A computational method for defeasible argumentation based on a recursive warrant semantics

Teresa Alsinet[1], Ramón Béjar[1], and Lluís Godo[2]

[1] Department of Computer Science – University of Lleida
C/Jaume II, 69 – 25001 Lleida, SPAIN
{tracy, ramon}@diei.udl.cat
[2] Artificial Intelligence Research Institute (IIIA-CSIC)
Campus UAB - 08193 Bellaterra, Barcelona, SPAIN
godo@iiia.csic.es

Abstract. In a recent paper [2] the authors have formalized a recursive semantics for warranted conclusions in a general defeasible argumentation framework based on a propositional logic. The warrant recursive semantics is based on a general notion of collective (non-binary) conflict among arguments allowing to ensure direct and indirect consistency properties. This general framework was also extended by allowing levels of defeasibility and providing a level-wise recursive definition of warranted and blocked conclusions. In this paper we focus on the particular framework of Defeasible Logic Programming (DeLP) extended with levels of defeasibility for which we characterize programs with a unique output (extension) for warranted conclusions, and we design, for this type of programs, an algorithm for computing warranted conclusions in polynomial space and with an upper bound on complexity equal to P^{NP}.

Keywords: defeasible argumentation, recursive semantics, computational aspects

1 Introduction and motivation

Possibilistic Defeasible Logic Programming (P-DeLP) [1] is a rule-based argumentation framework which incorporates the treatment of possibilistic uncertainty at the object-language level. Indeed, P-DeLP is an extension of Defeasible Logic Programming (DeLP) [17] in which program defeasible rules are attached with necessity degrees (belonging to the real unit interval $[0, 1]$) expressing their belief strength. Warranted (justified) conclusions or beliefs with a maximum necessity degree are formalized in terms of an exhaustive dialectical analysis of all possible arguments.

Because the dialectical tree-based P-DeLP semantics for warranted conclusions does not satisfy the *indirect consistency* property[3], in [3, 2] a new recursive semantics ensuring that property has been investigated. In particular, following the ideas of an approach by Pollock [19], the authors have defined in [2] a recursive semantics for warrant in a general defeasible argumentation framework based on a notion of collective (non-binary) conflict among arguments. This general framework has also been extended by allowing to introduce levels of defeasibility in the knowledge base and providing a

[3] This property is satisfied when the set of warranted conclusions is consistent with respect to the set of strict (non-defeasible) rules of a program and was identified in [6] as a basic postulate that rule-based argumentation systems should satisfy.

level-wise recursive definition of warranted and blocked conclusions. In this setting we distinguish between warranted and blocked conclusions. A warranted conclusion is a ultimately justified conclusion which is based only on warranted information and which does not generate any conflict, while a blocked conclusion is a conclusion which, like warranted conclusions, is based only on warranted information but which generates a conflict. Finally, in the same paper, this recursive warrant semantics is particularized to the framework of P-DeLP, the resulting formalism being called RP-DeLP (Recursive P-DeLP), and circular definitions of conflict between arguments that lead to multiple extensions (outputs) are identified by means of *warrant dependency graphs*.

In this paper, after overviewing in Section 2 the main elements of the warrant recursive semantics for RP-DeLP and the unique output property for RP-DeLP programs, in Section 3 we design for this type of programs an algorithm for computing warranted conclusions in polynomial space and with an upper bound on complexity equal to P^{NP}. In contrast with DeLP and other argument-based approaches [4, 8, 21, 22], the warrant computation algorithm for RP-DeLP does not require the use of dialectical trees and it is not necessary to explicitly compute all the possible arguments for a given literal in order to discover whether it is warranted, blocked or rejected.

2 Argumentation in RP-DeLP: an overview

In order to make this paper self-contained, we will present next the main definitions that characterize the RP-DeLP framework. For further details the reader is referred to [2].

The language of RP-DeLP, denoted $\mathcal{L}_R$, is inherited from the language of logic programming, including the notions of atom, literal, rule and fact. Formulas are built over a finite set of propositional variables $p, q, ...$ which is extended with a new (negated) atom "$\sim p$" for each original atom p. Atoms of the form p or $\sim p$ will be referred as literals, and if P is a literal, we will use $\sim P$ to denote $\sim p$ if P is an atom p, and will denote p if P is a negated atom $\sim p$. Formulas of $\mathcal{L}_R$ consist of rules of the form $Q \leftarrow P_1 \wedge \ldots \wedge P_k$, where $Q, P_1, \ldots, P_k$ are literals. A fact will be a rule with no premises. We will also use the name *clause* to denote a rule or a fact. The RP-DeLP framework is based on the propositional logic $(\mathcal{L}_R, \vdash_R)$ where the inference operator $\vdash_R$ is defined by instances of the modus ponens rule of the form: $\{Q \leftarrow P_1 \wedge \ldots \wedge P_k, P_1, \ldots, P_k\} \vdash_R Q$. A set of clauses Γ will be deemed as *contradictory*, denoted $\Gamma \vdash_R \bot$, if , for some atom q, $\Gamma \vdash_R q$ and $\Gamma \vdash_R \sim q$.

