RP-DeLP: A weighted defeasible argumentation framework based on a recursive semantics

Teresa Alsinet [a] Ramón Béjar [a] Lluís Godo [b] Francesc Guitart [a]

[a] *Department of Computer Science, Universitat de Lleida*
C/Jaume II 69, 25001 Lleida, SPAIN – *Email: {tracy,ramon,fguitart}@diei.udl.cat*

[b] *Artificial Intelligence Research Institute (IIIA), CSIC*
Campus UAB s/n, 08193 Bellaterra, SPAIN – *Email: {godo}@iiia.csic.es*

Abstract

In this paper we first define a recursive semantics for warranted formulas in a general defeasible argumentation framework by formalizing a notion of collective (non-binary) conflict among arguments. The recursive semantics for warranted formulas is based on the fact that if the argument is rejected, then all arguments built on it should also be rejected. The main characteristic of our recursive semantics is that an output (extension) of a knowledge base is a pair of sets of warranted and blocked formulas. Arguments for both warranted and blocked formulas are recursively based on warranted formulas but, while warranted formulas do not generate any collective conflict, blocked conclusions do. Formulas that are neither warranted nor blocked correspond to rejected formulas. Second we extend the general defeasible argumentation framework by attaching levels of preference to defeasible knowledge items and by providing a level-wise definition of warranted and blocked formulas. Third we formalize the warrant recursive semantics for the particular framework of Possibilistic Defeasible Logic Programming, we call this particular framework Recursive Possibilistic Defeasible Logic Programming (RP-DeLP for short), and we show its relevance in the scope of Political debates. An RP-DeLP program may have multiple outputs in case of circular definitions of conflicts among arguments. So, we tackle the problem of which output one should consider for an RP-DeLP program with multiple outputs. To this end we define the maximal ideal output of an RP-DeLP program as the set of conclusions which are ultimately warranted and we present an algorithm for computing them in polynomial space and with an upper bound on complexity equal to P^{NP}. Finally, we propose an efficient and scalable implementation of this algorithm that is based on implementing the two main queries of the system, looking for valid arguments and collective conflicts between arguments, using SAT encodings. We perform an experimental evaluation of our SAT based approach when solving test sets of instances with single and multiple preference levels for defeasible knowledge.

Key words: Defeasible Reasoning, Recursive Semantics, Collective Conflict, Rationality Postulates, SAT Encoding, Efficient Implementation

1 Introduction and motivation

Defeasible argumentation is a natural way of identifying relevant assumptions and conclusions for a given problem which often involves identifying conflicting information, resulting in the need to look for pros and cons for a particular conclusion [24]. This process may involve chains of reasoning, where conclusions are used in the assumptions for deriving further conclusions and the task of finding pros and cons may be decomposed recursively.

Defeasible Logic Programming (DeLP) [20] is a formalism that combines techniques of both logic programming and defeasible argumentation. As in logic programming, in DeLP knowledge is represented using facts and rules; however, DeLP also provides the possibility of representing defeasible knowledge under the form of weak (defeasible) rules, expressing reasons to believe in a given conclusion. In DeLP, a conclusion succeeds if it is warranted, i.e., if there exists an argument (a consistent sets of defeasible rules) that, together with the non-defeasible rules and facts, entails the conclusion, and moreover, this argument is found to be undefeated by the warrant procedure which builds a dialectical tree containing all arguments that challenge this argument, and all counterarguments that challenge those arguments, and so on, recursively. Actually, dialectical trees systematically explore the universe of arguments in order to present an exhaustive synthesis of the relevant chains of pros and cons for a given conclusion. In fact, the interpreter for DeLP [19] (http://lidia.cs.uns.edu.ar/DeLP) takes a knowledge base (program) P and a conclusion (query) Q as input, and it then returns one of the following four possible answers: YES, if Q is warranted from P; NO, if the complement of Q is warranted from P; UNDECIDED, if neither Q nor its complement are warranted from P; or UNKNOWN, if Q is not in the language of the program P.

Possibilistic Defeasible Logic Programming (P-DeLP) [6] is a rule-based argumentation framework which is an extension of DeLP in which defeasible rules are attached with weights (belonging to the real unit interval $[0, 1]$) expressing their belief or preference strength and formalized as necessity degrees. As many other argumentation frameworks [13,24], P-DeLP can be used as a vehicle for facilitating rationally justifiable decision making when handling incomplete and potentially inconsistent information. Actually, given a P-DeLP program, justifiable decisions correspond to warranted conclusions (with a maximum necessity degree), that is, those which remain undefeated after an exhaustive dialectical analysis of all possible arguments for and against.

In [11] Caminada and Amgoud proposed three *rationality postulates* which every rule-based argumentation system should satisfy. One of such postulates (called *Indirect Consistency*) claims that closure under the consequence operator of the underlying logic of warranted conclusions with respect to the set of strict rules must be consistent. In [11] a number of rule-based argumentation systems were identi-

fied in which such postulate does not hold (including DeLP [20] and Prakken & Sartor's [23], among others). As a way to solve this problem, the use of *transposed rules* is proposed in [11] to extend the representation of strict rules. Recently, in [7] Amgoud proposes a new rationality postulate (called *Closure under Subarguments*) which rule-based argumentation systems should satisfy. This postulate claims that the acceptance of an argument should imply also the acceptance of all its subarguments which reflect the different premises on which the argument is based.

Since the dialectical analysis based semantics of P-DeLP for warranted conclusions does not satisfy the Indirect Consistency postulate, our aim in this paper is to characterize P-DeLP with a new warrant semantics. To this end, we consider recursive semantics for defeasible argumentation as defined by Pollock in [22] where recursive definitions of conflict between arguments were characterized by means of *inference-graphs*, representing (binary) support and attack (pros and cons) relations between the conclusions of arguments. On the other hand, recursive semantics are based on the fact that if an argument is rejected, then all arguments built on it should also be rejected. On the other hand, as stated in [22], recursive definitions of conflict among arguments can cause to different outputs (extensions) for warranted conclusions. Hence, we need a new and general notion of conflict among the conclusions of arguments which ensures the Indirect Consistency postulate and which allows us to safely reason about recursive definitions of conflict between arguments.

The first contribution of this paper is to define a recursive semantics for warranted formulas in a quite general framework (without levels of strength) by formalizing a new collective (non-binary) notion of conflict between arguments. The main characteristic of our recursive semantics is that an output (extension) of a knowledge base is a pair of sets of warranted and blocked formulas. Arguments for both warranted and blocked formulas are recursively based on warranted formulas but, while warranted formulas do not generate any conflict with the set of already warranted formulas and the strict part of the knowledge base (information we take for granted they hold true), blocked formulas do. Formulas that are neither warranted nor locked correspond to rejected formulas. On the one hand, the key feature that address our warrant recursive semantics corresponds with the closure under subarguments postulate recently proposed by Amgoud and that is, if an argument is excluded from an output, then all arguments built on it should also be excluded from that output. On the other hand, the idea of defining an argumentation framework on the basis of conflicting sets of arguments was proposed in [26]. The difference with the collective conflict among arguments in our framework is that in [26] the conflict is not relative to a set of already warranted conclusions and the strict part of the knowledge base. Finally, in contrast with DeLP and other argument-based approaches [13,24,9,25], our argumentation framework do not require the use of dialectical trees as underlying structures for characterizing the semantics for warranted conclusions and ensures the three *rationality postulates* defined by Caminada and Amgoud in [11] without extending the representation of strict rules with

transposed rules.

The second contribution of this paper is to extend the recursive semantics based on the collective notion of conflict between arguments to a general argumentation framework with defeasibility (preference) levels by providing a level-wise definition of warranted and blocked conclusions. We characterize the properties of outputs in terms of some propagation criteria between defeasibility levels of warranted and blocked conclusions.

The third contribution of this paper is to specialize the warrant recursive semantics with defeasibility levels to the particular framework of P-DeLP, we refer to this formalism as *Recursive* P-DeLP (RP-DeLP for short). In [5] we proposed a level-wise approach to compute warranted conclusions which distinguished two types of conflicts between arguments, direct and indirect conflicts. Direct conflicts were due to binary attacks emerging from defeasible knowledge, while indirect conflicts were due to collective attacks emerging from strict knowledge. Then, direct conflicts invalidated indirect conflicts, and thus, an implicit evaluation order between conflicts were established in [5]. In contrast, the collective notion of conflict for RP-DeLP do not assume any implicit order of evaluation of conflicts which ensures that if a conclusion is included as warrant in an output, then the argument that justify the conclusion is not involved in any kind of conflict. Due to some circular definitions of warranty among arguments that emerge in case of circular definitions of conflicts among arguments, the recursive semantics for warranted conclusions may result in multiple outputs for RP-DeLP programs. Following the approach of Pollock [22], we characterize circular definitions of conflict among arguments that cause different outputs by means of what we call *Warrant Dependency Graphs* representing support and (collective) conflict relations between the conclusions of arguments. Moreover, for RP-DeLP programs with multiple outputs we consider the problem of deciding the set of conclusions that can be ultimately warranted. The usual skeptical approach would be to adopt the intersection of all possible outputs. However, in addition to the computational limitation, as stated in [22], adopting the intersection of all outputs may lead to an inconsistent output (in the sense of violating the base of the underlying recursive warrant semantics) in case some particular recursive situation among literals of a program occurs. Intuitively, for a conclusion, to be in the intersection does not guarantee the existence of an argument for it that is recursively based on ultimately warranted conclusions. With the aim of computing single outputs and based on the idea defined by Dung, Mancarella and Toni [16,17] as an alternative skeptical basis for defining collections of justified arguments in abstract argumentation frameworks, we characterize what we call *Maximal Ideal Output* for an RP-DeLP program based on a recursive level-wise definition considering at each level the maximum set of conclusions based on warranted information and not involved in neither a conflict nor a circular definition of conflict.

The fourth contribution of the paper is the development and experimental validation of an interpreter that computes the maximal ideal output for an RP-DeLP program.

To this end, first we define an efficient algorithm that computes the maximal ideal output in polynomial space and with an upper bound on complexity equal to P^{NP}. Second we present SAT encodings for the two main combinatorial subproblems that arise when computing warranted and blocked conclusions of he maximal ideal output for an RP-DeLP program, so that we can take profit of existing state-of-the-art SAT solvers for solving instances of big size. Finally we present empirical results obtained with an implementation of our algorithm that uses these SAT encodings. The results show that, at least on randomly generated instances, the practical complexity is strongly dependent on the size of the strict part of the program, as for the same number of variables RP-DeLP programs with different size for their strict part can range from trivially solvable to exceptionally hard. Moreover, the experimental results also show that the fraction of defeasible knowledge considered at each defeasible level is also relevant in defining the tractability and scalability of RP-DeLP programs.

This paper extends our previous work in [1,3] by providing new running examples, the characterization of the properties of the framework, the algorithm for the computation of the maximal ideal output for an RP-DeLP program based on SAT encodings, experimental results, and proofs for all outcomes. The rest of the paper is organized as follows. In Section 2, we define a general defeasible argumentation framework with recursive semantics. In Section 3, we introduce several levels of defeasibility or preference among different pieces of defeasible knowledge. In Section 4, we particularize the recursive warrant semantics to the case of the P-DeLP programs and we provide some examples in the context of political debates. In Section 5, we define the maximal ideal output for RP-DeLP programs and in Section 6, we present an algorithm for its computation. In Section 7, we present SAT encodings for the two main queries performed in the algorithm and in Section 8, we study the scaling behavior of the (average) computational cost of our implementation. Finally, in Section 9, we present some concluding remarks.

2 A general defeasible argumentation framework with recursive semantics

We will start by considering a rather general framework for defeasible argumentation based on a propositional logic $(\mathcal{L}, \vdash)$ with a special symbol $\bot$ for contradiction [1]. For any set of formulas A, if $A \vdash \bot$ we will say that A is contradictory, while if $A \nvdash \bot$ we will say that A is consistent. A knowledge base (KB) is a triplet $\mathcal{P} = (\Pi, \Delta, \Sigma)$, where $\Pi, \Delta, \Sigma \subseteq \mathcal{L}$, and $\Pi \nvdash \bot$. Π is a finite set of formulas representing strict knowledge (formulas we take for granted they hold to be true), Δ is another finite set of formulas representing the defeasible knowledge (formulas for which we have reasons to believe they are true) and Σ denotes the set of formulas

[1] If not stated otherwise, in this and next sections $(\mathcal{L}, \vdash)$ may be taken as classical propositional logic.

(conclusions) over which arguments can be built. In many argumentation systems, e.g. in rule-based argumentation systems, Σ is taken to be a set of literals.

The notion of *argument* is the usual one. Given a KB $\mathcal{P}$, an argument for a formula $\varphi \in \Sigma$ is a pair $\mathcal{A} = \langle A, \varphi \rangle$, with $A \subseteq \Delta$ such that:

(1) $\Pi \cup A \nvdash \bot$, and
(2) A is minimal (with respect to set inclusion) such that $\Pi \cup A \vdash \varphi$.

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). The notion of *subargument* is refereed to d-arguments and expresses an incremental proof relationship between arguments which is formalized as follows.

Definition 1 (Subargument) *Let $\langle B, \psi \rangle$ and $\langle A, \varphi \rangle$ be two d-arguments such that the minimal sets (with respect to set inclusion) $\Pi_\psi \subseteq \Pi$ and $\Pi_\varphi \subseteq \Pi$ such that $\Pi_\psi \cup B \vdash \psi$ and $\Pi_\varphi \cup A \vdash \varphi$ verify that $\Pi_\psi \subseteq \Pi_\varphi$. Then, $\langle B, \psi \rangle$ is* a subargument *of $\langle A, \varphi \rangle$, written $\langle B, \psi \rangle \sqsubset \langle A, \varphi \rangle$, when one of the following conditions holds:*

- *$B \subset A$ (strict inclusion for defeasible knowledge),*
- *$B = A$ and $\Pi_\psi \subset \Pi_\varphi$ (strict inclusion for strict knowledge), or*
- *$B = A$, $\Pi_\psi = \Pi_\varphi$ and $\psi \vdash \varphi$ but $\varphi \nvdash \psi$.*

Notice that if $(\Pi, \Delta, \Sigma) = (\{r\}, \{r \to p \wedge q\}, \{p, q, p \wedge q\})$ and $A = \{r \to p \wedge q\}$ then $\mathcal{A}_1 = \langle A, p \rangle$, $\mathcal{A}_2 = \langle A, q \rangle$ and $\mathcal{A}_3 = \langle A, p \wedge q \rangle$ are arguments for different formulas with a same support and thus, in our framework, $\mathcal{A}_3 \sqsubset \mathcal{A}_1$ and $\mathcal{A}_3 \sqsubset \mathcal{A}_2$ are the subargument relations between arguments $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ since $p \wedge q \vdash p$, $p \wedge q \vdash q$, $p \nvdash p \wedge q$ and $q \nvdash p \wedge q$.

A formula $\varphi \in \Sigma$ will be called *justifiable conclusion* with respect to $\mathcal{P}$ if there exists an argument for φ, i.e. there exists $A \subseteq \Delta$ such that $\langle A, \varphi \rangle$ is an argument.

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, mainly (but not only) when $\Pi \neq \emptyset$. For instance, consider the following KB $\mathcal{P}_1 = (\Pi, \Delta, \Sigma)$ with

$$\Pi = \{a \wedge b \to \neg p\},\ \Delta = \{a, b, p\} \text{ and } \Sigma = \{a, b, p, \neg p\}.$$

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, $\Pi \cup \{a, b\} \nvdash \bot$, $\Pi \cup \{a, p\} \nvdash \bot$ and $\Pi \cup \{b, p\} \nvdash \bot$. However the three arguments are collectively conflicting since $\Pi \cup \{a, b, p\} \vdash \bot$, hence in $\mathcal{P}_1$ there is a non-binary conflict relation among several arguments.

In the following we formalize this notion of collective conflict among what we will

call valid arguments which arises when we compare them with the strict part of the knowledge base and a consistent set of justifiable conclusions W. If we think of W of a consistent set of already warranted conclusions, a valid argument will capture the idea of an argument which is based on subarguments already warranted.

Definition 2 (Conflict among arguments) *Let $\mathcal{P} = (\Pi, \Delta, \Sigma)$ be a KB and let $W \subseteq \Sigma$ be a consistent set. We say that a set of arguments $\{\langle A_1, \varphi_1\rangle, \ldots, \langle A_k, \varphi_k\rangle\}$* minimally conflicts *with respect to W iff the two following conditions hold:*

(C) The set of argument conclusions $\{\varphi_1, \ldots, \varphi_k\}$ is contradictory with respect to W, i.e. it holds that $\Pi \cup W \cup \{\varphi_1, \ldots, \varphi_k\} \vdash \bot$.
(M) The set $\{\mathcal{A}_1, \ldots, \mathcal{A}_k\}$ is minimal with respect to set inclusion satisfying (C), i.e. if $S \subset \{\varphi_1, \ldots, \varphi_k\}$, then $\Pi \cup W \cup S \nvdash \bot$.

Notice that if a set of arguments $G = \{\langle A_1, \varphi_1\rangle, \ldots, \langle A_k, \varphi_k\rangle\}$ minimally conflicts with respect to a set of conclusions W, then the arguments $\langle A_i, \varphi_i\rangle$ cannot be s-arguments, i.e. for each i, $A_i \neq \emptyset$. Indeed, if $A_i = \emptyset$ for some i, then $\Pi \vdash \varphi_i$, and hence G would not satisfy the minimality Condition (M).

Consider the previous KB $\mathcal{P}_1$, and the set of arguments $\{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}$ for p, b and a, respectively, and let $W = \cup_{i=1,\ldots,3}\{\psi \mid \langle B, \psi\rangle \sqsubset \mathcal{A}_i\} = \emptyset$. According to the previous definition, it is clear that the set of arguments $\{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}$ minimally conflicts with respect to $\Pi = \{a \wedge b \rightarrow \neg p\}$. The intuition is that this collective conflict should block the conclusions a, b and p to be warranted. Now, this general notion of conflict is used to define a recursive semantics for warranted conclusions of a knowledge base. Actually we define an output of a KB $\mathcal{P} = (\Pi, \Delta, \Sigma)$ as a pair $(\mathit{Warr}, \mathit{Block})$ of subsets of Σ of warranted and blocked conclusions respectively all of them based on warranted information but while warranted conclusions do not generate any conflict, blocked conclusions do.

