup \{(P, \alpha) \mid P \in S\} \cup \{(Q, \alpha)\} \vdash \bot$
return$(conflict)$

function `compute_dependencies`

Input
α: Necessity degree $\in [0, 1]$

Q: Goal
C: Set of arguments
A: Set of goals
Output D: Set of goals
Method
$D :=$ $\{P \in A \mid \langle \mathcal{A}, \sim P, \alpha\rangle \in C$ and Q is not a subgoal of $\mathcal{A}\} \cup$
$\{P \mid \langle \mathcal{A}, P, \alpha\rangle \in C$ and Q is not a subgoal of $\mathcal{A}\}$
return(D)

The function `indirect_block` determines whether there exists an indirect conflict between the goal Q and the set of goals $S \subseteq (G \backslash \{Q\}) \cup D$ which do not depend on Q and which could be warranted. And the function `compute_dependencies` computes, from A and C, the set of goals D which do not depend on Q and which could be warranted.

function `direct_indeterminacy`
Input
α: Necessity degree $\in [0, 1]$
A: Set of goals
C: Set of arguments
Output S: Set of goals
Method
if ($\exists$ $S \subseteq A$ such that, for all $Q \in S$,
$\langle \mathcal{A}, \sim Q, \alpha\rangle \in C$ and R is a subgoal of $\mathcal{A}$,
for all $R \in S$ with $R \neq Q$) **then** **return**(S)
else return$(\emptyset)$

The function `direct_indeterminacy` checks whether, for some set of goals $S \subseteq A$, warranting each goal $Q \in S$ depends on warranting $\sim Q$ which in turn depends on warranting the rest of goals in S.

function `indirect_indeterminacy`
Input
α: Necessity degree $\in [0, 1]$
A: Set of goals
C: Set of arguments
W: Set of warranted goals
G: Set of goals
Output S: Set of goals
Method
if ($\exists$ $S \subseteq G$ such that for all $Q \in S$,
$\exists$ $D_Q \subseteq$ `compute_dependences`(α, Q, C, A)
such that `indirect_block`(α, Q, G, W, D_Q)
with D_Q minimal w.r.t. set inclusion and,
for all $P \in D_Q$, the warranty of P depends on
the warranty of $R \in S$ and, for all $R \in S$ with $R \neq Q$,
the warranty of $P \in D_Q$ depends on the warranty of R) **then**
return(S)
else return$(\emptyset)$

The function `indirect_indeterminacy` checks whether for some set of goals $S \subseteq G$ warranting each goal $Q \in S$ depends on warranting the set of goals $D_Q \subseteq$ `compute_dependences`(α, Q, C, A) which in turn depends on warranting the rest of goals in S.

Acknowledgements. This research was partially supported by the Spanish CICYT Projects MULOG2 (TIN2007-68005-C04-01/02) and IEA (TIN2006-15662-C02-01/02), by CONICET (Argentina), and by the Secretaría General de Ciencia y Tecnología de la Universidad Nacional del Sur (Project PGI 24/ZN10).

References

[1] T. Alsinet, C. I. Chesñevar and L. Godo. A Level-based Approach to Computing Warranted Arguments in Possibilistic Defeasible Logic Programming. In *Proc. of COMMA-2008 Conference*, 2008 (in press).

[2] T. Alsinet, C. I. Chesñevar, L. Godo, and G. Simari. A logic programming framework for possibilistic argumentation: Formalization and logical properties. *Fuzzy Sets and Systems*, 159(10):1208–1209, 2008.

[3] P. Besnard and A. Hunter. A logic-based theory of deductive arguments. *Artif. Intell.*, 128(1-2):203–235, 2001.

[4] R. Brena, J. Aguirre, C. Chesñevar, E. Ramírez, and L. Garrido. Knowledge and information distribution leveraged by intelligent agents. *Knowl. Inf. Syst.*, 12(2):203–227, 2007.

[5] M. Caminada and L. Amgoud. An axiomatic account of formal argumentation. In *Proc. of AAAI-2005 Conference*, pp. 608–613.

[6] M. Caminada and L. Amgoud. On the evaluation of argumentation formalisms. *Artif. Intell.*, 171(5-6):286–310, 2007.

[7] C. Cayrol and M. Lagasquie-Schiex. Graduality in argumentation. *J. Artif. Intell. Res. (JAIR)*, 23:245–297, 2005.

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

[9] C. Chesñevar, A. Maguitman, and G. Simari. Argument-Based Critics and Recommenders: A Qualitative Perspective on User Support Systems. *Journal of Data and Knowledge Engineering*, 59(2):293–319, 2006.

[10] A. García and G. Simari. Defeasible Logic Programming: An Argumentative Approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2004.

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

[12] H. Prakken and G. Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-classical Logics*, 7:25–75, 1997.

[13] H. Prakken and G. Vreeswijk. Logical Systems for Defeasible Argumentation. In D. Gabbay and F.Guenther (eds.), *Handbook of Phil. Logic*, pp. 219–318. Kluwer, 2002.