A RP-DeLP program $\mathcal{P}$ is a tuple $\mathcal{P} = (\Pi, \Delta, \preceq)$ over the logic $(\mathcal{L}_R, \vdash_R)$, where $\Pi, \Delta \subseteq \mathcal{L}_R$, and $\Pi \nvdash_R \bot$. Π is a finite set of clauses representing strict knowledge (information we take for granted they hold true), Δ is another finite set of clauses representing the defeasible knowledge (formulas for which we have reasons to believe they are true). Finally, $\preceq$ is a *suitable* total pre-order on the set of defeasible formulas Δ. Suitable means that this pre-order is representable by a necessity measure N defined on the set of formulas of $\mathcal{L}_R$. Namely, $\varphi \preceq \psi$ iff $N(\varphi) \leq N(\psi)$ for each $\varphi, \psi \in \Delta \cup \Pi$, where N is a mapping $N : \mathcal{L}_R \rightarrow [0, 1]$ such that $N(\top) = 1, N(\bot) = 0$, $N(\varphi \wedge \psi) = \min(N(\varphi), N(\psi))$, and further $N(\varphi) = 1$ iff $\Pi \vdash_R \varphi$. For the sake of a simpler notation we will often refer to numerical weights for defeasible clauses and arguments rather than to the pre-ordering $\preceq$.

The notion of *argument* is the usual one. Given a RP-DeLP program $\mathcal{P}$, an argument for a literal Q of $\mathcal{L}_R$ is a pair $\mathcal{A} = \langle A, Q\rangle$, with $A \subseteq \Delta$ such that $\Pi \cup A \nvdash_R \bot$, and A is minimal (w.r.t. set inclusion) such that $\Pi \cup A \vdash_R Q$. If $A = \emptyset$, then we will call $\mathcal{A}$ a s-argument (s for strict), otherwise it will be a d-argument (d for defeasible). We define the *strength of an argument* $\langle A, Q\rangle$, written $s(\langle A, Q\rangle)$, as follows[4]:

$$s(\langle A, Q\rangle) = 1 \text{ if } A = \emptyset, \text{ and } s(\langle A, Q\rangle) = \min\{N(\psi) \mid \psi \in A\}, \text{ otherwise.}$$

The notion of *subargument* is referred to d-arguments and expresses an incremental proof relationship between arguments which is defined as follows. Let $\langle B, Q\rangle$ and $\langle A, P\rangle$ be two d-arguments such that the minimal sets (w.r.t. set inclusion) $\Pi_Q \subseteq \Pi$ and $\Pi_P \subseteq \Pi$ such that $\Pi_Q \cup B \vdash_R Q$ and $\Pi_P \cup A \vdash_R P$ verify that $\Pi_Q \subseteq \Pi_P$. Then, $\langle B, Q\rangle$ is a *subargument* of $\langle A, P\rangle$, written $\langle B, Q\rangle \sqsubset \langle A, P\rangle$, when either $B \subset A$ (strict inclusion for defeasible knowledge), or $B = A$ and $\Pi_Q \subset \Pi_A$ (strict inclusion for strict knowledge). A literal Q of $\mathcal{L}_R$ is called *justifiable* w.r.t. $\mathcal{P}$ if there exists an argument for Q, i.e. there exists $A \subseteq \Delta$ such that $\langle A, Q\rangle$ is an argument.

The following notion of acceptable argument with respect to a set (possibly empty) of justifiable conclusions W plays a key role to formalize the recursive warrant semantics. If we think of W of a consistent set of already warranted conclusions, an acceptable argument captures the idea of an argument which is based on subarguments already warranted. Let W be a set of justifiable conclusions which is consistent w.r.t. Π, i.e. $\Pi \cup W \nvdash_R \bot$. A d-argument $\mathcal{A} = \langle A, Q\rangle$ is an *acceptable argument* for Q w.r.t. W iff:

1. if $\langle B, P\rangle$ is a subargument of $\langle A, Q\rangle$ then $P \in W$
2. $\Pi \cup W \cup \{Q\} \nvdash_R \bot$