Definition 3 (Output for a KB) *An output for a KB $\mathcal{P} = (\Pi, \Delta, \Sigma)$ is any pair $(\mathit{Warr}, \mathit{Block})$, where $\mathit{Warr} \cap \mathit{Block} = \emptyset$, $\mathit{Warr} \cup \mathit{Block} \subseteq \Sigma$ and $\{\varphi \in \Sigma \mid \Pi \vdash \varphi\} \subseteq \mathit{Warr}$, satisfying the following recursive constraints:*

(1) $\varphi \in \mathit{Warr} \cup \mathit{Block}$ iff there exists an argument $\langle A, \varphi\rangle$ such that for every $\langle B, \psi\rangle \sqsubset \langle A, \varphi\rangle$, $\psi \in \mathit{Warr}$. In that case call valid *such an argument $\langle A, \varphi\rangle$.*
(2) For each valid argument $\langle A, \varphi\rangle$:
- *$\varphi \in \mathit{Block}$ whenever there exists a set of valid arguments G such that*
  (i) $\langle A, \varphi\rangle \not\sqsubset \langle C, \chi\rangle$ for all $\langle C, \chi\rangle \in G$, and
  (ii) $\{\langle A, \varphi\rangle\} \cup G$ minimally conflicts with respect to the set $W = \{\psi \mid \langle B, \psi\rangle \sqsubset \langle D, \gamma\rangle$ for some $\langle D, \gamma\rangle \in G \cup \{\langle A, \varphi\rangle\}\}$.
- *otherwise, $\varphi \in \mathit{Warr}$.*

The intuition underlying this definition is as follows: an argument $\langle A, \varphi\rangle$ is either warranted or blocked whenever for each subargument $\langle B, \psi\rangle$ of $\langle A, \varphi\rangle$, ψ is warranted; then, it is eventually blocked if φ is involved in some conflict, otherwise it

is warranted.

Notice that if an argument $\langle A, \varphi \rangle$ is warranted, and $\langle A, \psi \rangle$ is another argument, then $\langle A, \psi \rangle$ is warranted as well.

Example 4 *Consider the KB* $\mathcal{P}_2 = (\Pi, \Delta, \Sigma)$*, with*

$$\Pi = \{a \to y, b \wedge c \to \ \neg y\},\ \Delta = \{a, b, c, \neg c\} \text{ and } \Sigma = \{a, b, c, \neg c, y, \neg y\}.$$

According to Definition 3 s-Warr $= \emptyset$ *and the arguments* $\langle \{a\}, a \rangle$*,* $\langle \{b\}, b \rangle$*,* $\langle \{c\}, c \rangle$ *and* $\langle \{\neg c\}, \neg c \rangle$ *are valid. Now, for every such valid argument there exists a set of valid arguments which minimally conflicts: indeed both sets of valid arguments* $\{\langle \{a\}, a \rangle, \langle \{b\}, b \rangle, \langle \{c\}, c \rangle\}$ *and* $\{\langle \{c\}, c \rangle, \langle \{\neg c\}, \neg c \rangle\}$ *minimally conflict (since* $\Pi \cup \{a, b, c\} \vdash \bot$ *and* $\Pi \cup \{c, \neg c\} \vdash \bot$*). Therefore* a*,* b*,* c *and* $\neg c$ *are blocked conclusions. On the other hand, the arguments* $\langle \{a, b\}, \neg c \rangle$*,* $\langle \{a\}, y \rangle$ *and* $\langle \{b, c\}, \neg y \rangle$ *are not valid since they are based on conclusions which are not warranted. Hence* y *and* $\neg y$ *are considered as rejected conclusions. Thus, the (unique) output for* $\mathcal{P}$ *is the pair* $(\mathit{Warr}, \mathit{Block}) = (\emptyset, \Delta)$*. Intuitively this output for* $\mathcal{P}_2$ *expresses that all conclusions in Block are (individually) valid, however all together are contradictory with respect to* Π*.*

We remark that, as it will be discussed in Section **??**, a KB may have multiple outputs. For instance, consider the KB $\mathcal{P}_3 = (\Pi, \Delta, \Sigma)$ with

$$\Pi = \emptyset,\ \Delta = \{p, q, \neg p \vee \neg q\} \text{ and } \Sigma = \{p, q, \neg p, \neg q\}.$$

Then, according to Definition 3, the pairs

$$(\mathit{Warr}_1, \mathit{Block}_1) = (\{p\}, \{q, \neg q\}) \quad \text{and}$$
$$(\mathit{Warr}_2, \mathit{Block}_2) = (\{q\}, \{p, \neg p\})$$

are two outputs for $\mathcal{P}_3$.

It can be proven that if $(\mathit{Warr}, \mathit{Block})$ is an output for a KB (Π, Δ, Σ), the set *Warr* of warranted conclusions is indeed non-contradictory and satisfies indirect consistency with respect to the strict knowledge.

Proposition 5 (Indirect consistency) *Let* $\mathcal{P} = (\Pi, \Delta, \Sigma)$ *be a KB and let the pair* $(\mathit{Warr}, \mathit{Block})$ *be an output for* $\mathcal{P}$*. Then,* $\Pi \cup \mathit{Warr} \nvdash \bot$*.*

Proof: By Definition 3, for every $\varphi \in \mathit{Warr}$ there does not exist a set $W \subseteq \mathit{Warr}$ such that $\Pi \cup W \cup \{\varphi\} \vdash \bot$, and therefore, $\Pi \cup W \nvdash \bot$ for all $W \subseteq \mathit{Warr}$. □

In the following we will see that the general defeasible argumentation framework we have defined satisfies the *closure* postulate (in the sense of Caminada and Amgoud [11]) with respect to the strict knowledge depending on how the set of formu-

las Σ over which arguments can be built is defined. For instance, consider the KB $\mathcal{P}_4 = (\Pi, \Delta, \Sigma)$, with

$$\Pi = \{a \wedge b \rightarrow y\},\ \Delta = \{a, b\} \text{ and } \Sigma = \{a, b\}.$$

Then, the pair

$$(\textit{Warr}, \textit{Block}) = (\{a, b\}, \emptyset)$$

is the only output for $\mathcal{P}_4$ and $\{a \wedge b \rightarrow y\} \cup \{a, b\} \vdash y$ (i.e. $\Pi \cup \textit{Warr} \vdash y$), however $y \notin \textit{Warr}$ since y is not a justifiable conclusion of $\mathcal{P}_4$ (i.e. $y \notin \Sigma$). A different case occurs when $\Pi \cup \textit{Warr} \vdash \varphi$ with $\varphi \in \Sigma$ and there does not exist a consistent proof for φ with respect to the set of warranted conclusions. For instance, consider now the KB $\mathcal{P}_5 = (\Pi, \Delta, \Sigma)$ with

$$\Pi = \{a \wedge b \rightarrow y\},\ \Delta = \{s \rightarrow a, \neg s \rightarrow b, s, \neg s\} \text{ and } \Sigma = \{a, b, y\}.$$

Again,

$$(\textit{Warr}, \textit{Block}) = (\{a, b\}, \emptyset)$$

is the only output for $\mathcal{P}_5$ and thus, although $\Pi \cup \textit{Warr} \vdash y$ and $y \in \Sigma$, $y \notin \textit{Warr}$. The problem here is that there does not exist an argument for y with respect to $\mathcal{P}_5$ since $\{s, \neg s\} \vdash \bot$ and the proof of y should be based on a and b which are respectively based on s and $\neg s$.

Finally, it can be the case that there exists an argument for conclusion φ with $\varphi \in \Sigma$ but, the argument is not valid with respect to the output $(\textit{Warr}, \textit{Block})$. For instance, consider now the KB $\mathcal{P}_6 = (\Pi, \Delta, \Sigma)$ with

$$\Pi = \{a \wedge b \rightarrow y\},\ \Delta = \{s, \neg s, s \rightarrow a, \neg s \rightarrow b, p, \neg p, p \rightarrow y\} \text{ and}$$
$$\Sigma = \{a, b, p, \neg p, y\}.$$

Now,

$$(\textit{Warr}, \textit{Block}) = (\{a, b\}, \{p, \neg p\})$$

is the only output for $\mathcal{P}_6$ and, again, we can see that $\Pi \cup \textit{Warr} \vdash y$ and $y \in \Sigma$, however, $y \notin \textit{Warr}$. The problem here is that although there exists an argument for conclusion y based on p, $\langle \{p, p \rightarrow y\}, y \rangle$, this argument is not valid with respect to $(\textit{Warr}, \textit{Block})$ since p is a blocked conclusion and, as it occurs with program $\mathcal{P}_5$, the proof of y based on a and b is not consistent.

Proposition 6 (Closure) *Let $\mathcal{P} = (\Pi, \Delta, \Sigma)$ be a KB and let the pair $(\textit{Warr}, \textit{Block})$ be an output for $\mathcal{P}$. If $\Pi \cup \textit{Warr} \vdash \varphi$ then $\varphi \in \textit{Warr}$ whenever there exits a valid argument for φ with respect to Warr.*

Proof: Assume $\varphi \in \Sigma$ and $\Pi \cup \mathit{Warr} \vdash \varphi$, but $\Pi \nvdash \varphi$, otherwise it is clear that $\varphi \in \mathit{Warr}$. Further, suppose there exists a valid argument $\langle A, \varphi\rangle$ with respect to *Warr*. By way of contradiction, let us suppose $\varphi \notin Warr$. Then, there would exist a set of valid arguments G such that $\langle A, \varphi\rangle$ is not a subargument of any argument in G and that $G \cup \{\langle A, \varphi\rangle\}$ minimally conflicts with respect to the set $W \subseteq \mathit{Warr}$ of conclusions of all subarguments of arguments in $G \cup \{\langle A, \varphi\rangle\}$. If $G \cup \{\langle A, \varphi\rangle\}$ minimally conflicts with respect to the set W, $\Pi \cup W \cup \{\varphi\} \cup \{\psi \mid \langle B, \psi\rangle \in G\} \vdash \bot$ (Condition (C)), and $\Pi \cup W \cup S \nvdash \bot$, for all set $S \subset \{\varphi\} \cup \{\psi \mid \langle B, \psi\rangle \in G\}$ (Condition (M)). Then, $\Pi \cup W \nvdash \varphi$, and thus, there would exist a set $W' \subseteq \mathit{Warr}$ such that $W \cap W' = \emptyset$ and $\Pi \cup W \cup W' \vdash \varphi$. Now, as for all subarguments of arguments in G theirs conclusions are in W and for all arguments in *Warr* there exist valid arguments, there would exist a conclusion $\phi \in W'$ such that its valid argument $\langle C, \phi\rangle$ and $G \cup \{\langle D, \chi\rangle \mid \chi \in W \cup (W' \backslash \{\phi\})\}$ minimally conflict, and thus, $\phi \notin \mathit{Warr}$. □

Remark that the particular behavior of above KBs $\mathcal{P}_4$, $\mathcal{P}_5$ and $\mathcal{P}_6$ can be avoided with a useful definition of the set of justifiable conclusions Σ. For instance, if we extend the set of justifiable conclusions of $\mathcal{P}_4$ with $\{y\}$ and of $\mathcal{P}_5$ and $\mathcal{P}_6$ with $\{s, \neg s\}$, we get that the pair

$$(\mathit{Warr}, \mathit{Block}) = (\{a, b, y\}, \emptyset)$$

is the only output for the new definition of $\mathcal{P}_4$, the pair

$$(\mathit{Warr}, \mathit{Block}) = (\emptyset, \{s, \sim s\})$$

is the only output for the new definition of $\mathcal{P}_5$ and the pair

$$(\mathit{Warr}, \mathit{Block}) = (\emptyset, \{s, \neg s, p, \neg p\})$$

is the only output for the new definition of $\mathcal{P}_6$.

Given a set of strict and defeasible formulas Π and Δ respectively, we define its set of justifiable conclusions as $Conc(\Pi, \Delta) = \{\varphi \mid \Pi \cup A \vdash \varphi$ for some set $A \subseteq \Delta$ such that $\Pi \cup A \nvdash \bot\}$. Then KBs of the form (Π, Δ, Σ) where the set of formulas over which arguments can be built includes $Conc(\Pi, \Delta)$ enjoy the following proper Closure property.

Corollary 7 (Closure) *Let $\mathcal{P} = (\Pi, \Delta, \Sigma)$ be a KB such that $Conc(\Pi, \Delta) \subseteq \Sigma$. For any output $(\mathit{Warr}, \mathit{Block})$ of $\mathcal{P}$, if $\Pi \cup \mathit{Warr} \vdash \varphi$, then $\varphi \in \mathit{Warr}$.*

Proof: For every $\psi_i \in \mathit{Warr}$, there exists a valid argument $\langle C_i, \psi_i\rangle$. Then, for every $B \subseteq C_i$ such that $\Pi \cup B \vdash \phi$ and $\psi_i \nvdash \phi$, we have that $\langle B, \phi\rangle \sqsubset \langle C_i, \psi_i\rangle$, $\phi \in \Sigma$ and $\phi \in \mathit{Warr}$. It is clear that $\Pi \cup (\cup_i C_i) \vdash \varphi$, and let A be a minimal subset of $\cup_i C_i$ such that $\Pi \cup A \vdash \varphi$. Then, it easily follows that $\langle A, \varphi\rangle$ is a valid argument with respect to *Warr*. □

3 Extending the framework with a preference ordering on arguments

In the previous section, we have considered knowledge bases containing formulas describing knowledge at two epistemic levels, strict and defeasible. A natural extension is to introduce several levels of defeasibility or preference among different pieces of defeasible knowledge.

A *stratified knowledge base* (sKB) is a tuple $\mathcal{P} = (\Pi, \Delta, \preceq, \Sigma)$, such that (Π, Δ, Σ) is a KB (in the sense of the previous section) and $\preceq$ is a total pre-order on $\Pi \cup \Delta$ representing levels of defeasibility: $\varphi \prec \psi$ means that φ is more defeasible than ψ. Actually, since formulas in Π are not defeasible, $\preceq$ is such that all formulas in Π are at the top of the ordering. For the sake of a simpler notation we will often refer in the paper to numerical levels for defeasible clauses and arguments rather than to the pre-ordering $\preceq$, so we will assume a mapping $N : \Pi \cup \Delta \to [0, 1]$ such that $N(\varphi) = 1$ for all $\varphi \in \Pi$ and $N(\varphi) < N(\psi)$ iff $\varphi \prec \psi$. [2] Then we define the *strength of an argument* $\langle A, \varphi\rangle$, written $s(\langle A, \varphi\rangle)$, as follows:

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

Since we are considering several levels of strength among arguments, the intended 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 a sKB $\mathcal{P} = (\Pi, \Delta, \preceq, \Sigma)$, 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 of strength α_i. Then, we safely write *d-Warrr*$(> \alpha_i)$ to denote $\cup_{\beta > \alpha_i}$*d-Warr*(β), and analogously for *Block*$(> \alpha_i)$, assuming *d-Warr*$(> \alpha_1) = \emptyset$ and *Block*$(> \alpha_1) = \emptyset$.

Definition 8 (Output for a sKB) *An output for a sKB $\mathcal{P} = (\Pi, \Delta, \preceq, \Sigma)$ is any pair (Warr, Block), where Warr = s-Warr$\cup$d-Warr with s-Warr $= \{\varphi \mid \Pi \vdash \varphi\} \cap \Sigma$, and d-Warr and Block are required to satisfy the following recursive constraints:* [3]

(1) A d-argument $\langle A, \varphi\rangle$ of strength α_i is called valid *(or not rejected) if it satisfies the following three conditions:*
- *(V1) for all subargument $\langle B, \psi\rangle \sqsubset \langle A, \varphi\rangle$ of strength $\beta \geq \alpha_i$, $\psi \in$ d-Warr(β);*
- *(V2) $\varphi \notin$ d-Warr$(> \alpha_i)$ and $\varphi \notin$ Block$(> \alpha_i)$;*
- *(V3) $\{\varphi, \psi\} \not\vdash \bot$ for all $\psi \in$ Block$(> \alpha_i)$ and $\Pi \cup$ d-Warr$(> \alpha_i) \cup \{\psi \mid$*

[2] Actually, a same pre-order $\preceq$ can be represented by many mappings, but we can take any of them since only the relative ordering is what actually matters.

[3] Remark that if we consider a single defeasibility level α for Δ, *d-Warr*$(> \alpha) = \emptyset$ and *Block*$(> \alpha) = \emptyset$, and therefore the recursive definition of output for a sKB turns equivalent to Definition 3.

$\langle B, \psi\rangle \sqsubset \langle A, \varphi\rangle\} \cup \{\varphi\} \not\vdash \bot$. [4]

(2) For every valid argument $\langle A, \varphi\rangle$ *of strength* α_i *we have that*

- $\varphi \in$ *Block*(α_i) *whenever there exists a set* G *of valid arguments of strength* α_i *such that*
  - *(i)* $\langle A, \varphi\rangle \not\sqsubset \langle C, \chi\rangle$ *for all* $\langle C, \chi\rangle \in G$*, and*
  - *(ii)* $G \cup \{\langle A, \varphi\rangle\}$ *minimally conflicts with respect to the set* $W =$ *d-Warr*$(> \alpha_i) \cup \{\psi \mid \langle B, \psi\rangle \sqsubset \langle D, \gamma\rangle$ *for some* $\langle D, \gamma\rangle \in G \cup \{\langle A, \varphi\rangle\}\}$.
- *otherwise,* $\varphi \in$ *d-Warr*(α_i).

There are two main remarks when considering several levels of strength among arguments. On the one hand a d-argument $\langle A, \varphi\rangle$ of strength α_i is valid whenever (V1) it is based on warranted conclusions; (V2) there does not exist a valid argument for φ with strength greater than α_i; and (V3) φ is consistent with both each blocked argument with strength greater than α_i and the set of already warranted conclusions *d-Warr*$(> \alpha_i) \cup \{\psi \mid \langle B, \psi\rangle \sqsubset \langle A, \varphi\rangle\}$. On the other hand, a valid argument $\langle A, \varphi\rangle$ of strength α_i becomes blocked as soon as it leads to some conflict among arguments with strength α_i with respect to the set of warranted conclusions with higher strengths.

Notice that Conditions (V2) and (V3) define how warranted and blocked conclusions of higher levels are taken into account in lower levels. In particular blocked conclusions play a key role at the propagation mechanism between defeasibility levels. In our approach if a conclusion φ is blocked at level α, then for any lower level than α, not only the conclusion φ is disabled but also every conclusion ψ such that $\{\varphi, \psi\} \vdash \bot$. Intuitively our mechanism tries to ensure that warranted conclusions at lower levels matches the result if conclusions were considered at higher levels. A different approach could be to consider that a blocked conclusion φ only disables the conclusion φ to be warranted for any lower level. In this case it may happen that a conclusion should be rejected at a higher level and warranted at a lower level.

The following examples show how warranted and blocked conclusions of higher levels are taken into account in lower levels.

Example 9 *Consider the KB* $\mathcal{P}_1$ *in the previous section*

$$\Pi = \{a \wedge b \rightarrow \neg p\},\ \Delta = \{a, b, p\} \text{ and } \Sigma = \{a, b, p, \neg p\}.$$

extended with two levels of defeasibility as follows: $\{a, b\} \prec p$*. Assume* α_1 *is the level of* p *and* α_2 *the level of* a *and* b*, obviously with* $1 > \alpha_1 > \alpha_2$*. According to Definition 8,* s*-Warr* $= \emptyset$ *and the argument for* $\langle\{p\}, p\rangle$ *is the only valid argument of strength* α_1*. Then, at level* α_1*, we get d-Warr*$(\alpha_1) = \{p\}$ *and Block*$(\alpha_1) = \emptyset$*. At level* α_2*, we have that* $\langle\{a\}, a\rangle$ *and* $\langle\{b\}, b\rangle$ *are valid arguments for conclusions* a *and* b *respectively. However, since* $\Pi \cup$ *d-Warr*$(\alpha_1) \cup \{a, b\} \vdash \bot$*, the conclusions*

[4] When we consider a single defeasibility level, the notion of argument subsumes condition $\Pi \cup \{\psi \mid \langle B, \psi\rangle \sqsubset \langle A, \varphi\rangle\} \cup \{\varphi\} \not\vdash \bot$.

a and b are blocked, and thus, d-Warr$(\alpha_2) = \emptyset$ and Block$(\alpha_2) = \{a, b\}$. Notice that the argument $\langle\{a, b\}, \neg p\rangle$ for $\neg p$ is not valid since it is based on a and b and $a, b \notin$ d-Warr(α_2).

Example 10 *Consider the KB $\mathcal{P}_2$ of Example 4:*

$$\Pi = \{a \to y, b \wedge c \to \neg y\},\ \Delta = \{a, b, c, \neg c\},\ \text{and}\ \Sigma = \{a, b, c, \neg c, y, \neg y\},$$

extended with three levels of defeasibility as follows: $\neg c \prec c \prec \{a, b\}$. Assume α_1 is the level of a and b, α_2 is the level of c, and α_3 is the level of $\neg c$, with $1 > \alpha_1 > \alpha_2 > \alpha_3$. Then, s-Warr $= \emptyset$ and, at level α_1, we have not only the conclusions a, b and y with valid arguments not generating conflict but also $\langle\{a, b\}, \neg c\rangle$ is a valid argument for $\neg c$ which does not generate conflict. Therefore, d-Warr$(\alpha_1) = \{a, b, y, \neg c\}$ and Block$(\alpha_1) = \emptyset$. At level α_2, we have arguments for c and $\neg y$. Since $\Pi \cup$ d-Warr$(\alpha_1) \cup \{c\} \vdash \bot$, the argument $\langle\{c\}, c\rangle$ is not valid with respect to d-Warr(α_1), and thus, c is a rejected conclusion. Then, as argument for $\neg y$ is based on c, $\neg y$ is also a rejected conclusion. Therefore, d-Warr$(\alpha_2) = \emptyset$ and Block$(\alpha_2) = \emptyset$. Finally, at level α_3 we have the argument $\langle\{\neg c\}, \neg c\rangle$, but since $\neg c$ is already in d-Warr(α_1), we also have d-Warr$(\alpha_3) = \emptyset$ and Block$(\alpha_3) = \emptyset$.

Consider now that the KB $\mathcal{P}_2$ is extended with a new defeasible formula $\neg a$, i.e. $\Pi = \{a \to y, b \wedge c \to \neg y\}$, $\Delta = \{a, b, c, \neg c, \neg a\}$ and $\Sigma = \{a, b, c, \neg c, y, \neg y, \neg a\}$, and two defeasibility levels as follows: $\{\neg a, c\} \prec \{a, b, c, \neg c\}$. Assume α_1 is the highest level and α_2 is the lowest level. Notice that $\neg a$ belongs to the defeasible level α_2 and c belongs to both defeasible levels α_1 and α_2. Again, s-Warr $= \emptyset$ but now, at level α_1 we have that the conclusions a, b, c and $\neg c$ have valid arguments all involved in conflicts, and thus, d-Warr$(\alpha_1) = \emptyset$ and Block$(\alpha_1) = \{a, b, c, \neg c\}$. At level α_2, we have arguments for c and $\neg a$. However, the argument for c is not valid because the conclusion c has been blocked at level α_1 (Condition (V2)), and the argument for $\neg a$ is not valid because the conclusion a has been blocked at level α_1 (Condition (V3)). Therefore, d-Warr$(\alpha_2) = \emptyset$ and Block$(\alpha_2) = \emptyset$.

The following results provide an interesting characterization of the relationship between warranted and blocked conclusions in stratified knowledge bases.

Proposition 11 *Let $\mathcal{P} = (\Pi, \Delta, \preceq, \Sigma)$ be a sKB and let (Warr, Block) be an output for $\mathcal{P}$. Then:*

(1) If $\varphi \in$ d-Warr$(\alpha) \cup$ Block(α), then there exists an argument $\langle A, \varphi\rangle$ of strength α such that for all subargument $\langle B, \psi\rangle \sqsubset \langle A, \varphi\rangle$ of strength β, $\psi \in$ d-Warr(β).
(2) If $\varphi \in$ d-Warr$(\alpha) \cup$ Block(α), then for any argument $\langle A, \varphi\rangle$ of strength β, with $\beta > \alpha$, there exists a subargument $\langle B, \psi\rangle \sqsubset \langle A, \varphi\rangle$ of strength γ and $\psi \notin$ Warr(γ).
(3) If $\varphi \in$ Warr, then $\varphi \notin$ Block and $\psi \notin$ Block, for all ψ such that $\{\varphi, \psi\} \vdash \bot$.
(4) If $\varphi \notin$ Warr $\cup$ Block, then either $\psi \in$ Block with $\{\varphi, \psi\} \vdash \bot$, or for all argument $\langle A, \varphi\rangle$ there exists a subargument $\langle B, \psi\rangle \sqsubset \langle A, \varphi\rangle$ such that $\psi \notin$

Warr or $\Pi \cup$ *d-Warr*$(> \alpha_i) \cup \{\psi \mid \langle B, \psi\rangle \sqsubset \langle A, \varphi\rangle\} \cup \{\varphi\} \nvdash \bot$.

Proof:

(1) Proof follows directly from Condition (V1).

(2) If $\varphi \in$ *d-Warr*$(\alpha) \cup$ *Block*(α), by Condition (V2), $\varphi \notin$ *d-Warr*$(\beta) \cup$ *Block*(β), for all $\beta > \alpha$. Suppose that there exists an argument $\langle A, \varphi\rangle$ of strength β, with $\beta > \alpha$, verifying Condition (V1). Now, as $\varphi \notin$ *d-Warr*$(\gamma) \cup$ *Block*(γ) for all $\gamma > \beta$, Condition (V3) must fail for $\langle A, \varphi\rangle$, and thus, Condition (V3) also must fail for any argument $\langle B, \varphi\rangle$ of strength α. Hence, Condition (V1) fails for any argument $\langle A, \varphi\rangle$ of strength β, with $\beta > \alpha$.

(3) Suppose that $\varphi \in$ *d-Warr*(α) and $\varphi \in$ *Block*(β). By Conditions (V2) and (V3), if $\beta > \alpha$, $\varphi \notin$ *d-Warr*(α) and, if $\beta < \alpha$, $\varphi \notin$ *Block*(β). Then, it must be that $\varphi \in$ *d-Warr*(α) and $\varphi \in$ *Block*(α), and thus, there exits two valid arguments of strength α such that one is involved in a conflict and the other is not. Suppose that $\langle A, \varphi\rangle$ is a valid argument involved in a conflict. Then, there should exist a set G of valid arguments of strength α such that $\langle A, \varphi\rangle$ is not a subargument of arguments in G and $G \cup \{\langle A, \varphi\rangle\}$ minimally conflicts. Hence, every valid argument $\langle B, \varphi\rangle$ is not a subargument of arguments in G, and thus, $\langle B, \varphi\rangle$ is involved in a conflict. Proof that $\psi \notin$ *Block*, for all ψ such that $\{\varphi, \psi\} \vdash \bot$, follows directly from Condition (V3).

(4) If $\varphi \notin$ *Warr* $\cup$ *Block*, then for all argument $\langle A, \varphi\rangle$ either Condition (V1) fails or, otherwise Condition (V3) fails.

□

4 A particular case: recursive P-DeLP

In this section we particularize the recursive warrant semantics for stratified knowledge bases to the case of the P-DeLP programs. As mentioned in Section 1, P-DeLP is a rule-based argumentation system extending the well-known DeLP system [20] in which weights are attached to defeasible rules expressing their belief or preference strength and formalized as necessity degrees. For a detailed description of the P-DeLP argumentation system based on dialectical trees the reader is referred to [6].

Although the original syntax and inference of P-DeLP are a bit different (e.g. the weights are explicit in the formulas and arguments), here we will present them in a way so to adapt them to the framework introduced in the previous sections. We will refer to this particular framework as RP-DeLP (recursive P-DeLP). Hence we define the logic $(\mathcal{L}_R, \vdash_R)$ underlying RP-DeLP as follows.

The language of RP-DeLP 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 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 is *contradictory*, denoted $\Gamma \vdash \bot$, if , for some atom q, $\Gamma \vdash_R q$ and $\Gamma \vdash_R \sim q$.

An RP-DeLP program $\mathcal{P}$ is just a stratified knowledge base $(\Pi, \Delta, \preceq, \Sigma)$ over the logic $(\mathcal{L}_R, \vdash_R)$, where Σ consists of the set of all literals of $\mathcal{L}_R$. As already pointed out, we will assume that $\preceq$ is representable by a mapping $N : \Pi \cup \Delta \rightarrow [0, 1]$ such that $N(\varphi) = 1$ for all $\varphi \in \Pi$ and $N(\varphi) < N(\psi)$ iff $\varphi \prec \psi$, so we will often refer to numerical weights for defeasible clauses and arguments rather than to the pre-ordering $\preceq$. Also, for the sake of a simpler notation we will get rid of Σ of a program specification.

4.1 Arguing with RP-DeLP

In this section we explore the application of the RP-DeLP argumentation framework to the extraction of consistent information from the scope of political debates. Suppose we have two opposite parties of the sphere of Spanish politics: a left-wing party (PSOE) and a right-wing party (PP). We are trying to find what are the positions we can expect both to agree as based on solid arguments considering the facts and rules from the law and their particular beliefs. We prefix the rules with a label (*L:*) so then it is easier to mention them in arguments.

First suppose they are discussing about possible ways to increase the Gross domestic product (GDP) of Spain (target represented by the literal *GDP_UP*). As possible actions, they discuss about:

$G1$: increase the education expenditure

$G2$: increase the infrastructures expenditure

$G3$: decrease taxes for private companies

In the discussion, we have to take into account that the current law only allows two of the previous actions to be executed at most, so executing all three actions is

forbidden by law. So, at the strict level we have that:

$$\begin{aligned}\Pi = \{\ & R1 : \sim G1 \leftarrow G2 \wedge G3, \\ & R2 : \sim G2 \leftarrow G1 \wedge G3, \\ & R3 : \sim G3 \leftarrow G1 \wedge G2\ \}\end{aligned}$$

and actions $\{G1, G2, G3\}$ become defeasible, since they can not be considered as strict facts but, as assumptions and hence, as defeasible information. Moreover, the left-wing party believes that executing $G1$ and $G2$ will increase the GDP, and that the same result will hold if executing $G1$ and $G3$. Finally, the right-wing party believes that executing $G2$ and $G3$ will increase the GDP. So, at the defeasible level we have the following set of facts and rules:

$$\begin{aligned}\Delta = \{\ & G1, G2, G3, \\ & \text{PSOE1} : GDP\_UP \leftarrow G1 \wedge G2, \\ & \text{PSOE2} : GDP\_UP \leftarrow G1 \wedge G3, \\ & \text{PP1} : \quad GDP\_UP \leftarrow G2 \wedge G3\ \}.\end{aligned}$$

In this case *s-Warr* $= \emptyset$ and it happens that $\langle\{G1\}, G1\rangle$, $\langle\{G2\}, G2\rangle$ and $\langle\{G3\}, G3\rangle$ are valid arguments, but each one is blocked by the others two due to the strict knowledge (i.e., $\Pi \cup \{G1\} \not\vdash \bot$, $\Pi \cup \{G2\} \not\vdash \bot$ and $\Pi \cup \{G3\} \not\vdash \bot$ but, $\Pi \cup \{G1, G2, G3\} \vdash \bot$), so we end up with an empty warrant set and with the actions $G1$, $G2$ and $G3$ blocked. As a result all the arguments for the literal *GDP_UP* are rejected and the target *GDP_UP* is not achieved. Suppose now that there is a stronger belief in the possibility of implementing the action $G1$, than in actions $G2$ and $G3$. In this case, we get two defeasibility levels for Δ: α_1 and α_2 with $1 > \alpha_1 > \alpha_2 > 0$. [5] Then, Δ is stratified as follows:

$$\text{level } \alpha_1\text{: } \{G1, \text{PSOE1}, \text{PSOE2}, \text{PP1}\} \qquad \text{level } \alpha_2\text{: } \{G2, G3\}.$$

So in this case $G1$ is the only warranted action at level α_1 (i.e., *d-Warr*$(\alpha_1) = \{G1\}$), but the actions $G2$ and $G3$ become blocked at level α_2 because $\Pi \cup$ *d-Warr*$(\alpha_1) \cup \{G2, G3\} \vdash \bot$. Again, even if now the action $G1$ is warranted, not enough is warranted to have a valid argument for target *GDP_UP* with any of the rules (i.e., all the arguments for *GDP_UP* are based on some blocked argument), and thus, we finally get:

$$\textit{Warr} = \{G1\} \text{ and } \textit{Block} = \{G2, G3\}.$$

[5] As it is assumed in many scenarios of non-monotonic reasoning or belief revision, defeasibility levels are specified by the knowledge engineer according to their (subjective) priority or preference: the higher is the priority, the higher is the level.

Suppose that now there is a stronger belief in the possibility of implementing the actions $G1$ and $G2$, than the belief for action $G3$. In this case, the defeasible knowledge Δ becomes:

$$\text{level } \alpha_1\text{: } \{G1, G2, \text{PSOE1}, \text{PSOE2}, \text{PP1}\} \quad \text{level } \alpha_2\text{: } \{G3\}.$$

So in this case, at level α_1, $G1$ and $G2$ are warranted actions and consequently so are the literals *GDP_UP* and $\sim G3$ because $\langle \{G1, G2, \text{PSOE1}\}, \textit{GDP\_UP} \rangle$ and $\langle \{G1, G2\}, \sim G3 \rangle$ are valid arguments at level α_1, and thus, the action $G3$ becomes an invalid (rejected) position at level α_2. Therefore, in this case we have the following output:

$$\textit{Warr} = \{G1, G2, GDP\_UP, \sim G3\} \text{ and } \textit{Block} = \emptyset.$$

Suppose now that the right-wing party hardens its speech and adds a new belief to the discussion: "increasing the education expenditure will cause the GDP to not increase". This new information is represented by the defeasible rule

$$\text{PP2}: \ \sim \textit{GDP\_UP} \leftarrow G1$$

and is incorporated with the same strength than the previous rules into the debate. So, the defeasible knowledge Δ becomes:

$$\begin{aligned} &\text{level } \alpha_1\text{: } \{G1, G2, \text{PSOE1}, \text{PSOE2}, \text{PP1}, \text{PP2}\} \\ &\text{level } \alpha_2\text{: } \{G3\}. \end{aligned}$$

In this case, as before, we warrant $G1$ and $G2$, and thus, we have valid arguments for *GDP_UP* and $\sim$*GDP_UP* both of strength α_1. So at level α_1, *GDP_UP* and $\sim$*GDP_UP* become blocked. Finally, as before, $\sim G3$ is warranted at level α_1 and the action $G3$ is rejected at level α_2. Therefore, in this case we have the following output:

$$\begin{aligned} &\textit{Warr} = \{G1, G2, \sim G3\} \quad \text{and} \\ &\textit{Block} = \{\textit{GDP\_UP}, \sim \textit{GDP\_UP}\}. \end{aligned}$$

Remark that if we instead consider that the PP2 rule is weaker than the other rules, the defeasible knowledge Δ becomes:

$$\begin{aligned} &\text{level } \alpha_1\text{: } \{G1, G2, \text{PSOE1}, \text{PSOE2}, \text{PP1}\} \\ &\text{level } \alpha_2\text{: } \{G3, \text{PP2}\}, \end{aligned}$$

and thus, the argument $\langle \{G1, G2, \text{PSOE1}\}, \textit{GDP\_UP} \rangle$ is warranted at level α_1 and the argument $\langle \{G1, \text{PP2}\}, \sim \textit{GDP\_UP} \rangle$ is rejected at level α_2 because it is inconsistent with the previous warrant set (i.e., $\Pi \cup \textit{d-Warr}(\alpha_1) \cup \{\sim \textit{GDP\_UP}\} \vdash \bot$).

So, in this case we have the following output:

$$\textit{Warr} = \{G1, G2, \textit{GDP\_UP}, \sim G3\} \text{ and } \textit{Block} = \emptyset.$$

4.2 RP-DeLP programs with multiple outputs

As we have mentioned in Section 2, in some cases the output $(\textit{Warr}, \textit{Block})$ for a knowledge base in general, and for an RP-DeLP program in particular, is not unique, due to some recursive definitions of conflict that emerge when considering inference (support) and conflict relations among arguments. The following example shows a recursion case from the scope of political debates.

This time the law changes and what we have now is a relaxation of the strict rules, and thus, $\Pi = \emptyset$ and the rules $R1$, $R2$ and $R3$ become questionable (defeasible). Thus, we can consider a defeasibility level α_1 for rules $R1$, $R2$ and $R3$. Remember from the previous section that

$G1$: increase the education expenditure	$R1 : \sim G1 \leftarrow G2 \wedge G3$
$G2$: increase the infrastructures expenditure	$R2 : \sim G2 \leftarrow G1 \wedge G3$
$G3$: decrease taxes for private companies	$R3 : \sim G3 \leftarrow G1 \wedge G2$

As in the first example suppose that the left-wing party believes that executing $G1$ and $G2$ will increase the GDP (PSOE1 rule), and that the same result will hold if executing $G1$ and $G3$ (PSOE2 rule). The right-wing party believes that executing $G2$ and $G3$ will increase the GDP (PP1 rule). So we can consider a second defeasibility level α_2 with $1 > \alpha_1 > \alpha_2 > 0$, and then we stratify the defeasible knowledge as follows:

$$\text{level } \alpha_1\text{: } \{R1, R2, R3\}$$
$$\text{level } \alpha_2\text{: } \{G1, G2, G3, \text{PSOE1}, \text{PSOE2}, \text{PP1}\}.$$

In this case, *s-Warr* $=$ *d-Warr*$(\alpha_1) =$ *Block*$(\alpha_1) = \emptyset$ and $\langle\{G1\}, G1\rangle$, $\langle\{G2\}, G2\rangle$ and $\langle\{G3\}, G3\rangle$ are valid arguments of strength α_2, but each one can be warranted if and only if one of the other two is blocked. Hence, we have three possible outputs: $(\textit{Warr}_1, \textit{Block}_1)$, $(\textit{Warr}_2, \textit{Block}_2)$ and $(\textit{Warr}_3, \textit{Block}_3)$ where

$$\begin{array}{ll} \textit{Warr}_1 = \{G1, G2, \textit{GDP\_UP}\}, & \textit{Block}_1 = \{G3, \sim G3\}, \\ \textit{Warr}_2 = \{G1, G3, \textit{GDP\_UP}\}, & \textit{Block}_2 = \{G2, \sim G2\}, \\ \textit{Warr}_3 = \{G2, G3, \textit{GDP\_UP}\}, & \textit{Block}_3 = \{G1, \sim G1\}. \end{array}$$

In the rest of this section we formalize recursive definitions of conflict in RP-DeLP by means of what we call *Warrant Dependency Graph*. In [22] a similar graph struc-

ture, called inference-graph, was defined to represent inference (support) and defeat relations among arguments allowing to detect recursive defeat relations when considering recursive semantics for defeasible reasoning. The main difference between both approaches is that in our case we handle collective conflicts among arguments in order to preserve indirect consistency and closure among warranted conclusions with respect to the strict knowledge.