The usual notion of attack or defeat relation in an argumentation system is binary. However in certain situations, the conflict relation among arguments is hardly representable as a binary relation. For instance, consider the following RP-DeLP program $\mathcal{P} = (\Pi, \Delta, \preceq)$ with $\Pi = \{\sim p \leftarrow a \wedge b\}$ and $\Delta = \{a, b, p\}$, and consider just one level for defeasible information Δ. Clearly, $\mathcal{A}_1 = \langle\{p\}, p\rangle, \mathcal{A}_2 = \langle\{b\}, b\rangle, \mathcal{A}_3 = \langle\{a\}, a\rangle$ are arguments that justify p, b and a respectively, and which do not pair-wisely generate a conflict. Indeed, $\Pi \cup \{a, b\} \nvdash_R \bot$, $\Pi \cup \{a, p\} \nvdash_R \bot$ and $\Pi \cup \{b, p\} \nvdash_R \bot$. However the three arguments are collectively conflicting since $\Pi \cup \{a, b, p\} \vdash_R \bot$, hence in this program $\mathcal{P}$ there is a non-binary conflict relation among several arguments. Next we formalize this notion of collective conflict among acceptable arguments and which arises when we compare them with the strict part of a RP-DeLP program.

Let $\mathcal{P} = (\Pi, \Delta, \preceq)$ be a RP-DeLP program, let W be a consistent set of justifiable conclusions w.r.t. Π and let $\mathcal{A}_1 = \langle A_1, L_1\rangle, \ldots, \mathcal{A}_k = \langle A_k, L_k\rangle$ be acceptable arguments w.r.t. W of a same strength, i.e. such that $s(\langle A_1, L_1\rangle) = \ldots = s(\langle A_k, L_k\rangle)$. We say that the set of arguments $\{\mathcal{A}_1, \ldots, \mathcal{A}_k\}$ generates a conflict w.r.t. W iff the two following conditions hold:

(C) The set of argument conclusions $\{L_1, \ldots, L_k\}$ is contradictory w.r.t. $\Pi \cup W$, i.e. $\Pi \cup W \cup \{L_1, \ldots, L_k\} \vdash_R \bot$.

[4] Actually, several necessity measures N may lead to a same pre-order $\preceq$, but we can take any of them to define the degree of strength since only the relative ordering is what matters.

(M) The set $\{\mathcal{A}_1, \ldots, \mathcal{A}_k\}$ is minimal w.r.t. set inclusion satisfying (C), i.e. if $S \subset \{L_1, \ldots, L_k\}$, then $\Pi \cup W \cup S \not\vdash_R \bot$.

In the above example, arguments $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ are acceptable w.r.t. Π and the empty set of conclusions $W = \emptyset$ and, according to our definition, it is clear that the set of acceptable arguments $\{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}$ for p, b and a respectively, generates a collective conflict (with respect to $W = \emptyset$). The intuition is that this collective conflict should block the conclusions a, b and p to be warranted, and therefore, for instance, $\mathcal{A}_4 = \langle\{a, b\}, \sim p\rangle$ is an argument for $\sim p$, but $\sim p$ should be a rejected conclusion since $\mathcal{A}_4$ is not acceptable w.r.t. $W = \emptyset$ because $\mathcal{A}_2$ and $\mathcal{A}_3$ are subarguments of $\mathcal{A}_4$ but obviously $a, b \notin W$. This general notion of collective conflict is used to define a recursive semantics for warranted conclusions of a RP-DeLP program.

Actually we define an output of a RP-DeLP program $\mathcal{P} = (\Pi, \Delta, \preceq)$ as a pair $(Warr, Block)$ of warranted and blocked conclusions (literals) respectively, all of them based on warranted information but, while warranted conclusions do not generate any conflict, blocked conclusions do. Because we are considering several levels of strength among arguments, the construction of the sets of conclusions *Warr* and *Block* is done level-wise, starting from the highest level and iteratively going down from one level to next level below. If $1 > \alpha_1 > \ldots > \alpha_p \geq 0$ are the strengths of d-arguments that can be built within $\mathcal{P}$, we define *d-Warr* $= \{$*d-Warr*$(\alpha_1), \ldots,$ *d-Warr*$(\alpha_p)\}$ and *Block* $= \{$*Block*$(\alpha_1), \ldots,$ *Block*$(\alpha_p)\}$, where *d-Warr*(α_i) and *Block*(α_i) are respectively the sets of the warranted and blocked justifiable conclusions with strength α_i. In the following, we write *d-Warrr*$(> \alpha_i)$ to denote $\cup_{\beta > \alpha_i}$*d-Warr*(β), and analogously for *Block*$(> \alpha_i)$, taking *d-Warr*$(> \alpha_1) = \emptyset$ and *Block*$(> \alpha_1) = \emptyset$.