The characterization of the *Unique Output Property* for an RP-DeLP program $\mathcal{P} = (\Pi, \Delta, \preceq)$is done level-wise, starting from the highest level and iteratively going down from one level to next level below. For every level α it consists in checking whether for some literal L, the warranty of L recursively depends on itself based on the topology of a warrant dependency graph for a set of valid arguments of strength α and a set of what we call *Almost Valid Arguments* of strength α. A valid argument captures the idea of a non-rejected d-argument (i.e. a warranted or blocked d-argument, but not rejected) while an almost valid argument captures the idea of a d-argument whose rejection is conditional to the warranty of some valid argument.

Notation: In the rest of the paper, given an RP-DeLP program $\mathcal{P} = (\Pi, \Delta, \preceq)$ with defeasibility levels $1 > \alpha_1 > \ldots > \alpha_m > 0$, if W and B denote sets of warranted and blocked conclusions, respectively, we will write $W(\alpha_i)$ and $B(\alpha_i)$ to denote the sets of the warranted and blocked conclusions of strength α_i from W and B, respectively. Then, we will also write $W(> \alpha_i)$ to denote $\cup_{\beta > \alpha_i} W(\beta)$ and $W(\geq \alpha_i)$ to denote $\cup_{\beta \geq \alpha_i} W(\beta)$, and analogously for $B(> \alpha_i)$ and $B(\geq \alpha_i)$, assuming $W(> \alpha_1)$ is the set of the warranted conclusions of W from the strict knowledge represented as $W(1)$, and $B(> \alpha_1) = \emptyset$.

Definition 12 (Almost valid argument) *Let $\mathcal{P} = (\Pi, \Delta, \preceq)$ be an RP-DeLP program, let W and B be two sets of warranted and blocked conclusions, respectively, and let $\mathcal{A}$ be a set of valid d-arguments of strength α.*[6] *An argument $\langle F, P\rangle$ of strength α is* almost valid *w.r.t. $\mathcal{A}$ if it satisfies the following six conditions:*

(AV1) for all subargument $\langle C, R\rangle \sqsubset \langle F, P\rangle$ of strength $\beta > \alpha$, $R \in W(\beta)$;
(AV2) $P \notin W(> \alpha)$ and $P \notin B(> \alpha)$;
(AV3) $\sim P \notin B(> \alpha)$ and $\Pi \cup W(> \alpha) \cup \{R \mid \langle C, R\rangle \sqsubset \langle F, P\rangle\} \cup \{P\} \not\vdash \bot$;
(AV4) there does not exist a valid d-argument for conclusion P of strength α;
(AV5) for all subargument $\langle C, R\rangle \sqsubset \langle F, P\rangle$ of strength α such that $R \notin W(\alpha)$, it holds that $\langle C, R\rangle \in \mathcal{A}$, otherwise R and $\sim R \notin B(\geq \alpha)$; and
(AV6) there exists at least an argument $\langle C, R\rangle \in \mathcal{A}$ such that $\langle C, R\rangle \sqsubset \langle F, P\rangle$.

Intuitively, an almost valid argument captures the idea of an argument based on

[6] Remember that a d-argument $\langle A, Q\rangle$ of strength α is *valid* with respect to (W, B) if it satisfies Conditions (V1)-(V3); i.e. (V1) for all subargument $\langle C, R\rangle \sqsubset \langle A, Q\rangle$ of strength $\beta \geq \alpha$, $R \in W(\beta)$; (V2) $Q \notin W(> \alpha) \cup B(> \alpha)$; (V3) $\sim Q \notin B(> \alpha)$ and $\Pi \cup W(> \alpha) \cup \{R \mid \langle C, R\rangle \sqsubset \langle A, Q\rangle\} \cup \{Q\} \not\vdash \bot$.

valid arguments and which status is warranted (not rejected) whenever these sub-arguments are warranted, and rejected, otherwise. In particular, Condition (AV1) corresponds to a smoothed version of Condition (V1). Conditions (AV2) and (AV3) are equivalent to Conditions (V2) and (V3), respectively. Condition (V4) ensures that there does not exist a valid argument for the literal, and Conditions (AV5) and (AV6) ensure that the status of an almost valid argument depends on the status of at least one valid argument.

For instance, in the above example, $\langle\{G1\}, G1\rangle$, $\langle\{G2\}, G2\rangle$ and $\langle\{G3\}, G3\rangle$ are valid arguments, while

$$\langle\{G2, G3, R1\}, \sim G1\rangle, \langle\{G1, G3, R2\}, \sim G2\rangle \text{ and } \langle\{G1, G2, R3\}, \sim G3\rangle$$

are almost valid arguments based on them.

At this point we are ready to define the warrant dependency graph for a set of valid arguments and a set of almost valid arguments.

Definition 13 (Warrant dependency graph) *Let $\mathcal{P} = (\Pi, \Delta, \preceq)$ be an RP-DeLP program and let W and B be two sets of warranted and blocked conclusions, respectively. Moreover, let $\mathcal{A}_1 = \langle A_1, Q_1\rangle, \ldots, \mathcal{A}_k = \langle A_k, Q_k\rangle$ be valid d-arguments of strength α, and let $\mathcal{F}_1 = \langle F_1, P_1\rangle, \ldots, \mathcal{F}_n = \langle F_n, P_n\rangle$ be d-arguments of strength α that are almost valid with respect to $\{\mathcal{A}_1, \ldots, \mathcal{A}_k\}$. The warrant dependency graph (V, E) for $\{\mathcal{A}_1, \ldots, \mathcal{A}_k\}$ and $\{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ is defined as follows:*

(1) For every literal $L \in \{Q_1, \ldots, Q_k\} \cup \{P_1, \ldots, P_n\}$, the set of vertices V includes one vertex v_L.

(2) For every pair of literals $(L_1, L_2) \in \{Q_1, \ldots, Q_k\} \times \{P_1, \ldots, P_n\}$ such that the argument of L_1 is a subargument of the argument of L_2, the set of directed edges E includes one edge (v_{L_1}, v_{L_2}). [7]

(3) For every pair of literals $(L_1, L_2) \in \{P_1, \ldots, P_n\} \times \{Q_1, \ldots, Q_k\}$ such that $L_1 = \sim L_2$, the set of directed edges E includes one edge (v_{L_1}, v_{L_2}). [8]

(4) For every strict rule $R \leftarrow R_1 \wedge \ldots \wedge R_p \in \Pi$ such that $\{\sim R, R_1, \ldots, R_p\} \subseteq W(\geq \alpha) \cup \{Q_1, \ldots, Q_k\} \cup \{P_1, \ldots, P_n\}$, the set of directed edges E includes one edge (v_{L_1}, v_{L_2}) for every pair of literals $(L_1, L_2) \in \{P_1, \ldots, P_n\} \times \{Q_1, \ldots, Q_k\}$ such that the argument of L_2 is not a subargument of the argument of L_1, $L_1 \in \{\sim R, R_1, \ldots, R_p\}$ and, either $L_2 \in \{\sim R, R_1, \ldots, R_p\}$ or, L_2 is a subargument of the argument of L_3, for some $L_3 \in \{P_1, \ldots, P_n\}$ such that $L_3 \in \{\sim R, R_1, \ldots, R_p\}$. [9]

[7] The directed edge (v_{L_1}, v_{L_2}) represents an inference (subargument) relation from a valid argument to an almost valid argument.

[8] The directed edge (v_{L_1}, v_{L_2}) represents a direct conflict, inconsistency due to defeasible rules, between an almost valid argument and a valid argument.

[9] The directed edge (v_{L_1}, v_{L_2}) represents an indirect conflict, inconsistency due to strict rules, between an almost valid argument and a valid argument.

(5) Elements of V and E are only obtained by applying the above construction rules.

Intuitively, the warrant dependency graph for $\{\mathcal{A}_1, \ldots, \mathcal{A}_k\}$ and $\{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ represents conflict and support relationships among these sets of arguments of strength α with respect to the set $W(\geq \alpha)$ of warranted conclusions of equal or higher strength.

Figure 1 shows the warrant dependence graph for the above example. Remember that $\langle\{G1\}, G1\rangle$, $\langle\{G2\}, G2\rangle$ and $\langle\{G3\}, G3\rangle$ were valid arguments, while $\langle\{G2, G3, R1\}, \sim G1\rangle$, $\langle\{G1, G3, R2\}, \sim G2\rangle$ and $\langle\{G1, G2, R3\}, \sim G3\rangle$ were almost valid arguments based on them. Conflict and support relationships among these arguments are represented as dashed and solid arrows, respectively. The graph contains many cycles. For instance, the set of edges

$$\{(\sim G1, G1), (G1, \sim G2), (\sim G2, G2), (G2, \sim G1)\}$$

expresses that (1) the warranty of $G1$ depends on a (possible) conflict with $\sim G1$ (direct conflict between $G1$ and $\sim G1$ if $\sim G1$ was valid); (2) the support of $\sim G2$ depends on $G1$ (i.e. the validity of $\sim G2$ depends on the warranty of $G1$); (3) the warranty of $G2$ depends on a (possible) conflict with $\sim G2$ (direct conflict between $G2$ and $\sim G2$ if $\sim G2$ was valid); and (4) the support of $\sim G1$ depends on $G2$ (i.e. the validity of $\sim G1$ depends on the warranty of $G2$).

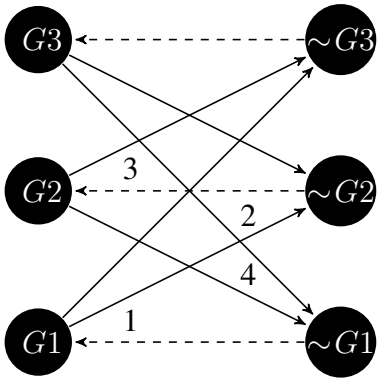

Fig. 1. Recursion case from the scope of political debates.

The following example shows a recursive definition of conflict which arises from the strict knowledge.

Example 14 *Consider the RP-DeLP program $\mathcal{P}_{R1} = (\Pi, \Delta, \preceq)$ with*

$$\Pi = \{y, \sim y \leftarrow p \wedge r, \sim y \leftarrow q \wedge s\} \text{ and } \Delta = \{p, q, r \leftarrow q, s \leftarrow p\},$$

and a single defeasibility level α for Δ.

Consider the sets $W(1) = \{Q \mid \Pi \vdash_R Q\} = \{y\}$, $B(1) = \emptyset$, $W(\alpha) = \emptyset$ and $B(\alpha) = \emptyset$. Now consider arguments for conclusions p and q; i.e.

$$\mathcal{A}_1 = \langle \{p\}, p \rangle \text{ and } \mathcal{A}_2 = \langle \{q\}, q \rangle.$$

Finally, consider arguments for conclusions r and s; i.e.

$$\mathcal{F}_1 = \langle \{q, r \leftarrow q\}, r \rangle \text{ and } \mathcal{F}_2 = \langle \{p, s \leftarrow p\}, s \rangle.$$

Obviously, $\mathcal{A}_1$ and $\mathcal{A}_2$ are valid arguments with respect to $(W(\geq \alpha), B(> \alpha))$ and $\mathcal{F}_1$ and $\mathcal{F}_2$ are almost valid arguments with respect to $\{\mathcal{A}_1, \mathcal{A}_2\}$ and $(W(\geq \alpha), B(> \alpha))$. Figure 2 shows the warrant dependency graph for $\{\mathcal{A}_1, \mathcal{A}_2\}$ and $\{\mathcal{F}_1, \mathcal{F}_2\}$. The cycle of the graph expresses that (1) the warranty of p depends on a (possible) conflict with r; (2) the support of r depends on q;(3) the warranty of q depends on a (possible) conflict with s; and (4) the support of s depends on p.

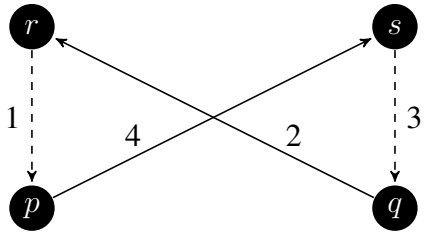

Fig. 2. Warrant dependency graph for $\mathcal{P}_{R1}$.

Proposition 15 (RP-DeLP program with unique output) *Let $\mathcal{P} = (\Pi, \Delta, \preceq)$ be an RP-DeLP program and let $(Warr, Block)$ be an output for $\mathcal{P}$. $(Warr, Block)$ is the unique output for $\mathcal{P}$ iff, for all defeasibility level α and literal $L \in$ d-Warr(α), there is no cycle in the warrant dependency graph for the set of arguments $\mathcal{A}$ and the set of arguments $\mathcal{F}$ where*

- *$\mathcal{A}$ is the set of all d-arguments of strength α that are valid w.r.t $($Warr$(\geq \alpha) \backslash \{L\},$ Block$(> \alpha))$, and*
- *$\mathcal{F}$ is the set of all d-arguments of strength α that are almost valid with respect to $\mathcal{A}$ and $($Warr$(\geq \alpha) \backslash \{L\},$ Block$(> \alpha))$.*

Proof: Suppose that $(Warr, Block)$ is the unique output for $\mathcal{P}$ and there is a cycle in the graph for some literal $L \in$ d-Warr(α). On the one hand, if $(Warr, Block)$ is the unique output for $\mathcal{P}$, there does not exist a pair $(Warr', Block')$ that satisfies Definition 8 and $Warr' \neq Warr$ or $Block' \neq Block$, and thus, every literal is either warranted, or blocked, or rejected. On the other hand, given $L \in$ d-Warr(α), $\mathcal{A}$ is the set of arguments of strength α which are valid with respect to $($Warr$(\geq \alpha) \backslash \{L\},$ Block$(> \alpha))$, hence, arguments in $\mathcal{A}$ do not depend on L and there is an argument for L in $\mathcal{A}$. Similarly, $\mathcal{F}$ is the set of arguments of strength α that are almost valid with respect to $\mathcal{A}$ and $($Warr$(\geq \alpha) \backslash \{L\},$ Block$(> \alpha))$, hence, the support of arguments in $\mathcal{F}$ depends on L or some argument in $\mathcal{A}$. Now, according to Definition 13, if there is a cycle in the warrant dependency graph, it must be that the warranty of the argument for L depends on the validity of at least an argument $\langle F, P \rangle \in \mathcal{F}$, which depends on the warranty of some argument $\langle A, L' \rangle \in \mathcal{A}$ with $L \neq L'$, which depends on the validity of at least an argument $\langle F', P' \rangle \in \mathcal{F}$

with $P' \neq P$, which in turn depends on the warranty of L. Then, according to Definition 8, either L is warranted and L' is blocked, or L' is warranted and L is blocked, and therefore, there exists at least two different outputs for $\mathcal{P}$. Finally, if for all defeasibility level α and literal $L \in$ *d-Warr*(α), there is no cycle in the the warrant dependency graph with respect to $(\textit{Warr}(\geq \alpha)\backslash\{L\}, \textit{Block}(> \alpha))$, there exists a unique warranty evaluation order between arguments, and thus, there exists a unique output for $\mathcal{P}$. □

For instance, consider the RP-DeLP program $\mathcal{P}_{R1}$ from Example 14. According to Definition 8, *Output* $= (\textit{Warr}, \textit{Block})$ with *Warr* $=$ *s-Warr* $\cup$ *d-Warr*(α), *s-Warr* $= \{y\}$, *d-Warr*$(\alpha) = \{p\}$ and *Block* $=$ *Block*$(\alpha) = \{q, \sim s\}$, is an output for $\mathcal{P}_{R1}$. Then, as $p \in$ *d-Warr*(α), defining $W =$ *Warr*$(\geq \alpha)\backslash\{p\} =$ *s-Warr* $= \{y\}$ and $B =$ *Block*$(> \alpha) = \emptyset$, we get that $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2\}$ is the set of all valid arguments with respect to W and B, and $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2\}$ is the set of all almost valid arguments with respect to $\mathcal{A}$. Moreover, the warrant dependency graph for $\mathcal{A}$ and $\mathcal{F}$ contains a cycle (see Figure 2) proving that the output for $\mathcal{P}_{R1}$ is not unique. Indeed, notice that *Output*$' = (\textit{Warr}', \textit{Block}')$ with *Warr*$'$ $=$ *s-Warr* $\cup$ *d-Warr*$'(\alpha)$, *d-Warr*$'(\alpha) = \{q\}$ and *Block*$' =$ *Block*$'(\alpha) = \{p, r\}$, is also an output for $\mathcal{P}_{R1}$. Moreover, as *Warr*$'(\geq \alpha)\backslash\{q\} =$ *Warr*$(\geq \alpha)\backslash\{p\} =$ *s-Warr* $= \{y\}$ and *Block*$'(> \alpha) =$ *Block*$(> \alpha) = \emptyset$, the warrant dependency graph for $q \in$ *d-Warr*$'(\alpha)$ also corresponds to the graph in Figure 2.

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 the warrant dependency graph for all sets of valid and almost valid arguments with respect to the sets of already warranted and blocked conclusions. In [2] we designed 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 rest of the paper we tackle the problem of which output one should consider for an RP-DeLP program with multiple outputs. To this end we define the *maximal ideal output* of an RP-DeLP program as the set of conclusions which are ultimately warranted and we design an algorithm for computing them in polynomial space and with an upper bound on complexity equal to P^{NP}.

5 Maximal ideal output

In the previous section we have characterized the unique output property for the particular framework of RP-DeLP programs. Now in this section we are interested in the problem of deciding the set of conclusions that can be ultimately warranted in RP-DeLP programs with multiple outputs. The usual skeptical approach would

be to adopt he intersection of all possible outputs. However, in addition to the computational limitation, as stated in [22], adopting the intersection of all outputs may lead to an inconsistent output (in the sense of violating the base of the underlying recursive warrant semantics) in case some particular recursive situation among literals of a program occurs. Intuitively, for a conclusion, to be in the intersection does not guarantee the existence of an argument for it that is recursively based on ultimately warranted conclusions.

For instance, consider the following situation involving three conclusions P, Q, and T, where P can be warranted whenever Q is blocked, and vice-versa. Moreover, suppose that T can be warranted when either P or Q are warranted. Then, according to the warrant recursive semantics, we would get two different outputs: one where P and T are warranted and Q is blocked, and the other one where Q and T are warranted and P is blocked. Then, adopting the intersection of both outputs we would get that T would be ultimately warranted, however T should be in fact rejected since neither P nor Q are ultimately warranted conclusions.

According to this example, one could take then as the set of ultimately warranted conclusions of RP-DeLP programs those conclusions in the intersection of all outputs which are recursively based on ultimately warranted conclusions. However, as in RP-DeLP there are levels of defeasibility, this approach could lead to an incomplete solution since we are interested in determining the biggest set of ultimately warranted conclusions with maximum strength.

For instance consider the above example extended with two defeasibility levels as follows. Suppose that P can be warranted with strength α whenever Q is blocked, and vice-versa. Moreover, suppose that T can be warranted with strength α whenever P is warranted at least with strength α and that T can be warranted with strength β, with $\beta < \alpha$, independently of the status of conclusions P and Q. Then, again we get two different outputs: one output warrants conclusions P and T with strength α and blocks conclusion Q, and the other one warrants conclusions Q and T with strengths α and β, respectively, and blocks P. Now, adopting conclusions of the intersection which are recursively based on ultimately warranted conclusions, we get that conclusion T is finally rejected, since conclusion T is warranted with a different argument and strength in each output. However, as we are interested in determining the biggest set of warranted conclusions with maximum strength, it seems quite reasonable to reject T at level α but to warrant it at level β.

Therefore, the *maximal ideal output* for an RP-DeLP program $\mathcal{P} = (\Pi, \Delta, \preceq)$ is a pair $(\mathit{Warr}, \mathit{Block})$ of warranted and blocked conclusions, respectively, with a maximum strength level such that the arguments of all of them are recursively based on warranted conclusions but, while warranted conclusions do not generate any conflict with the set of already warranted conclusions and any circular definition of warranty characterized by a warrant dependency graph, blocked conclusions do. In fact, in a different context, this idea corresponds to the *maximal ideal exten-*

sion defined by Dung, Mancarella and Toni [16,17] as an alternative skeptical basis for defining collections of justified arguments in the abstract argumentation frameworks promoted by Dung [15] and Bondarenko *et al.* [10].

Definition 16 (Maximal ideal output) *The* maximal ideal output *for an RP-DeLP program $\mathcal{P} = (\Pi, \Delta, \preceq)$ is a pair (Warr, Block), where Warr = s-Warr ∪ d-Warr with s-Warr = $\{Q \mid \Pi \vdash_R Q\}$, such that d-Warr and Block are required to satisfy the following recursive constraint: for every valid argument $\langle A, Q\rangle$ of strength α it holds that:*

- *$Q \in$ Block(α) whenever one of the two following conditions holds:*
  - *(a) There exists a set G of valid arguments of strength α with $\langle A, Q\rangle \notin G$ such that the two following conditions hold:*
    - *(i) $\langle A, Q\rangle \not\sqsubset \langle C, R\rangle$, for all $\langle C, R\rangle \in G$, and*
    - *(ii) $G \cup \{\langle A, Q\rangle\}$ generates a conflict with respect to d-Warr$(> \alpha) \cup \{P \mid$ there exists $\langle B, P\rangle \sqsubset \langle C, R\rangle$ for some $\langle C, R\rangle \in G \cup \{\langle A, Q\rangle\}\}$.*
  - *(b) There exists a set $\mathcal{H}$ of valid arguments of strength α such that the three following conditions hold:*
    - *(i) $\langle A, Q\rangle \not\sqsubset \langle C, R\rangle$, for all $\langle C, R\rangle \in \mathcal{H}$.*
    - *(ii) There exists a set of arguments $\mathcal{F}$ of strength α that are almost valid with respect to $\mathcal{H} \cup \langle A, Q\rangle$ and such that there is a cycle in the warrant dependence graph (V, E) for $\mathcal{H} \cup \langle A, Q\rangle$ and $\mathcal{F}$, and all argument $\langle C, R\rangle \in \mathcal{H}$ is such that R is either a vertex of the cycle or $\langle C, R\rangle$ does not satisfy Condition (a).*
    - *(iii) For some vertex $v \in V$ of the cycle either v is the vertex of conclusion Q or v is the vertex of some other conclusion in $\mathcal{H}$ and there exists a path from v to the the vertex of conclusion Q.*
- *Otherwise, $Q \in$ d-Warr(α).*

The intuition underlying the maximal ideal output definition is as follows. The conclusion of every valid (not rejected) argument $\langle A, Q\rangle$ of strength α is either warranted or blocked. Then, it is eventually blocked if either (a) it is involved in some conflict with respect to *d-Warr*$(> \alpha)$ and a set G of valid arguments whose supports do not depend on $\langle A, Q\rangle$, or (b) the warranty of $\langle A, Q\rangle$ depends on some circular definition of conflict between valid and almost valid arguments; otherwise, it is warranted. Condition (b) checks whether the warranty of $\langle A, Q\rangle$ depends on some circular definition of conflict between a set of valid arguments $\mathcal{H}$ whose supports do not depend on $\langle A, Q\rangle$ and a set of almost valid arguments $\mathcal{F}$ whose supports depend on some argument in $\mathcal{H} \cup \langle A, Q\rangle$. In fact, the idea here is that if the warranty of $\langle A, Q\rangle$ depends on some circular definition of conflict between the arguments of $\mathcal{H}$ and $\mathcal{F}$, one could consider two different outputs (status) for conclusion Q: one with Q warranted and another one with Q blocked. Therefore, conclusion Q is blocked for the maximal ideal output. In general, the arguments of $\mathcal{H}$ and $\mathcal{F}$ involved in a cycle are respectively warranted and rejected for the maximal ideal output.

For instance, consider again the recursion case from the scope of political debates developed in the last section. Figure 1 showed the warrant dependency graph for the set of valid arguments $\mathcal{H} = \{\langle\{G1\}, G1\rangle, \langle\{G2\}, G2\rangle, \langle\{G3\}, G3\rangle\}$ and the set of almost valid arguments

$$\mathcal{F} = \{\langle\{G2, G3, R1\}, \sim G1\rangle, \langle\{G1, G3, R2\}, \sim G2\rangle, \langle\{G1, G2, R3\}, \sim G3\rangle\}.$$

Then, as for every valid argument in $\mathcal{H}$ there is a a cycle, the maximal ideal output for the RP-DeLP program is *Warr* $= \emptyset$ and *Block* $= \{G1, G2, G3\}$.

Consider now the RP-DeLP program $\mathcal{P}_{R1}$ of the last section. Figure 2 showed the warrant dependency graph for the set of valid arguments $\mathcal{H} = \{\langle\{p\}, p\rangle, \langle\{q\}, q\rangle\}$ and the set of almost valid arguments $\mathcal{F} = \{\langle\{q, r \leftarrow q\}, r\rangle, \langle\{p, s \leftarrow p\}, s\rangle\}$. Again, as for every valid argument there is a cycle, the maximal ideal output for $\mathcal{P}_{R2}$ is *Warr* $= \{y\}$ and *Block* $= \{p, q\}$.

Finally, consider that we extend $\mathcal{P}_{R1}$ with the following defeasible rules $\{t, t \leftarrow p, t \leftarrow q\}$ and two defeasibility levels; i.e. consider the RP-DeLP program $\mathcal{P}_{R2} = (\Pi, \Delta, \preceq)$ with

$$\Pi = \{y, \sim y \leftarrow p \wedge r, \sim y \leftarrow q \wedge s\},$$
$$\Delta = \{p, q, t, r \leftarrow q, s \leftarrow p, t \leftarrow p, t \leftarrow q\} \text{ and}$$

two defeasibility levels for Δ as follows: $\{t\} \prec \{p, q, r \leftarrow q, s \leftarrow p, t \leftarrow p, t \leftarrow q\}$. Assume α_1 is the level of $\{p, q, r \leftarrow q, s \leftarrow p, t \leftarrow p, t \leftarrow q\}$ and α_2 is the level of $\{t\}$, with $1 > \alpha_1 > \alpha_2 > 0$. Obviously, *s-Warr* $= \{y\}$ and, at level α_1, $\mathcal{H}_1 = \langle\{p\}, p\rangle$ and $\mathcal{H}_2 = \langle\{q\}, q\rangle$ are valid arguments. Moreover, $\mathcal{F}_1 = \langle\{q, r \leftarrow q\}, r\rangle$, $\mathcal{F}_2 = \langle\{p, s \leftarrow p\}, s\rangle$, $\mathcal{F}_3 = \langle\{q, t \leftarrow q\}, t\rangle$ and $\mathcal{F}_4 = \langle\{p, t \leftarrow p\}, t\rangle$ are almost valid arguments with respect to $\{\mathcal{H}_1, \mathcal{H}_2\}$. Figure 3 shows the warrant dependency graph for $\{\mathcal{H}_1, \mathcal{H}_2\}$ and $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4\}$. As for every valid argument there is a a cycle, conclusions p and q are blocked, and conclusions r and s are rejected for the maximal ideal output. Remark that conclusion t is also rejected at level α_1 since the support of $\mathcal{F}_3$ depends on p, the support of $\mathcal{F}_4$ depends on q, and p and q are blocked. Therefore, *s-Warr* $= \{y\}$, *d-Warr*$(\alpha_1) = \emptyset$ and *Block*$(\alpha_1) = \{p, q\}$. Finally, at level α_2, $\langle\{t\}, t\rangle$ is the unique valid argument and therefore conclusion t is warranted. Hence, *d-Warr*$(\alpha_2) = \{t\}$ and *Block*$(\alpha_2) = \emptyset$, and thus, *Warr* $= \{y, t\}$ and *Block* $= \{p, q\}$.

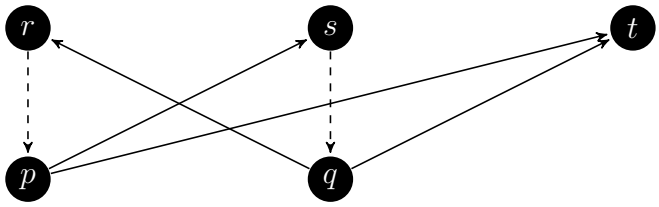

Fig. 3. Warrant dependency graph for $\mathcal{P}_{R2}$ at level α_1.

Next proposition states that the maximal ideal output for an RP-DeLP program is unique.

Proposition 17 (Unicity of the maximal ideal output) *Let $\mathcal{P} = (\Pi, \Delta, \preceq)$ be an RP-DeLP program. The pair $(Warr, Block)$ of warranted and blocked conclusions that satisfies the maximal ideal output characterization for $\mathcal{P}$ is unique.*

Proof: Suppose that $(Warr, Block)$ and $(Warr', Block')$ are pairs of warranted and blocked conclusions that satisfy the maximal ideal output characterization for $\mathcal{P}$. Obviously, *s-Warr* $=$ *s-Warr'*. Suppose that for some α, *d-Warr*$(\alpha) \neq$ *d-Warr'*(α) and *d-Warr*$(\beta) =$ *d-Warr'*(β), for all $\beta > \alpha$. As *d-Warr*$(\alpha) \neq$ *d-Warr'*(α), suppose that $\langle A, Q\rangle$ of strength α is valid with respect to $(Warr, Block)$ and $(Warr', Block')$ but $Q \notin$ *d-Warr*(α) and $Q \in$ *d-Warr'*(α). Then, $Q \in$ *Block*(α) and $\langle A, Q\rangle$ is (a) either involved in a conflict with respect to *d-Warr*$(> \alpha)$ and a set G of valid arguments of strength α which supports do not depend on $\langle A, Q\rangle$, or (b) the warranty of $\langle A, Q\rangle$ depends on a circular definition of conflict between a set $\mathcal{H}$ of valid arguments which supports do not depend on $\langle A, Q\rangle$ and a set $\mathcal{F}$ of almost valid arguments which supports depend on some argument in $\mathcal{H} \cup \langle A, Q\rangle$. Moreover, $\langle A, Q\rangle$ is not involved in any conflict with respect to *d-Warr'*$(> \alpha)$ and all set G' of valid arguments of strength α which do not depend on $\langle A, Q\rangle$ and the warranty of $\langle A, Q\rangle$ do not depend on any circular definition of conflict between all set $\mathcal{H}'$ of valid arguments and the set $\mathcal{F}'$ of almost valid arguments which supports depend on some argument in $\mathcal{H}' \cup \langle A, Q\rangle$. As all sets G and $\mathcal{H}$ of valid arguments of strength α which supports do not depend on $\langle A, Q\rangle$ are also valid with respect to $(Warr', Block')$ and all sets G' and $\mathcal{H}'$ of valid arguments of strength α which supports do not depend on $\langle A, Q\rangle$ are also valid with respect to $(Warr, Block)$, there should exist at least an argument $\langle B, P\rangle$ such that (i) it is almost valid with respect to a set $\mathcal{H}$ of valid arguments that satisfy Condition (b)) for argument $\langle A, Q\rangle$ and output $(Warr, Block)$, and (ii) it is not almost valid with respect to $\mathcal{H}$ and $(Warr', Block')$. Therefore, $\langle B, P\rangle$ should violate Condition (AV5) with respect to $\mathcal{H}$ and $(Warr', Block')$, and thus, for some subargument $\langle C, R\rangle \sqsubset \langle B, P\rangle$ of strength α it must hold that $R \notin$ *Warr'*(α) and $\langle C, R\rangle \notin \mathcal{H}$ and R or $\sim R \in$ *Block'*$(\geq \alpha)$. Now, as $\langle C, R\rangle \notin \mathcal{H}$ and $\langle B, P\rangle$ is almost valid with respect to $\mathcal{H}$ and $(Warr, Block)$, either $R \in$ *d-Warr*(α), or $R, \sim R \notin$ *Block*$(\geq \alpha)$. If $R \in$ *d-Warr*(α), because of the recursive warrant semantics, $\langle A, Q\rangle \not\sqsubset \langle C, R\rangle$, and thus, $R \in$ *Warr'*(α). If $R \notin$ *d-Warr*(α), we have $R, \sim R \notin$ *Block*$(\geq \alpha)$ and R or $\sim R \in$ *Block'*$(\geq \alpha)$. As *Block*$(\beta) =$ *Block'*(β) for all $\beta > \alpha$, $R, \sim R \notin$ *Block*(α) and R or $\sim R \in$ *Block'*(α). Then either $R \in$ *d-Warr*(α) or $\langle C, R\rangle$ is not valid with respect to $(Warr, Block)$, and thus, $\langle A, Q\rangle \sqsubset \langle C, R\rangle$. Now, as the warranty of $\langle A, Q\rangle$ depends on a circular definition of conflict between the set $\mathcal{H}$ and a set $\mathcal{F}$ of almost valid arguments which supports depend on some argument in $\mathcal{H} \cup \langle A, Q\rangle$ with $\langle B, P\rangle \in \mathcal{F}$, there is a cycle in the warrant dependence graph (V, E) for $\mathcal{H}$ and $\mathcal{F}$ and any argument $\mathcal{C} \in \mathcal{H}$ is such that the conclusion of $\mathcal{C}$ is either a vertex of the cycle or $\mathcal{C}$ does not satisfy Condition (a). Then, if R or $\sim R \in$ *Block'*(α) and $\langle C, R\rangle \notin \mathcal{H}$, R or $\sim R \in$ *Block*(α). Hence, *d-Warr*$(\alpha) =$ *d-Warr'*(α) and *Block*$(\alpha) =$ *Block'*(α) for all defeasibility

level α. $\square$

Since the set of conclusions that are blocked at a level is decisive for determining which arguments are valid at the next level, we have to analyze how the closure postulate reads for the maximal ideal output for an RP-DeLP program.