Formally, an *output for a RP-DeLP program* $\mathcal{P} = (\Pi, \Delta, \preceq)$ is any pair (*Warr*, *Block*), where *Warr* = *s-Warr* $\cup$ *d-Warr* with *s-Warr* $= \{Q \mid \Pi \vdash_R Q\}$, such that *d-Warr* and *Block* are required to satisfy the following recursive constraints:

1. A d-argument $\langle A, Q\rangle$ of strength α_i is called *valid* (or not rejected) if it satisfies the following three conditions[5]:
   (i) for every subargument $\langle B, P\rangle \sqsubset \langle A, Q\rangle$ of strength α_i, $P \in$ *d-Warr*(α_i);
   (ii) $\langle A, Q\rangle$ is acceptable w.r.t. $W =$ *d-Warr*$(> \alpha_i) \cup \{P \mid \langle B, P\rangle \sqsubset \langle A, Q\rangle$ *and* $s(\langle B, P\rangle) = \alpha_i\}$;
   (iii) $Q \notin$ *d-Warr*$(> \alpha_i) \cup$ *Block*$(> \alpha_i)$ and $\sim Q \notin$ *Block*$(> \alpha_i)$;
2. For every valid argument $\langle A, Q\rangle$ of strength α_i we have that
   - $Q \in$ *d-Warr*(α_i) whenever there does not exist a set G of valid arguments of strength α_i such that
     (i) $\langle A, Q\rangle \not\sqsubset \langle C, R\rangle$ for all $\langle C, R\rangle \in G$
     (ii) $G \cup \{\langle A, Q\rangle\}$ generates a conflict w.r.t. $W =$ *d-Warr*$(> \alpha_i) \cup \{P \mid$ there exists $\langle B, P\rangle \sqsubset \langle D, L\rangle$ for some $\langle D, L\rangle \in G \cup \{\langle A, Q\rangle\}\}$
   - otherwise, $\varphi \in$ *Block*(α_i).

[5] Notice that if $\langle A, Q\rangle$ is a d-argument, $A \neq \emptyset$ and, because of the notion of argument, $\Pi \not\vdash_R Q$ and hence $Q \notin$ *s-Warr*. Moreover, if $\langle A, Q\rangle$ is an acceptable d-argument w.r.t. *d-Warr*$(> \alpha_i)$, then $\langle A, Q\rangle$ is valid whenever condition (iii) holds.

The intuition underlying this definition is as follows: a d-argument $\langle A, Q\rangle$ of strength α_i is either warranted or blocked whenever for every subargument $\langle B, P\rangle$ of $\langle A, Q\rangle$, P is warranted and there does not exist a different valid argument for Q of strength greater than α_i; then, it is eventually warranted if Q is not involved in any conflict, otherwise it is blocked.

For instance, consider the RP-DeLP program $\mathcal{P}$ of the previous example with $\Pi = \{\sim p \leftarrow a \wedge b\}$ and $\Delta = \{a, b, p\}$ extended with three levels of defeasibility as follows: $p \prec b \prec a$. Assume α_1, α_2 and α_3 are the levels of a, b and c respectively, obviously with $1 > \alpha_1 > \alpha_2 > \alpha_3$. Then, *s-Warr* $= \emptyset$ and the argument for $\langle\{a\}, a\rangle$ is the only valid argument with strength α_1. Then, at level α_1, we get *d-Warr*$(\alpha_1) = \{a\}$ and *Block*$(\alpha_1) = \emptyset$. At level α_2, we have that $\langle\{b\}, b\rangle$ is a valid argument w.r.t. *d-Warr*(α_1) because $\Pi \cup \{a\} \cup \{b\} \nvdash_R \bot$, and $\langle\{b\}, b\rangle$ does not produce any conflict at level α_2. Then, b is a warranted conclusion at level α_2, and thus, $\langle\{a, b\}, \sim p\rangle$ is also a valid argument w.r.t. *d-Warr*$(\alpha_1) \cup \{b\}$. Hence, at level α_2 we get *d-Warr*$(\alpha_2) = \{b, \sim p\}$ and *Block*$(\alpha_1) = \emptyset$. Finally, at level α_3 the argument $\langle\{p\}, p\rangle$ for p is not valid w.r.t. *d-Warr*$(\alpha_1) \cup$ *d-Warr*(α_2) since $\Pi \cup \{a\} \cup \{b, \sim p\} \cup \{p\} \vdash_R \bot$, and thus, at level α_3 we get *d-Warr*(α_3) = *Block*$(\alpha_3) = \emptyset$.

It can be proven that if (*Warr*, *Block*) is an output for a RP-DeLP program $\mathcal{P} = (\Pi, \Delta, \preceq)$, the set *Warr* of warranted conclusions is indeed non-contradictory and satisfies the indirect consistency property ($\Pi \cup$ *Warr* $\nvdash_R \bot$) and it is closed with respect to the strict knowledge (if $\Pi \cup$ *Warr* $\vdash_R Q$ then $Q \in$ *Warr*).

In [2] we showed that, in some cases, a RP-DeLP program may have multiple outputs $(Warr, Block)$ due to some circular definitions of warranty that emerge when considering conflicts among arguments. Such circular definitions of warranty are identified by means of what we called *warrant dependency graph* of a RP-DeLP program. Intuitively, the warrant dependency graph for a set of arguments represents conflict and support dependences among arguments with respect to a set of justified conclusions.

For instance, consider a RP-DeLP program with an empty set of strict clauses and the set of defeasible clauses $\Delta = \{p, q, \sim p \leftarrow q, \sim q \leftarrow p\}$ with just one defeasibility level. Obviously, *s-Warr* $= \emptyset$ and the arguments $\mathcal{A}_1 = \langle\{p\}, p\rangle$ and $\mathcal{A}_2 = \langle\{q\}, q\rangle$ for conclusions p and q, respectively, are valid arguments with respect to *s-Warr*. Now consider the arguments for conclusions $\sim p$ and $\sim q$; i.e. $\mathcal{B}_1 = \langle\{q,\ \sim p \leftarrow q\}, \sim p\rangle$ and $\mathcal{B}_2 = \langle\{p,\ \sim q \leftarrow p\}, \sim q\rangle$. In this example, the arguments $\mathcal{A}_1$ and $\mathcal{A}_2$ are valid arguments, and thus, conclusions p and q may be warranted or blocked but not rejected. Moreover, the argument $\mathcal{B}_1$ may be valid whenever the conclusion q is warranted, and the argument $\mathcal{B}_2$ may be valid whenever the conclusion q is warranted. However, if the argument $\mathcal{B}_1$ is valid, then p and $\sim p$ are blocked conclusions, and if the argument $\mathcal{B}_2$ is valid, then q and $\sim q$ are blocked conclusions. Hence, in that case we have two possible outputs: $(Warr_1, Block_1)$ with $Warr_1 = \{p\}$ and $Block_1 = \{q, \sim q\}$, and $(Warr_2, Block_2)$ with $Warr_2 = \{q\}$ and $Block_2 = \{p, \sim p\}$.

3 On the computation of the unique output

From a computational point of view, the unique output property for RP-DeLP programs can be checked by means of a level-wise procedure, starting from the highest level and

iteratively going down from one level to next level below, and for every level verifying that there is no cycle in the graph for every valid argument with respect to the set of warranted conclusions at previous levels and the current level. Next we define an algorithm which implements this level-wise procedure computing warranted and blocked conclusions until a cycle is found or the unique output is obtained. In the following we use the notation $W(1)$ for *s-Warr*, $W(\alpha)$ for *d-Warr*(α) and $B(\alpha)$ for *Block*(α). Then, W denotes $\cup_{1\geq\alpha>0}W(\alpha)$, B denotes $\cup_{1>\alpha>0}B(\alpha)$, $W(\geq\alpha)$ denotes $\cup_{\beta\geq\alpha}W(\beta)$ and $B(\geq\alpha)$ denotes $\cup_{\beta\geq\alpha}B(\beta)$.

Algorithm `Computing warranted conclusions`
Input $\mathcal{P} = (\Pi, \Delta, \preceq)$: a RP-DeLP program
Output
$unicity$: Boolean value for the unique output property
(W, B): unique output for $\mathcal{P}$
Method
$unicity$:= True
$W(1) := \{Q \mid \Pi \vdash_R Q\}$
$B := \emptyset$
$\alpha :=$ `maximum_defeasibility_level`(Δ)
while ($unicity$ and $\alpha > 0$)
    `level_computing`(α, W, B, $unicity$)
    $\alpha :=$ `next_defeasibility_level`(Δ)
end while
end algorithm `Computing warranted conclusions`

The algorithm `Computing warranted conclusions` first computes the set of warranted conclusions $W(1)$ form the set of strict clauses Π. Then, for each defeasibility level $1 > \alpha > 0$, the procedure `level_computing` determines all warranted and blocked conclusions with necessity degree α whenever there does not exist a recursive warranty definition between arguments.