Proposition 18 (Closure for RP-DeLP programs) *Let $\mathcal{P} = (\Pi, \Delta, \preceq)$ be an RP-DeLP program with defeasibility levels $1 > \alpha_1 > \ldots > \alpha_p > 0$ for Δ, and let $(Warr, Block)$ be the maximal ideal output for $\mathcal{P}$. Then, if $\Pi \cup$ d-Warr$(\geq \alpha_i) \vdash_R Q$ and $\Pi \cup$ d-Warr$(> \alpha_i) \nvdash_R Q$, then either $Q \in$ d-Warr(α_i), or $Q \in$ Block$(> \alpha_i)$, or $\sim Q \in$ Block$(> \alpha_i)$.*

Proof: Suppose that for some $\alpha_i \in \{\alpha_1, \ldots, \alpha_p\}$, $\Pi \cup$ Warr$(\geq \alpha_i) \vdash_R Q$ and $\Pi \cup$ Warr$(> \alpha_i) \nvdash_R Q$ and $Q \notin$ d-Warr(α_i) and $Q, \sim Q \notin$ Block$(> \alpha_i)$. Then, as $\Pi \cup$ Warr $\nvdash \bot$, $\Pi \cup$ Warr$(\geq \alpha_i) \cup \{Q\} \nvdash \bot$, and thus, there exists a valid argument $\langle A, Q\rangle$ for conclusion Q of strength α_i. Now, as $Q \notin$ d-Warr(α_i), it can be the case that there exists a set G of valid arguments of strength α_i, with $\langle A, Q\rangle \notin G$, such that (i) $\langle A, Q\rangle \not\sqsubset C$ for all $C \in G$, and (ii) $G \cup \{\langle A, Q\rangle\}$ generates a conflict with respect to $W =$ d-Warr$(> \alpha_i) \cup \{P \mid$ there exists $\langle B, P\rangle \sqsubset \mathcal{C}$ for some $\mathcal{C} \in G \cup \{\langle A, Q\rangle\}\}$. If $G \cup \{\langle A, Q\rangle\}$ generates a conflict with respect to W, Conditions (C) and (M) hold for W, and thus, $\Pi \cup W \cup \{Q\} \cup \{P \mid \langle B, P\rangle \in G\} \vdash \bot$ and $\Pi \cup W \cup S \nvdash \bot$, for all $S \subset \{Q\} \cup \{P \mid \langle B, P\rangle \in G\}$. Consider $W' = \{R \mid \langle B, R\rangle \sqsubset \langle A, Q\rangle\}$. Then, as $W' \subseteq W$ and $\Pi \cup W' \vdash_R Q$, if $\Pi \cup W \cup \{Q\} \cup \{P \mid \langle B, P\rangle \in G\} \vdash \bot$, then $\Pi \cup W \cup \{P \mid \langle B, P\rangle \in G\} \vdash \bot$, and thus, either $Q \in$ d-Warr(α_i) , or $Q \in$ Block$(> \alpha_i)$, or $\sim Q \in$ Block$(> \alpha_i)$; i.e. either Q is warranted at level α_i, or Q is rejected at level α_i because Q or $\sim Q$ are blocked at level β with $\beta > \alpha_i$. On the other hand, it can be the case that there exists a set of valid arguments $\mathcal{H}$ of strength α_i and a set of arguments $\mathcal{F}$ of strength α_i which are almost valid with respect to $\mathcal{H} \cup \langle A, Q\rangle$, and there is a cycle in the warrant dependence graph (V, E) for $\mathcal{H}$ and $\mathcal{F}$, and any argument $\langle C, R\rangle \in \mathcal{H}$ is such that R is either a vertex of the cycle or $\langle C, R\rangle$ does not generate any conflict. As $\Pi \cup$ Warr$(\geq \alpha_i) \vdash_R Q$ and $Q \notin$ Warr$(\geq \alpha_i)$, we have that there exists a strict rule of the form $Q \leftarrow L_1 \wedge \ldots \wedge L_p \in \Pi$ with $p \geq 1$ such that $L_i \in$ Warr$(\geq \alpha_i)$ for all $L_i \in \{L_1, \ldots, L_p\}$. Moreover, as $\Pi \cup$ Warr$(> \alpha_i) \nvdash_R Q$, there exists at least one literal $L \in \{L_1, \ldots, L_p\}$ such that $L \in$ d-Warr(α_i), and thus, L is not involved in a cycle for all warrant dependence graph built over arguments at level α_i. According with the warrant dependence graph definition, as the vertex of conclusion Q is a vertex of the cycle or there exists a path from a vertex v of the cycle which is associated with some conclusion in $\mathcal{H}$ to the vertex of conclusion Q, there exists a set of valid arguments $\mathcal{H}'$ of strength α_i with $\langle A, Q\rangle \notin \mathcal{H}'$ and $\langle D, L\rangle \in \mathcal{H}'$, a set $\mathcal{F}'$ of almost valid arguments of strength α_i with $\langle A, Q\rangle \in \mathcal{F}'$, and there is a cycle in the warrant dependence graph (V, E) for $\mathcal{H}'$ and $\mathcal{F}'$, and any argument $\langle C, R\rangle \in \mathcal{H}'$ is such that R is either a vertex of the cycle or $\langle C, R\rangle$ does not generate any conflict, and there exists a path from a vertex v of the cycle which is associated with some conclusion in $\mathcal{H}'$ to the vertex of conclusion L. Hence, either $Q \in$ d-Warr(α_i) , or

$Q \in$ *Block*$(> \alpha_i)$, or $\sim Q \in$ *Block*$(> \alpha_i)$. □

Notice that for the particular case of considering a single defeasibility level for Δ, the closure property reads as follows : if $\Pi \cup$ *Warr* $\vdash_R Q$, then $Q \in$ *Warr*.

The following example shows the closure result for the maximal ideal output for an RP-DeLP program.

Example 19 *Consider the RP-DeLP program* $\mathcal{P}_{R3} = (\Pi, \Delta, \preceq)$ *with*

$$\Pi = \{\sim s \leftarrow q, \sim r \leftarrow h\} \text{ and } \Delta = \{q \leftarrow r, h \leftarrow s, r, s, q, h\},$$

and two defeasibility levels for Δ*:* α_1 *and* α_2 *with* $1 > \alpha_1 > \alpha_2 > 0$*. Consider that* Δ *is stratified as follows:*

$$\textit{level } \alpha_1\textit{:} \{q \leftarrow r, h \leftarrow s, r, s\} \quad \textit{level } \alpha_2\textit{:} \{q, h\}.$$

Obviously, s-Warr $= \emptyset$*. Then, at level* α_1*, we have two valid arguments:*

$$\mathcal{H}_1 = \langle \{r\}, r \rangle \text{ and } \mathcal{H}_2 = \langle \{s\}, s \rangle.$$

and four almost valid arguments with respect to $\{\mathcal{H}_1, \mathcal{H}_2\}$*:*

$$\mathcal{F}_1 = \langle \{r, q \leftarrow r\}, q \rangle, \quad \mathcal{F}_3 = \langle \{r, q \leftarrow r, \sim s \leftarrow q\}, \sim s \rangle,$$
$$\mathcal{F}_2 = \langle \{s, h \leftarrow s\}, h \rangle, \quad \mathcal{F}_4 = \langle \{s, h \leftarrow s, \sim r \leftarrow h\}, \sim r \rangle.$$

Figure 4 shows the warrant dependency graph for $\{\mathcal{H}_1, \mathcal{H}_2\}$ *and* $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4\}$*. The cycles express that either* r *or* s *can be warranted, but not both. Hence, at level* α_1*, we have two possible outputs for* $\mathcal{P}_{R3}$*:*

$$\textit{d-Warr}_1(\alpha_1) = \{r\}, \quad \textit{Block}_1(\alpha_1) = \{s, q\},$$
$$\textit{d-Warr}_2(\alpha_1) = \{s\}, \quad \textit{Block}_2(\alpha_1) = \{h, r\}.$$

Then at level α_2*, all arguments are rejected in both outputs, and thus, d-Warr*$_1(\alpha_2) =$ *d-Warr*$_2(\alpha_2) = \emptyset$ *and Block*$_1(\alpha_2) =$ *Block*$_2(\alpha_2) = \emptyset$*. Therefore, the two possible outputs for* $\mathcal{P}_{R3}$ *are:*

$$\textit{Warr}_1 = \{r\}, \quad \textit{Block} = \{s, q\},$$
$$\textit{Warr}_2 = \{s\}, \quad \textit{Block}_2 = \{h, r\}.$$

Consider now the maximal ideal output for $\mathcal{P}_{R3}$ *in which valid arguments involved in cycles are blocked and almost valid arguments involved in cycles are rejected. Obviously, s-Warr*$_{maximal} = \emptyset$ *and, at level* α_1*, the maximal ideal output for* $\mathcal{P}_{R3}$ *is:*

$$\textit{d-Warr}_{maximal}(\alpha_1) = \emptyset, \quad \textit{Block}_{maximal}(\alpha_1) = \{r, s\}.$$

Now, at level α_2 we have that arguments

$$\langle \{q\}, q \rangle \textit{ and } \langle \{h\}, h \rangle$$

are valid with respect to d-Warr$_{maximal}(\geq \alpha_1)$ and Block$_{maximal}(\geq \alpha_1)$ and none of them is involved in a cycle neither in a conflict, and thus, q and h are warranted conclusions at level α_2 (i.e. $\{q, h\} \subseteq$ d-Warr$_{maximal}(\alpha_2)$. Finally, although arguments

$$\langle \{q, \sim s \leftarrow q\}, \sim s \rangle \textit{ and } \langle \{h, \sim r \leftarrow h\}, \sim r \rangle$$

are recursively based on warranted conclusions, both violate Condition (V3) (i.e. $s, r \in$ Block$_{maximal}(\geq \alpha_1)$), and thus, both arguments are rejected since they are not valid. Hence, at level α_2, s and r are rejected for the maximal ideal output:

$$\textit{d-Warr}_{maximal}(\alpha_2) = \{q, h\}, \quad \textit{Block}_{maximal}(\alpha_2) = \emptyset.$$

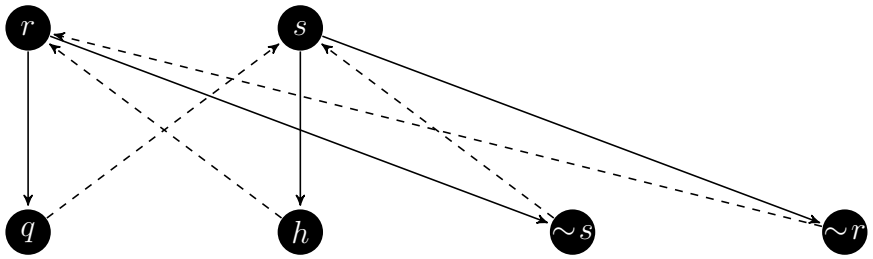

Fig. 4. Warrant dependency graph for $\mathcal{P}_{R3}$.

6 On the computation of the maximal ideal output

From a computational point of view, the maximal ideal output of an RP-DeLP program can be computed by means of a level-wise procedure, starting from the highest level and iteratively going down from one level to next level below. Then, at every level it is necessary to determine the status (warranted or blocked) of each valid argument. Next we design an algorithm which implements this level-wise procedure computing warranted and blocked conclusions by checking the existence of conflicts between arguments and cycles at some warrant dependence graph. In the following we use the notation $W(1)$ for *s-Warr*, $W(\alpha)$ and $W(\geq \alpha)$ for *d-Warr*(α) and *d-Warr*$(\geq \alpha)$, respectively, and $B(\alpha)$ and $B(\geq \alpha)$ for *Block*(α) and *Block*$(\geq \alpha)$, respectively.

Algorithm `Computing warranted conclusions`
Input $\mathcal{P} = (\Pi, \Delta, \preceq)$: An RP-DeLP program
Output (W, B): maximal ideal output for $\mathcal{P}$
Method

$W(1) := \{Q \mid \Pi \vdash_R Q\}$

$B := \emptyset$

$\alpha :=$ `maximum_level`(Δ)

while $(\alpha > 0)$ **do**

`level_computing`(α, W, B)

$\alpha :=$ `next_level`(Δ)

end while

end algorithm

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 strength α. Remark that for every level α, the procedure `level_computing` receives $W(> \alpha)$ and $B(> \alpha)$ as input and produces $W(\geq \alpha)$ and $B(\geq \alpha)$ as output.

Procedure `level_computing` (**in** α; **in_out** W, B)

VA : = $\{\langle A, Q\rangle$ with strength $\alpha \mid \langle A, Q\rangle$ is valid$\}$

while $(VA \neq \emptyset)$ **do**

while $(\exists \langle A, Q\rangle \in VA \mid \neg$ `cycle`$(\alpha, \langle A, Q\rangle, VA, W,$ `almost_valid`$(\alpha, VA, W, B))$

and $\neg$ `conflict`$(\alpha, \langle A, Q\rangle, VA, W,$ `not_dependent`$(\alpha, \langle A, Q\rangle, VA, W, B))$ **do**

$W(\alpha) := W(\alpha) \cup \{Q\}$

$VA := VA \backslash \{\langle A, Q\rangle\} \cup \{\langle C, P\rangle$ with strength $\alpha \mid \langle C, P\rangle$ is valid$\}$

end while

$I := \{\langle A, Q\rangle \in VA \mid$ `conflict`$(\alpha, \langle A, Q\rangle, VA, W, \emptyset)$

or `cycle`$(\alpha, \langle A, Q\rangle, VA, W,$ `almost_valid`$(\alpha, VA, W, B))\}$

$B(\alpha) := B(\alpha) \cup \{Q \mid \langle A, Q\rangle \in I\}$

$VA := VA \backslash I$

end while

end procedure

For all level α the procedure `level_computing` first computes the set *VA* of valid arguments with respect to $W(> \alpha)$ and $B(> \alpha)$. Then, this set of valid arguments is dynamically updated depending on new warranted and blocked conclusions with strength α. The procedure `level_computing` finishes when the status for every valid argument is computed. The status of a valid argument is computed by means of the four following auxiliary functions.

Function `almost_valid`(**in** α, *VA*, W, B) **return** *AV*: set of arguments

AV := $\{\langle C, P\rangle$ with strength α **such that**

$\langle C, P\rangle$ satisfies Conditions (AV1)-(AV6) with respect to *VA*$\}$

end function

Function `not_dependent`(**in** α, $\langle A, Q\rangle$, *VA*, W, B)
return *ND*: set of almost valid arguments which do not depend on Q

AV := `almost_valid`(α, *VA*, W, B)

ND := $\{\langle C, P\rangle \in AV \mid \langle A, Q\rangle \not\sqsubset \langle C, P\rangle\}$

end function

Function `conflict`(**in** α, $\langle A, Q\rangle$, *VA*, W, *ND*) **return** `con`: Boolean

`con` := $\exists\, S \subseteq VA \backslash \{\langle A, Q\rangle\} \cup ND$ **such that**

$\Pi \cup W(\geq \alpha) \cup \{P \mid \langle C, P\rangle \in S\} \nvdash \bot$ **and**

$\Pi \cup W(\geq \alpha) \cup \{P \mid \langle C, P\rangle \in S\} \cup \{Q\} \vdash \bot$

end function

Function `cycle`(**in** α, $\langle A, Q\rangle$, *VA*, W, *AV*) **return** `cy`: Boolean

`cy` := there is a cycle in the warrant dependence graph for *VA* and *AV*

and the vertex of $\langle A, Q\rangle$ is a vertex of the cycle **or** there exists a

path from a vertex in *VA* of the cycle to the the vertex of $\langle A, Q\rangle$

end function

The function `conflict` checks (possible) conflicts among the argument $\langle A, Q\rangle$ and the set *VA* of valid arguments extended with the set *ND* of arguments. The set *ND* of arguments takes two different values: the empty set and the set of almost valid arguments whose supports depend on some argument in $VA \backslash \{\langle A, Q\rangle\}$. The empty set value is used to detect conflicts between the argument $\langle A, Q\rangle$ and the arguments in *VA*, and thus, every valid argument involved in a conflict is blocked. On the other hand, the value set of almost valid arguments which do not depend on argument $\langle A, Q\rangle$ is used to detect possible conflicts between the argument $\langle A, Q\rangle$ and the arguments in $VA \cup ND$, and thus, every valid argument involved in a possible conflict remains as valid. In fact, the function `almost_valid` computes the set of almost valid arguments that satisfy Conditions (AV1)-(AV6) with respect to the current set of valid arguments. The function `not_dependent` considers almost valid arguments with respect to the current set of valid arguments which do not depend on $\langle A, Q\rangle$. Finally, the function `cycle` checks the existence of a cycle in the warrant dependence graph for the current set of valid arguments and its set of almost valid arguments, and verifies whether the vertex of argument $\langle A, Q\rangle$ is in the cycle or there exists a path from a vertex of the cycle to it.

One of the main advantages of the maximal ideal warrant recursive semantics for RP-DeLP is from the implementation point of view. Warrant semantics based on dialectical trees, like DeLP [12,14], might consider an exponential number of arguments with respect to the number of rules of a given program. The previous algorithm can be implemented to work in polynomial space, with a complexity upper bound equal to P^{NP}.

This can be achieved because it is not actually necessary to find all the valid arguments for a given literal Q, but only one witnessing a valid argument for Q is enough. Analogously, function `not_dependent` can be implemented to generate at most one almost valid argument, not dependent on $\langle A, Q\rangle$, for a given literal. The only function that in the worst case can need an exponential number of arguments is `cycle`, but next we show that whenever `cycle` returns true for $\langle A, Q\rangle$, then a conflict will be detected with the almost valid arguments not dependent on $\langle A, Q\rangle$, so warranted literals can be detected without function `cycle`. Also, blocked literals detected by function `cycle` can also be detected by checking the stability of the set of valid arguments after two consecutive iterations, so it is not necessary to explicitly compute warrant dependency graphs.

Proposition 20 (Optimization) *Let $\mathcal{P} = (\Pi, \Delta, \preceq)$ be an RP-DeLP program with defeasibility levels $1 > \alpha_1 > \ldots > \alpha_p > 0$ for Δ, and let W and B be two sets of warranted and blocked conclusions with strength $\geq \alpha_i$, respectively. If VA is the set of all d-arguments of strength α_i that are valid with respect to (W, B) and AV is the set of all d-arguments of strength α_i that are almost valid with respect to VA, we get the following results:*

- *(i) If there is a cycle in the warrant dependence graph for VA and AV, and $\langle A, Q\rangle \in$ VA is such that the vertex of conclusion Q is a vertex of the cycle or there exists a path from a vertex of the cycle to the the vertex of conclusion Q, then there exists a set ND $\subseteq$ AV such that $\langle A, Q\rangle \not\sqsubseteq \langle R, P\rangle$ for all $\langle R, P\rangle \in$ ND, and there exists a set $S \subseteq$ VA$\backslash\{\langle A, Q\rangle\}$ such that $\Pi \cup W \cup \{P \mid \langle B, P\rangle \in S\} \cup$ ND $\not\vdash \bot$ and $\Pi \cup W \cup \{P \mid \langle B, P\rangle \in S\} \cup$ ND $\cup \{Q\} \vdash \bot$.*
- *(ii) If for all $\langle A, Q\rangle \in$ VA there exists a set ND $\subseteq$ AV such that $\langle A, Q\rangle \not\sqsubseteq \langle R, P\rangle$, for all $\langle R, P\rangle \in$ ND, and there exists a set $S \subseteq$ VA$\backslash\{\langle A, Q\rangle\}$ such that $\Pi \cup W \cup \{P \mid \langle B, P\rangle \in S\} \cup$ ND $\not\vdash \bot$ and $\Pi \cup W \cup \{P \mid \langle B, P\rangle \in S\} \cup$ ND $\cup \{Q\} \vdash \bot$, then there is at least a cycle in the warrant dependence graph for VA and AV, and every $\langle A, Q\rangle \in$ VA is such that the vertex of conclusion Q is a vertex of a cycle or there exists a path from a vertex of a cycle to the the vertex of conclusion Q.*

Proof:

(i) If the vertex of conclusion Q is a vertex of the cycle, because of the warranty dependency graph definition, we can consider the set *ND* $\subseteq$ *AV* such that the vertex of each conclusion in ND is a vertex of the cycle and $\langle A, Q\rangle \not\sqsubseteq \langle R, P\rangle$ for all $\langle R, P\rangle \in$ *ND*, and then, there should exist a set $S \subseteq$ *VA*$\backslash\{\langle A, Q\rangle\}$ such

that $\Pi \cup W \cup \{P \mid \langle B, P\rangle \in S\} \cup \mathit{ND} \nvdash \bot$ and $\Pi \cup W \cup \{P \mid \langle B, P\rangle \in S\} \cup \mathit{ND} \cup \{Q\} \vdash \bot$. If the vertex of conclusion Q is not a vertex of the cycle and there exists a path from a vertex of the cycle to the the vertex of conclusion Q, we can consider the set $ND \subseteq \mathit{AV}$ such that the vertex of each conclusion in ND is a vertex of the cycle. Now, because of the warranty dependency graph definition, $\langle A, Q\rangle \not\sqsubset \langle R, P\rangle$ for all $\langle R, P\rangle \in ND$ and there should exist a set $S \subseteq \mathit{VA} \backslash \{\langle A, Q\rangle\}$ such that $\Pi \cup W \cup \{P \mid \langle B, P\rangle \in S\} \cup \mathit{ND} \nvdash \bot$ and $\Pi \cup W \cup \{P \mid \langle B, P\rangle \in S\} \cup \mathit{ND} \cup \{Q\} \vdash \bot$.