Procedure `level_computing` (**in** α; **in_out** W, B, $unicity$)
VC : = $\{Q \mid \langle C, Q\rangle$ with strength α is valid w.r.t. $(W, B)\}$
while ($unicity$ and *VC* $\neq \emptyset$)
    while ($\exists\, Q \in$ *VC* $\mid \neg$ `conflict`(α, Q, *VC*, W, `not_depend`(α, Q, *VC*, W, B)))
        $W(\alpha) := W(\alpha) \cup \{Q\}$
        VC := *VC*$\backslash\{Q\} \cup \{P \mid \langle C, P\rangle$ with strength α is valid w.r.t. $(W, B)\}$
    end while
    $I := \{Q \in$ *VC* $\mid$ `conflict`(α, Q, *VC*, W, $\emptyset$)$\}$
    $B(\alpha) := B(\alpha) \cup I$
    VC := *VC*$\backslash I$
    if (for some $Q \in$ *VC* there is a cycle in the graph w.r.t. W) **then** $unicity$:= False
end while
end procedure `level_computing`

For every level α the procedure `level_computing` computes the set of valid conclusions *VC* with respect to the current sets of warranted and blocked conclusions (W, B)[6]. The set of valid conclusions *VC* is dynamically updated depending on new warranted conclusions. The procedure `level_computing` is based on the two following auxiliary functions.

Function `conflict`(**in** α, Q, *VC*, W, D): **return Boolean**
return ($\exists$ $S \subseteq$ *VC*$\backslash\{Q\} \cup D$ such that $\Pi \cup W(\geq \alpha) \cup S \not\vdash_R \bot$
and $\Pi \cup W(\geq \alpha) \cup S \cup \{Q\} \vdash_R \bot$)
end function `conflict`

Function `not_depend`(**in** α, Q, *VC*, W, B): **return** set of conclusions
D := $\{P \notin$ *VC* $\mid \langle C, P\rangle$ with strength α is almost valid w.r.t. (W, B)
and literal Q is not a conclusion in $C\}$
return(D)
end function `not_depend`

The function `conflict` checks conflicts among Q, the set of valid conclusions *VC*$\backslash\{Q\}$ and the set of conclusions D. The set of conclusions D takes two different values: the empty set and what we call *almost valid* conclusions with respect to (W, B)[7] which do not depend on Q; i.e. conclusions which depend on some valid conclusion in *VC*$\backslash\{Q\}$, do not depend on blocked conclusions and do not generate conflicts. In fact, the function `not_depend` computes the set of conclusions D for a given literal $Q \in$ *VC*. Finally, we would remark that the existence of a cycle in the graph for some $Q \in$ *VC* with respect to the current set of warranted conclusions W can be determined by checking the stability of *VC* after two consecutive iterations instead of explicitly building the warranty dependence graph for every $Q \in$ *VC*, since the function `conflict` considers recursive warranty definitions among arguments by means of the set of conclusions D.

One of the main advantages of the warrant recursive semantics for RP-DeLP is from the implementation point of view. Actually, warrant semantics based on dialectical trees and, in general, rule-based argumentation frameworks like DeLP [7, 9], might consider an exponential number of arguments with respect to the number of rules of a given program. In contrast, in our framework, at least for the particular case of RP-DeLP programs with unique output, it is not necessary to explicitly compute all the possible arguments for a given literal, in order to discover whether it is warranted, as we can implement the previous algorithm with a worst-case complexity in P^{NP}.

First, observe that the incremental discovery of valid conclusions (set *VC* in the algorithm) can be performed in P-time, as for every defeasibility level α all that we

[6] Notice that for every level of execution α, an argument $\langle C, Q\rangle$ with strength α is *valid* w.r.t. the current sets (W, B) iff (i) $Q \notin W(\geq \alpha)$ and $Q, \sim Q \notin B(\geq \alpha)$, and (ii) $\langle C, Q\rangle$ is acceptable w.r.t. $W(\geq \alpha)$.

[7] For every level of execution α, an argument $\langle C, P\rangle$ with strength α is called *almost valid* w.r.t. the current sets (W, B) iff (i) $P \notin W(\geq \alpha)$ and $P, \sim P \notin B(\geq \alpha)$; (ii) for all $\langle E, R\rangle \sqsubset \langle C, P\rangle$ with strength $\beta > \alpha$, $R \in W(\beta)$; (iii) for all $\langle E, R\rangle \sqsubset \langle C, P\rangle$ with strength α, $R, \sim R \notin B(\geq \alpha)$; (iv) for some $\langle E, R\rangle \sqsubset \langle C, P\rangle$ with strength α, $R \notin W(\alpha)$; and (v) $\Pi \cup W(\geq \alpha) \cup \{R \mid \langle E, R\rangle \sqsubset \langle C, P\rangle$ with strength $\alpha\} \cup \{P\} \not\vdash_R \bot$.

need for a literal Q to be in *VC* is either that $Q \in \Delta$ with level α, or an α-rule with warranted body and conclusion Q. An $\alpha-$rule R is a rule with either level α or greater than α but with $Body(R) \setminus W(\geq \alpha) \neq \emptyset$, so that it can only conclude its conclusion with strength equal or less than α. Secondly, remember that we can effectively avoid building and checking the dependency graph for literals, as if the program has unique output, the set *VC* changes at every iteration of the main loop at `level_computing`. Finally, we need only to check the complexity of the following problems:

1. Whether a literal P is in the set D = `not_depend`(α, Q, *VC*, W, B). We can non-deterministically guess a subset $S \subseteq (W(\geq \alpha) \cup \Delta(\alpha))$ and check in polynomial time whether it encodes an almost valid argument for P without using Q. The set $\Delta(\alpha)$ is the set of facts with level α and $\alpha-$rules, as defined before. So, we can check whether P has such almost valid argument with an NP algorithm [8].
2. Whether the function `conflict`(α, Q, *VC*, W, D) returns true. Remark that we can non-deterministically guess a subset $S \subseteq VC \setminus \{Q\} \cup D$ and check in polynomial time whether $\Pi \cup W(\geq \alpha) \cup S \not\vdash_R \bot$ and $\Pi \cup W(\geq \alpha) \cup S \cup \{Q\} \vdash_R \bot$). So, again this can be checked with an NP algorithm.

Finally, these basic problems are solved at most a polynomial number of times in the `level_computing` procedure. In `level_computing` a literal is never inserted again in the set *VC* once it has been removed from it, and the number of iterations of the inner loop, the one that discovers warranted literals, is bounded by the size of *VC*. So, given that the outer loop of `level_computing` will end as soon as no more literals are found in *VC* or as soon as *VC* becomes stable (because no valid conclusions can be either warranted or blocked), the number of steps of the procedure is polynomially bounded by the size of the program. Also, the number of times that `level_computing` is called is bounded by the number of levels of defeasibility. So, this gives an upper bound on complexity equal to P^{NP} for discovering whether a literal is warranted in the unique output, or a function complexity of FP^{NP} for computing the unique output.

It is worth noticing that in a recent work [18], the authors have studied the complexity of the warranted formula problem in a framework for propositional argumentation with classical logic and general formulae (not only horn clauses), and they have shown the problem to be PSPACE-complete. In our case, at least for the particular case of RP-DeLP programs with unique output, the complexity is upper bounded with a subclass of PSPACE, P^{NP}, but we suspect that for general RP-DeLP programs the complexity of determining whether a literal is in the skeptical output is at least PSPACE-hard. In contrast, in abstract argumentation frameworks similar reasoning problems have lower complexity than for propositional argumentation. For example, checking whether an argument is included on some preferred extension is NP-complete [10] and checking whether an argument is included on every preferred extension (skeptical reasoning) is $coNP^{NP}$-complete [15]. Recently, the related problem of checking whether an argu-

[8] The argument found by such NP algorithm may not be minimal, but as we are concerned with the existence problem, if there is an almost valid argument, then a subset of it will be a minimal one. It can be shown that the main algorithm only stores a literal in the warrant set once it has found a minimal argument that is conflict free.

ment is in the maximal ideal extension has been shown to be between coNP and a subclass of P^{NP} [16, 14].

4 Conclusions and future work

In this paper we have designed an algorithm for computing the warranty status of arguments according to the new recursive semantics defined in [2] for defeasible argumentation with defeasibility levels, for the particular case of programs with unique output. It discovers warranted literals using a polynomial amount of memory, as it avoids maintaining sets of arguments that could be of exponential size with respect to program size, and its worst-case complexity is upper bounded with the class P^{NP}. Moreover, taking profit of the recursive semantics, it allows to recover the argument for every warranted literal and even for blocked literals it is possible to store the conflict sets that were found as responsible of their blocking. So in possible applications of this framework this algorithm could provide useful information for users in case of unexpected outputs.

Future work will be addressed in two main directions. On the one hand we plan to design an efficient version of the algorithm we have shown here, by minimizing the effective number of NP queries that have to be made during the execution of the `level_computing` procedure. There are several ways of saving NP queries. For example, every time an almost valid argument has been found for a literal Q, this argument can be saved, so next time we need to check an argument for Q, if the one saved is still almost valid (i.e. it is not based on blocked conclusions), there is no need to find a new one. Also, with the aim of obtaining an algorithm able to scale up with problem size, we will design polynomial time reductions of the NP queries to be performed to the SAT problem, so that we can take profit of state-of-the-art SAT solvers for solving the most critical subproblems during the search. We also plan to study particular cases of RP-DeLP programs that have a better worst-case complexity, looking for tractable cases, like it has been done recently for other argumentation frameworks, like the case of bipartite abstract argumentation frameworks [14].