(ii) We have that for all $S \subseteq \mathit{VA}$, $\Pi \cup W \cup \{P \mid \langle R, P\rangle \in S\} \nvdash \bot$ and that for all $\langle A, Q\rangle \in \mathit{VA}$ there exists a set $\mathit{ND} \subseteq \mathit{AV}$ such that $\langle A, Q\rangle \not\sqsubset \langle R, P\rangle$ for all $\langle R, P\rangle \in \mathit{ND}$, and there exists a set $S \subseteq \mathit{VA} \backslash \{\langle A, Q\rangle\}$ such that $\Pi \cup W \cup \{P \mid \langle B, P\rangle \in S\} \cup \mathit{ND} \nvdash \bot$ and $\Pi \cup W \cup \{P \mid \langle B, P\rangle \in S\} \cup \mathit{ND} \cup \{Q\} \vdash \bot$. Then, for all $\langle A, Q\rangle \in \mathit{VA}$, we have that the warranty of Q depends on a possible conflict with a set $S \subseteq \mathit{VA} \backslash \{\langle A, Q\rangle\}$ and a set $\mathit{ND} \subseteq \mathit{AV}$ such that $\langle A, Q\rangle \not\sqsubset \langle R, P\rangle$ for all $\langle R, P\rangle \in \mathit{ND}$. Therefore, because of the warranty dependency graph definition, there should exists a cycle in the warrant dependence graph (V, E) for *VA* and *AV* such that the vertexes of conclusions of ND are vertexes of the cycle and the vertexes of conclusions of S and $\{\langle A, Q\rangle\}$ are vertexes of the cycle or there exists a path from a vertex of the cycle to the the vertex of these conclusions.

□

Finally, observe that the following queries can be implemented with NP algorithms:

(1) Whether a literal P is a conclusion of some argument returned by
`not_dependent`(α, $\langle A, Q\rangle$, *VA*, W, B).
To check the existence of an almost valid argument $\langle C, P\rangle$ not dependent on $\langle A, Q\rangle$, we can non-deterministically guess a subset of rules, and check in polynomial time whether they actually generate the desired argument for P, as all the conditions for an almost valid argument can be checked in polynomial time and also the condition of not being dependent on the literal Q.

(2) Whether the function
`conflict`(**in** α, $\langle A, Q\rangle$, *VA*, W, *ND*)
returns true. To check the existence of a conflict, we can non-deterministically guess a subset of literals S from $\{P \mid \langle C, P\rangle \in \mathit{VA} \setminus \{\langle A, Q\rangle\} \cup \mathit{ND}\}$ and check in polynomial time whether i) $\Pi \cup W(\geq \alpha) \cup S \nvdash \bot$ and ii) $\Pi \cup W(\geq \alpha) \cup S \cup \{Q\} \vdash \bot$.

Then, as the maximum number of times that these queries need to be executed before the set of conclusions associated with VA becomes stable is polynomial in the size of the input program, the P^{NP} upper bound follows.

7 SAT encodings for finding warranted literals

From a computational point of view, the maximal ideal output for an RP-DeLP program can be computed by means of a level-wise procedure, starting from the highest level and iteratively going down from one level to next level below. At every level it is necessary to determine the status (warranted or blocked) of each valid argument by checking the existence of both conflicts between arguments, and cycles at the warrant dependence graphs. In the previous section we showed that this level-wise procedure can be implemented to work in polynomial space. On the one hand this can be achieved because it is not actually necessary to find all the valid arguments for a given literal, it is enough to find only one. Actually, in our implementation to explain the existence of a valid argument for a literal Q we simply record the *last* rule of the argument, that is, a rule with Q as conclusion, and with all the literals of its body as warrants. To give a full explanation for a valid argument, we recursively give explanations for all the warrants of the body of the rule. Something similar applies to the computation of at most one almost valid argument for a given literal. This will be done with the first of the two SAT encodings we present next, and it allows also to explicitly give an almost valid argument for a literal, not only to check the existence. On the other hand, the existence of cycles in the warrant dependency graph among valid and almost valid arguments can be validated by checking the stability of conflicts between valid and almost valid arguments, so it is not necessary to explicitly compute the warrant dependency graphs. Hence, the procedure to find warranted literals needs to compute two main queries during its execution: (i) whether an argument is almost valid, and (ii) whether there is a conflict among valid and almost valid arguments.

In this section we present SAT encodings for these two main combinatorial queries. The input and output specification of each query is as follows:

(i) `Almost valid argument`: It takes as **input** a defeasibility degree α, a literal P, sets W and B of warranted and blocked literals of strength $\geq \alpha$, respectively, a set *VA* of valid arguments of strength α, and an argument $\langle A, Q\rangle \in$ *VA*. It **computes** an almost valid argument $\langle C, P\rangle$ of strength α that does not depend on $\langle A, Q\rangle$.

(i) `Conflict`: It takes as **input** a defeasibility degree α, a set W of warranted literals of strength $\geq \alpha$, a set *VA* of valid arguments of strength α, a valid argument $\langle A, Q\rangle$ of strength α, and a set *ND* of almost valid arguments of strength α that do not depend on $\langle A, Q\rangle$. It **checks** (possible) conflicts among the argument $\langle A, Q\rangle$ and the set *VA* of valid arguments extended with the set *ND* of almost valid arguments.

The idea for encoding the problem of searching almost valid arguments is based on the same behind successful SAT encodings for solving STRIPS planning problems [21]. In a STRIPS planning problem, given an initial state, described with a set of predicates, the goal is to decide whether a desired goal state can be achieved by means of the application of a suitable sequence of actions. Each action has a set preconditions, when they hold true the action can be executed and as a result certain facts become true and some others become false (its effects). Hence executing an action changes the current state, and the application of a sequence of actions creates a sequence of states. The planning problem is to find a sequence of actions such that, when executed, the obtained final state satisfies the goal state.

In our case, the search for an almost valid argument $\langle C, P\rangle$ can be seen as the search for a *plan* for producing P, taking as the initial set of facts some subset of a set of literals in which we already trust. We call such initial set the base set of literals [10], and we say that they are true at the first step of the argument. For looking for an almost valid argument $\langle C, P\rangle$, we will consider what rules should be executed, such that starting from the initial set will finally obtain the desired goal P. We say that a rule R can be executed starting from a set of literals S, when $Body(R) \subseteq S$, and that when it is executed we obtain a new set $S \cup \{Head(R)\}$. We have to consider only some rules for looking for almost valid arguments of strength α for literals not yet warranted, as we have explained in the previous section, that is, the $\alpha-$rules we have defined before.

We use the following sets of literals and rules to define our SAT encoding. Consider first the initial set S_0:

$$S_0 = \{L \mid L \in W(\geq \alpha) \text{ or } \exists\langle C, L\rangle \in VA\}$$

which is the base set of warranted and valid literals. If we execute *all* the $\alpha-$rules that can be executed from S_0, that is:

$$R_0 = \{R \mid R \in \alpha-\text{rules}\ ,\ Body(R) \subseteq S_0\}$$

we obtain a new state S_1 that contains S_0 plus the heads of all the executed rules. This process can be repeated iteratively, obtaining a sequence of sets of literals $\mathcal{S} = \{S_0, S_1, \ldots, S_t\}$ and a sequence of sets of executed rules $\mathcal{R} = \{R_0, R_1, \ldots, R_{t-1}\}$, until we reach a final state S_t in which the execution of any possible rule does not increase the set of literals already in S_t. If starting from an initial set S_0 that contains all the current valid and warranted literals the final state S_t contains P, that means that an almost valid argument for P could be obtained from the sequence of executed rules, if we could find a subset of rules such that they can form an argument that satisfies all the conditions for an almost valid argument for P.

[10] For an almost valid argument, the base set can contain only warranted and valid literals.

Observe that an almost valid argument $\langle C, P\rangle$ with strength α can only exist if the following conditions, that can be checked in polynomial time, are satisfied:

(1) $P \notin \boldsymbol{W}(> \alpha) \cup \boldsymbol{B}(> \alpha)$. This is actually condition (AV2).
(2) $\sim P \notin \boldsymbol{B}(> \alpha)$. This is actually the first part of condition (AV3).
(3) There does not exist a valid d-argument for conclusion P of strength α. This is actually condition (AV4).
(4) $P \in S_t$.

If the previous conditions are satisfied, we proceed the search for $\langle C, P\rangle$ with strength α with a SAT encoding from the sequences $\mathcal{S}$ and $\mathcal{R}$ defined above.

That is, a SAT instance with variables to represent all the possible literals we can select from each set S_i:

$$\{v_L^i \mid L \in S_i, 0 \leq i \leq t\}$$

plus variables to represent all the possible rules R we can select from each set R_i:

$$\{v_R^i \mid R \in R_i, 0 \leq i < t\}$$

In order to check that the variables set to true represent an almost valid argument, we add clauses for ensuring that:

(1) If variable v_L^i is true, then either v_L^{i-1} is true or one of the variables v_R^{i-1}, with $\boldsymbol{Head}(R) = L$, is true.
(2) If a variable v_R^i is true, then for all the literals L in its body v_L^i must be true.
(3) If variable v_L^i is true, then v_L^{i+1} is also true.
(4) The variable v_P^t must be true.
(5) No two contradictory variables v_L^t and $v_{\sim L}^t$ can be both true.

In addition, in order to satisfy the consistency of the literals of the argument with respect to the closure of the strict knowledge Π, we create also an additional set of variables V_Π and set of clauses R_Π. The set of variables V_Π contains a variable v_L^Π for each literal that appears in the logical closure of the set $S_t \cup W$ with respect to the strict rules.

Then, we add the following clauses to check the consistency with Π:

(1) If a literal is selected for the argument (v_L^t set to true) then v_L^Π must also be true.
(2) For any $L \in W$, v_L^Π must be true.
(3) For any rule $R \in \Pi$ that was executed when computing the logical closure, if for all the literals L in its body v_L^Π is true, then $v_{\boldsymbol{Head}(R)}^\Pi$ must be true.
(4) No two contradictory variables v_L^Π and $v_{\sim L}^\Pi$ can be both true.

Observe that this *layered* encoding for searching almost valid arguments allows to explicitly recover the full structure of the argument, because we have both the

literals and the rules that have generated them at each step of the argument.

We next show that any solution for a formula obtained with this SAT encoding gives an almost valid argument $\langle C, P\rangle$; i.e. we show that $\langle C, P\rangle$ satisfy Conditions (AV1)-(AV6):

(AV1) Given that the only possible rules that can be selected for building the almost valid argument are those defined as α-rules, the conclusion of an α-rule can only become valid with strength at most α. So, the only possible subarguments with strength greater than α are the ones corresponding to literals that can selected from the set S_0. Observe that the literals in the set S_0 are all the literals at the current set $W(\geq \alpha)$ plus the current set of literals with valid arguments with strength α. It follows that the only subarguments of strength $\beta > \alpha$ that can be implicitly used are the ones corresponding to warranted literals.

(AV2) This condition is actually checked before creating the SAT encoding. That is, if the condition is not satisfied, we answer that there is no such almost valid argument.

(AV3) The first part of this condition $\sim P \notin B(> \alpha)$ is also checked before creating the SAT encoding. For the second part, first observe that for any subargument $\langle C, R\rangle \sqsubset \langle B, P\rangle$, v_R^t will be true, due to the clauses in $(A3)$, as long as v_P^t (due to the clauses in $(A4)$). Then the clauses in $(B1)$ ensure that for any literal L with v_L^t true, the corresponding variable v_L^Π of the second part of the encoding will be also true. Finally, the clauses in $(B2)$, $(B3)$ and $(B4)$ will ensure that all such true literals are consistent with $\Pi \cup W$.

(AV4) As in the condition $(AV2)$, this condition is checked before creating the SAT encoding.

(AV5) Any literal L that is part of the argument ($v_L^t = true$) will be either generated by an α-rule, so it holds that $L, \sim L \notin W(\geq \alpha) \cup B(\geq \alpha) \cup$ *VA*, or it is already true at the initial set ($v_L^0 = true$), so it is warranted or valid.

(AV6) Observe that there is no $R \in \alpha$-rules such that *Head(R)* $\in B \cup W \cup$ *VA*, then it cannot be that all the rules used in the argument for P depend only on warranted literals from S_0 because that would mean that P is indeed valid (so P would have to be in *VA*). So, from the initial set S_0 at least one valid, but not warranted, literal will be activated, if any almost valid argument for P exists.

7.2 Looking for collective conflicts

We reduce the query computed by function `conflict`, to a query where we consider finding the set of conflict literals that are the conclusions of the corresponding conflict set of arguments. Basically, for finding this conflict set of literals S for a valid argument $\langle A, Q\rangle$ from the base set of literals considered in function `conflict`, i.e. the set $G = \{P \mid \langle C, P\rangle \in$ *VA* $\setminus \{\langle A, Q\rangle\} \cup$ *ND*$\}$, we have to find two arguments $\langle A_1, L\rangle$, $\langle A_2, \sim L\rangle$ using only rules from Π, literals $W \cup \{Q\}$ and a

subset S from G, but such that when Q is not used, no conflict (generation of L and $\sim L$ for any L with strict rules) is produced with such set S. So, this can be seen as a simple extension of the previous query, where now we have to look for two arguments, instead of only one, although both arguments must be for two contradictory literals. That is, the SAT formula contains variables for encoding arguments that use as base literals $W \cup G \cup \{Q\}$ and rules from Π (with the same scheme of the previous SAT encoding for almost valid arguments), with an additional set of conflict variables to encode the set of possible conflicts that can be, potentially, generated from $W \cup G \cup \{Q\}$ using rules from Π, in order to be able to force the existence of at least one conflict. There is also an additional set of variables and clauses for encoding the subproblem of checking that S, when Q is not used, does not generate any conflict.

So, the SAT formula contains two different parts. A first part is devoted to checking that the selected set of literals S plus $\{Q\}$ is a conflict set (i.e. if $\Pi \cup W(\geq \alpha) \cup S \cup \{Q\} \vdash \bot$). This set of variables and clauses is similar to the previous one for finding almost valid arguments, but in this case is used for finding two arguments starting from a subset of $W \cup G$ and forcing the inclusion of $\{Q\}$. That is, the SAT clauses of this first part are:

(1) A clause that states that the literal Q must be true at the first step.
(2) A clause that states that at least one conflict variable c_L must be true.
(3) For every conflict variable c_L, a clause that states that if c_L is true then literals L and $\sim L$ must be true at the final step of the argument.
(4) The rest of clauses are the same ones described in the first part of the previous encoding, except the clauses of the item 5 that are not included, but now considering as possible literals and rules at every step the ones computed from the base set $W \cup G \cup \{Q\}$ and using only strict rules.

The process for computing the possible literals and rules that can be potentially applied in every step of the argument is the same forward reasoning process presented for the previous encoding. This same process is used for discovering the set of conflict variables c_L that need to be considered, because we can potentially force the conflict c_L if at the end of this process both L and $\sim L$ appear as reachable literals.

A second part of the SAT formula is devoted to checking that the selected set of variables and clauses S at the first step, without using Q, does not cause any conflict with the strict rules. So this second part of the formula contains a variable for any literal that appears in the logical closure of $G \cup W$ with respect to the strict rules. Actually, this second part of the formula is analogous to the second part of the formula for the previous encoding.

Observe that this encoding for searching conflicts for Q not only allows to check the existence of conflicts, but it also gives an explicit conflict set: the variables set to true that represent the chosen set S, together with almost valid arguments for

those literals in S that have arguments in *ND*. So, we can explain the reasons for each conflict detected.

8 Average computational cost and easy/hard problem instances

To study the scaling behavior of the (average) computational cost of our P^{NP} algorithm as the size increases, as well as how different characteristics of the problem instances affect its computational cost, we have implemented our algorithm and conducted a series of experiments.

The main algorithm has been implemented with python, but for solving the SAT formulas presented in the previous section, the algorithm uses a SAT solver, that can be either MiniSAT [18] or Glucose [8]. However, our architecture easily allows to use any other SAT solver that appears in the future. Minisat is one of the publicly available SAT solvers which implements most of the current state-of-the-art solving techniques such as conflict-clause recording and conflict-driven backjumping, among others. Glucose, that has some common parts with MiniSAT, implements some new learning mechanisms which made it award winning on SAT 2011 competition. As we have mentioned at the introduction, we have a preliminary version of the algorithm that works with ASP encodings [4] instead of with SAT encodings, although the current ASP based version only works with one defeasible level. In the near future, we plan to improve the ASP based version to be able to work with multiple levels.

In the experiments the algorithm solves different test-sets of problem instances obtained with a random generation algorithm. To study and analyze how our RP-DeLP algorithm behaves as different characteristics of the problem change, we generated our instances using one and two levels of defeasibility and changing the other parameters of the problem instances.

8.1 Random generation of RP-DeLP problem instances

We used different parameters to control the generation of random RP-DeLP problem instances with different sizes, defeasibility levels and other characteristics. We focused or experimentation first on one set of problems with only one defeasible level and then on another set with two defeasible levels. In both cases we were interested in how the resolution time differs when the ratio of clauses to the number of variables increases. Then, in the first case with only one defeasible level, we were also interested in the results when the fraction of clauses of the program at the strict knowledge level is modified, ranging from no strict knowledge at all to all clauses at the strict knowledge level. For the case of two defeasible levels, we have investigated the effect of modifying the fraction of clauses between the two defeasible levels. We next explain the generation of our problem instances.

Generation of instances with one defeasible level. Given a number of variables

(V), a maximum clause length (ML), a ratio of clauses to variables (C/V), and a fraction (f), between 0.0 an 1.0, of strict knowledge, the algorithm generates a RP-DeLP problem instance by generating C clauses, such that the length of every clause is selected uniformly at random from $[1, ML]$ (clauses with length 1 are facts). The variables of the literals of a clause are selected uniformly at random without repetition, and are negated with probability 0.5. From the C clauses, $f \cdot C$ clauses are in the strict knowledge and the rest in the defeasible set.

Two defeasible levels instance generation. Similar to the previous instance generator with a number of variables (V), a maximum clause length (ML), a ratio of clauses to variables (C/V), now we fix the fraction of strict knowledge (f) to 0.1. Then two defeasible levels are built assigning a fraction l between 0.0 and 1.0 of the total number of defeasible clauses to the first defeasible level and 1-l to the second defeasible level.

8.2 Test instances considered

We generated two different groups of test sets: test sets with one defeasible level and test sets with two defeasible levels. In both groups, test instances were created with a number of variables (V) selected from $\{20, 30\}$, [11] and with maximum clause length (ML) selected from $\{2, 4\}$.

In the case of one defeasible level, for each combination (V, ML), different test sets of instances were created by selecting a number of total clauses, such that the ratio C/V ranged from 1 to 12 in steps of 1, and the fraction of clauses in the strict knowledge ranged from 0 to 0.9 in steps of 0.1. So, the total number of test sets for each combination (V, ML) was 90. The number of instances generated in each test set was 50.