On the other hand we plan to come to the problem of deciding which output should be considered for RP-DeLP programs with multiple outputs. A natural solution to this problem could be to adopt the intersection of all possible outputs in order to define the skeptical output as the set of those literals which are ultimately warranted. However, as stated in [19], adopting the intersection of all possible outputs can lead to an inconsistent output when some recursive situation occurs between the literals of a program. So that we plan to define an ultimately warranted conclusion for RP-DeLP programs with multiple outputs as a conclusion of the intersection which is recursively based on ultimately warranted conclusions. In fact, this idea corresponds with the maximal ideal extension defined by Dung, Mancarella and Toni [12, 13] as an alternative skeptical basis for defining collections of justified arguments in the abstract argumentation frameworks promoted by Dung [11] and Bondarenko *et al.* [5].

Acknowledgments Authors are thankful to the anonymous reviewers for their helpful comments. Research partially funded by the Spanish MICINN projects MULOG2 (TIN2007-68005-C04-01/02) and ARINF (TIN2009-14704-C03-01/03), CONSOLIDER (CSD2007-0022), and ESF

Eurocores-LogICCC/MICINN (FFI2008-03126-E/FILO), and the grant JC2009-00272 from the Ministerio de Educación.

References

1. 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):1208–1228, 2008.
2. T. Alsinet, R. Béjar, and L. Godo. A characterization of collective conflict for defeasible argumentation. In *Proc. of COMMA 2010*.
3. T. Alsinet, C.I. Chesñevar and L. Godo. A Level-based Approach to Computing Warranted Arguments in Possibilistic Defeasible Logic. In *Proc. of COMMA 2008*, pages 1–12, 2008.
4. P. Besnard and A. Hunter. *Elements of Argumentation*. The MIT Press, 2008.
5. A. Bondarenko, P.M. Dung, R.A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artif. Intell.*, 93:63–101, 1997.
6. M. Caminada and L. Amgoud. On the evaluation of argumentation formalisms. *Artif. Intell.*, 171(5-6):286–310, 2007.
7. L. Cecchi, P. Fillottrani, and G. Simari. On the complexity of delp through game semantics. In *Proc. of NMR 2006*, pages 386–394, 2006.
8. C.I. Chesñevar, A. Maguitman, and R. Loui. Logical Models of Argument. *ACM Computing Surveys*, 32(4):337–383, 2000.
9. C.I. Chesñevar, G. Simari, and L. Godo. Computing dialectical trees efficiently in possibilistic defeasible logic programming. In *Proc. of LPNMR 2005*, pages 158–171, 2005.
10. Y. Dimopoulos and A. Torres. Graph theoretical structures in logic programs and default theories. *Theoretical Computer Science*, 170(1-2):209 – 244, 1996.
11. P.M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995.
12. P.M. Dung, P. Mancarella, and F. Toni. A dialectic procedure for sceptical, assumption-based argumentation. In *Proc. of COMMA 2006*, pages 145–156, 2006.
13. P.M. Dung, P. Mancarella, and F. Toni. Computing ideal sceptical argumentation. *Artif. Intell.*, 171(10-15):642–674, 2007.
14. P.E. Dunne. The computational complexity of ideal semantics. *Artif. Intell.*, 173(18):1559–1591, 2009.
15. P.E. Dunne and T.J.M. Bench-Capon. Coherence in finite argument systems. *Artif. Intell.*, 141(1-2):187 – 203, 2002.
16. P.E. Dunne. The computational complexity of ideal semantics i: Abstract argumentation frameworks. In *Proc. of COMMA 2008*, pages 147–158, 2008.
17. A. García and G. Simari. Defeasible Logic Programming: An Argumentative Approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2004.
18. R. Hirsch and N. Gorogiannis. The complexity of the warranted formula problem in propositional argumentation. *J. of Logic and Computation*, 20(2), 2009.
19. J.L. Pollock. A recursive semantics for defeasible reasoning. In I. Rahwan and G. Simari, editors, *Argumentation in Artificial Intelligence*, chapter 9, pages 173–198. Springer, 2009.
20. H. Prakken and G. Sartor. Argument-based extended logic programming with defeasible priorities. *J. of Applied Non-classical Logics*, 7:25–75, 1997.
21. H. Prakken and G. Vreeswijk. Logical Systems for Defeasible Argumentation. In D. Gabbay and F. Guenther, editors, *Handbook of Phil. Logic*, pages 219–318. Kluwer, 2002.
22. I. Rahwan and G. Simari, editors. *Argumentation in Artificial Intelligence*. Springer, 2009.