In the case of two defeasible levels, for each combination (V, ML) and an strict knowledge fraction set to 0.1, different test sets of instances were created by selecting a number of total clauses, such that the ratio C/V ranged from 1 to 12 in steps of 1, and the fraction of clauses in the first level l ranged from 0.1 to 0.9 in steps of 0.1.

8.3 Empirical results

For one defeasible level, we analyze the results for instances with 30 variables and maximum clause length 2. The left plot of Figure 5 shows the median time to solve

[11] Notice that the total number of literals will be two times the number of variables.

the instances with our algorithm when solving instances with different ratio of the number of total clauses to number of variables (axis labelled with *C/V* in the plots) and with different fraction of strict knowledge (axis labelled with *strict knowledge*). The plot shows that for a strict knowledge fraction of 0.0, there is an increase of the median time as the total number of clauses increases. By contrast, as the strict knowledge fraction increases, the time increases only up to certain value of the number of total clauses, and then drops significantly. This is probably because of two causes. The more strict knowledge we have, the more possibilities to have inconsistent instances, that are detected in polynomial time by our algorithm, and the more unacceptable arguments and blocked literals we can have.

To check the possible role of inconsistent instances on the complexity of the problem, we have also computed what fraction of the instances, for each test set of 50 instances, are inconsistent ($\Pi \vdash_R \bot$). The right plot of Figure 5 shows this information. The color scale ranges from points with a fraction of instances with inconsistent strict knowledge equal to 0 (dark blue color) to points with such fraction equal to 1.0 (red color). Apart for the obvious case of strict knowledge fraction equal to 0.0, where there are never inconsistent instances, for a fraction of strict knowledge equal to 0.1 up to the ratio $C/V = 6$ no inconsistent strict knowledge is generated, but the time needed to solve the instances is smaller than the one needed for instances with no strict knowledge at all.

As the fraction of strict knowledge increases, test instances with inconsistent strict knowledge appear more frequently for a lower ratio C/V and the interval of values of C/V with instances with significant computation time (greater than 0) decreases. Also, the highest computation time obtained decreases as the fraction of strict knowledge increases.

To further understand the reasons for such differences on the computation time, we have also studied the average ratio of warranted literals and average ratio of blocked literals, with respect to the total number of variables, for each test set. The left plot of Figure 6 shows the ratio of warranted literals and the right plot the ratio of blocked literals. Looking at both plots, we observe that for instances with low C/V, if its strict knowledge fraction is also low, we have a small, but non-negligible, fraction of warranted literals, that starts to increase as we increase C/V, but only up to certain limit C/V (around 2.0), and above that limit the fraction of warranted literals starts to decrease, coinciding with an increase in the fraction of blocked literals. A plausible explanation for this is that for very low C/V instances have very few valid arguments, so few warranted and blocked literals are produced. As C/V increases, more valid arguments start to appear, but obviously as the number of valid arguments increases more and more of them will be part of a conflict set of arguments. So, it seems that the highest computation times are found for instances with enough clauses such that many valid arguments are found, but many of them are also found to be part of a collective conflict set.

When the strict knowledge fraction increases, as the fraction of instances with inconsistent strict knowledge increases, it is clear that on average warranted and blocked literals will decrease, and this is observed on both plots. It is also natural that even on instances with a consistent strict knowledge, when this fraction is larger, less literals will have valid arguments, because consistency with the strict knowledge will hold for less arguments. However, we still find remarkable the increase of easy instances for a strict knowledge fraction of only 0.2, because at this strict knowledge fraction for C/V up to 5.0 instances still have warranted literals. A possible explanation for this increase of easy instances even when we still have a considerable number of warranted literals, is that the fraction of strict knowledge produces the pruning of larger arguments, so the arguments found for warranted literals are shorter and easier to find.

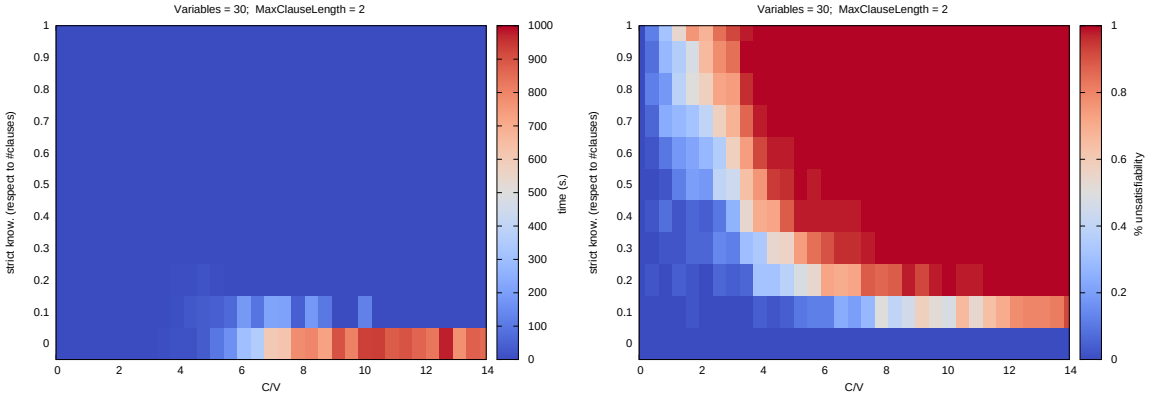

Fig. 5. Median time to solve the instances (left) and fraction of inconsistent instances (right) for $V = 30, ML = 2$.

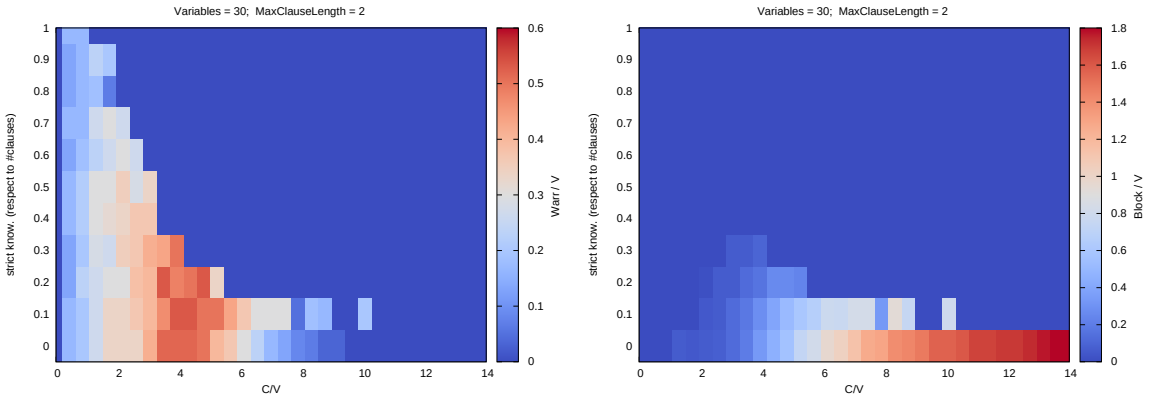

Fig. 6. Warranted literals (left) and blocked literals (right) for $V = 30, ML = 2$.

Next, we analyze the effect on complexity of having two defeasible levels, instead of just one. For these instances we have fixed the strict knowledge fraction to 0.1 because we wanted to test the hardest possible instances we can have when there is a fraction of strict knowledge greater than zero, so we still can have non-trivial conflicts between arguments due to the role of the strict knowledge on collective conflicts.

The left plot of Figure 7 shows the median time to solve the instances with our algorithm when solving instances with different ratio of the number of total clauses to number of variables (axis labelled with *C/V* in the plots) and with different fraction of defeasible knowledge at the first defeasible level (axis labelled with *fraction* l). The right plot of the same figure shows the percentage of consistent instances. We observe that as before, just up to the ratio where almost all instances are inconsistent, there is an increase on the median time.

However, the lowest computation times are found on a range of values for the first level fraction around 0.5, and where this fraction is near 0 or 1 the computation time increases. A possible explanation for this concentration of the hardest instances when the defeasible knowledge is unbalanced (concentrated almost in one level) may be the following.

When almost all the clauses are in one level, we have more possible acceptable arguments in that level. Then, the space of possible collective conflicts at that level is also larger, so the computation times for the conflict queries will be higher. However, there is an slight difference in the computational cost when ($l \approx 0$) and when ($l \approx 1$). Despite in both cases we have the same unbalance of clauses between levels, having ($l \approx 0$) means that the contribution to the output of the program due to the first level will be small and quickly computed. At the second level, where the input will include the warrants from the strict part plus some warrants obtained from the first defeasible level, we will have less possible acceptable arguments from the second level than we would have if the first defeasible level would be empty. So, the total computational effort should be smaller than if all the defeasible knowledge would be only at one level. When we have the situation where ($l \approx 1$), almost all the defeasible knowledge is at the first level, so the computational effort to compute the output of the first level increases. That is, the input for the first defeasible level will contain only the warrants from the strict part, so the set of possible acceptable arguments will be larger (compared with the second defeasible level when ($l \approx 0$)) and the set of possible warrants and blocked literals to check will be larger. Observe that the plot for blocked literals at the right of Figure 8, shows a larger ratio $|B|/V$ for $l = 0.9$, as the number of clauses increases, than for $l = 0.1$.

So, when the fraction of clauses at the two defeasible levels is near 0.5, the number of warrants obtained from the first defeasible level will increase with respect to $l = 0.1$, but the number of blocked literals will be smaller than for $l = 0.9$, because the number of clauses at the first deafeasible level is smaller. At the second defeasible level, the warrants and blocked literals will decrease, with respect to the case $l = 0.1$, given the input from the previous level. Looking at the left plot of Figure 8, that shows the ratio of warranted literals and the right plot the ratio of blocked literals, we clearly observe that more warranted literals are obtained around $l = 0.5$ but less blocked literals than at the extreme values of l.

Those results show that when defeasible levels are balanced in terms of number of

clauses, there are less conflicts between arguments at the same level. That means that more literals can be warranted, and as it has been shown the lack of conflicts decreases the computation time of the output.

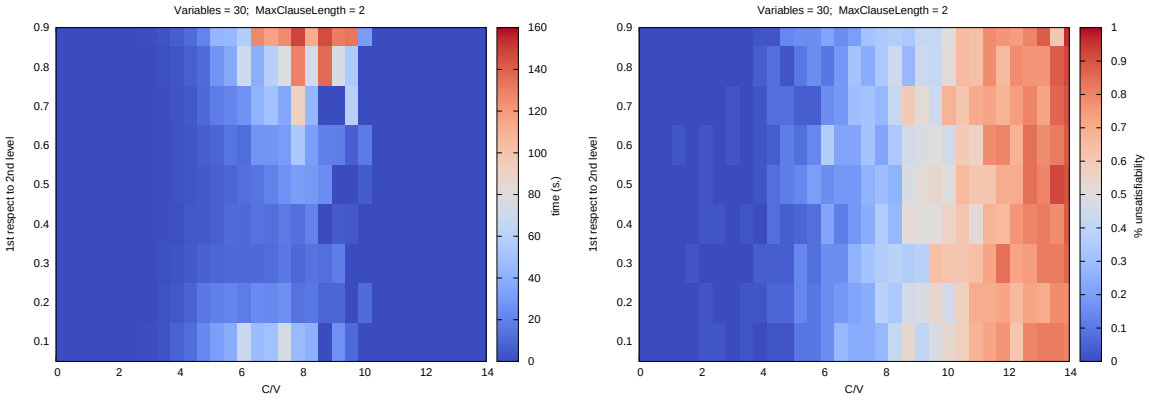

Fig. 7. Average computational cost (left) and fraction of inconsistent instances (right) for $V = 30, ML = 2$, fraction of strict knowledge = 0.1 and two defeasible levels.

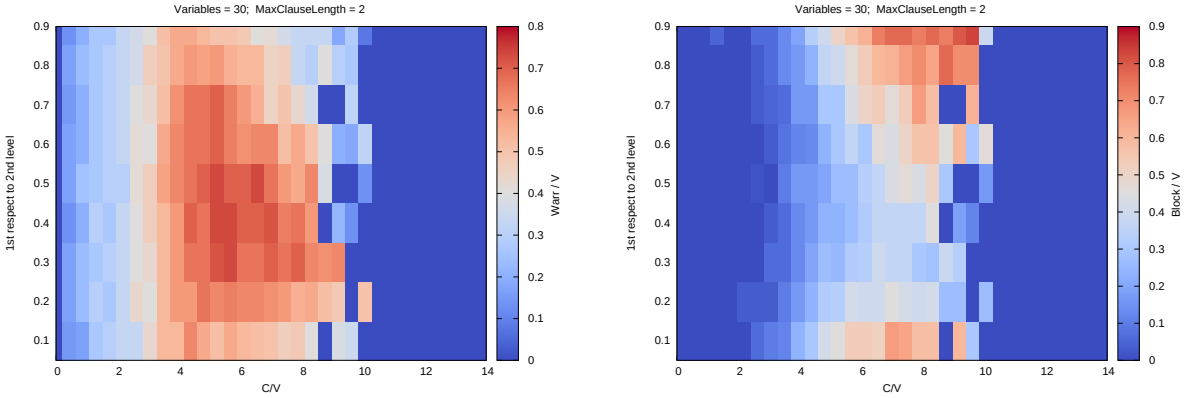

Fig. 8. Warranted literals (left) and blocked literals (right) for $V = 30, ML = 2$, fraction of strict knowledge = 0.1 and two defeasible levels.

9 Conclusions and future work

In this paper we have introduced a new recursive semantics for determining the warranty status of arguments in defeasible argumentation. The distinctive features of this semantics, e.g. with respect to Pollock's critical link semantics, are: (i) it is based on a non-binary notion of conflict in order to preserve consistency with the strict knowledge and (ii) besides the set of warranted and rejected conclusions, we introduce the set of blocked conclusions, which are those conclusions which are based on warranted information but they generate a conflict with other already warranted arguments of the same strength.

As future work, we plan to improve the efficiency of an algorithm we have already designed by minimizing the effective number of NP queries that have to be made during its execution. 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.

References

[1] Teresa Alsinet, Ramón Béjar, and Lluis Godo. A characterization of collective conflict for defeasible argumentation. In *Computational Models of Argument: Proceedings of COMMA 2010*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, pages 27–38. IOS Press, 2010.

[2] Teresa Alsinet, Ramón Béjar, and Lluis Godo. A computational method for defeasible argumentation based on a recursive warrant semantics. In *Advances in Artificial Intelligence - IBERAMIA 2010*, volume 6433 of *Lecture Notes in Computer Science*, pages 40–49. Springer, 2010.

[3] Teresa Alsinet, Ramón Béjar, Lluis Godo, and Francesc Guitart. Maximal ideal recursive semantics for defeasible argumentation. In *Proceedings of the 5th International Conference on Scalable Uncertainty Management (SUM 2011)*, pages 96–109, 2011.

[4] Teresa Alsinet, Ramón Béjar, Lluis Godo, and Francesc Guitart. Using answer set programming for an scalable implementation of defeasible argumentation. In *Proceedings of Tools with Artificial Intelligence (ICTAI), 2012 24th IEEE International Conference on*, pages 1016–1021, 2012.

[5] Teresa Alsinet, Carlos I. Chesñevar, and Lluis Godo. A level-based approach to computing warranted arguments in possibilistic defeasible logic programming. In *Computational Models of Argument: Proceedings of COMMA 2008*, volume 172 of *Frontiers in Artificial Intelligence and Applications*, pages 1–12. IOS Press, 2008.

[6] Teresa Alsinet, Carlos I. Chesñevar, Lluis Godo, and Guillermo R. Simari. A logic programming framework for possibilistic argumentation: Formalization and logical properties. *Fuzzy Sets and Systems*, 159(10):1208–1228, 2008.

[7] Leila Amgoud. Postulates for logic-based argumentation systems. In *Proceedings of the ECAI-2012 Workshop WL4AI*, pages 59–67, 2012.

[8] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern sat solvers. In *Proceedings of the 21st international jont conference on Artifical intelligence*, IJCAI'09, pages 399–404, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[9] Philippe Besnard and Anthony Hunter. *Elements of Argumentation*. The MIT Press, 2008.

[10] Andrei Bondarenko, Phan Minh Dung, Robert A. Kowalski, and Francesca Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artif. Intell.*, 93:63–101, 1997.

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

[12] Laura A. Cecchi, Pablo R. Fillottrani, and Guillermo R. Simari. On the complexity of DeLP through game semantics. In *Proc. 11th Intl. Workshop on Nonmonotonic Reasoning (NMR 2006)*, pages 386–394, May 2006.

[13] Carlos I. Chesñevar, Ana G. Maguitman, and Ronald P. Loui. Logical Models of Argument. *ACM Computing Surveys*, 32(4):337–383, December 2000.

[14] Carlos I. Chesñevar, Guillermo R. Simari, and Lluis Godo. Computing dialectical trees efficiently in possibilistic defeasible logic programming. In *LPNMR*, pages 158–171, 2005.

[15] Phan Minh 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.

[16] Phan Minh Dung, Paolo Mancarella, and Francesca Toni. A dialectic procedure for sceptical, assumption-based argumentation. In *Computational Models of Argument: Proceedings of COMMA 2008*, volume 172 of *Frontiers in Artificial Intelligence and Applications*, pages 145–156. IOS Press, 2006.

[17] Phan Minh Dung, Paolo Mancarella, and Francesca Toni. Computing ideal sceptical argumentation. *Artif. Intell.*, 171(10-15):642–674, 2007.

[18] Niklas Eén and Niklas Sörensson. An extensible sat-solver. In *SAT*, volume 2919 of *LNCS*, pages 502–518. Springer, 2003.

[19] Alejandro J. García, Jürgen Dix, and Guillermo R. Simari. Argument-based logic programming. In Iyad Rahwan and Guillermo R. Simari, editors, *Argumentation in Artificial Intelligence*, chapter 8, pages 153–171. Springer, 2009.

[20] Alejandro J. García and Guilermo R. Simari. Defeasible Logic Programming: An Argumentative Approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2004.

[21] Henry A. Kautz and Bart Selman. Unifying sat-based and graph-based planning. In *IJCAI*, pages 318–325, 1999.

[22] John L. Pollock. A recursive semantics for defeasible reasoning. In Iyad Rahwan and Guillermo R. Simari, editors, *Argumentation in Artificial Intelligence*, chapter 9, pages 173–198. Springer, 2009.

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

[24] Henry Prakken and Gerard Vreeswijk. Logical Systems for Defeasible Argumentation. In D. Gabbay and F.Guenther, editors, *Handbook of Phil. Logic*, pages 219–318. Kluwer, 2002.

[25] Iyad Rahwan and Guillermo R. Simari, editors. *Argumentation in Artificial Intelligence*. Springer, 2009.

[26] Gerard Vreeswijk. Abstract argumentation systems. *Artif. Intell.*, 90(1-2):225–279, 1997.