Ann Math Artif Intell (2009) 57:181–204
DOI 10.1007/s10472-010-9184-z

The foundations of DeLP: defeating relations, games and truth values

Ignacio Darío Viglizzo · Fernando A. Tohmé · Guillermo R. Simari

Published online: 12 May 2010

Abstract In this paper we examine the mechanism of DeLP (Defeasible Logic Programming). We first study the definition of the *defeating* relation in a formal setting that allows us to uncover some hidden assumptions, and suggest an alternative definition. Then we introduce a game-theoretic characterization of the system. We obtain a new set of truth values arising from games in which arguments for and against a given literal are played out. We study how additional constraints define protocols of admissible attacks. The DeLP protocol ensures the finiteness of the games, and therefore the existence of winning strategies for the corresponding games. The *defeating* relation among arguments determines the strategies that will win and consequently the truth values of queries. We find that the DeLP protocol also excludes the warranting of a literal and its negation.

Keywords Argumentation protocol · Defeasible logic programming · Game theory · Truth values · Defeating relation

Mathematics Subject Classifications (2010) 68T27 · 91A80

I. D. Viglizzo · F. A. Tohmé
CONICET, Avda. Rivadavia 1917, CP C1033AAJ,
Cdad. de Buenos Aires, Argentina

I. D. Viglizzo (✉) · F. A. Tohmé · G. R. Simari
Universidad Nacional del Sur, Av. Alem 1253, B8000CPB,
Bahía Blanca, Argentina
e-mail: viglizzo@criba.edu.ar, viglizzo@gmail.com

F. A. Tohmé
e-mail: ftohme@criba.edu.ar

G. R. Simari
e-mail: grs@cs.uns.edu.ar

1 Introduction

The formalism of DeLP (Defeasible Logic Programming) [7] extends declarative programming [10] to capture features common to various systems of defeasible reasoning [4, 14, 16, 17, 19, 20, 23]. The inference mechanism of DeLP, upon a query about a literal, answers by indicating whether it is or not warranted by a program in which less than certain knowledge is represented [5].

From Logic Programming (LP), DeLP draws the formal characterization of programs as sets of rules, except that it considers two kinds of rules. Strict rules represent sound knowledge while defeasible rules, instead, represent less than certain knowledge that may be defeated by other information.

As said, DeLP works by querying the status of pieces of knowledge, represented by literals drawn from the program. The query of a literal l succeeds if l is supported by a warranted *argument*. Arguments are built using both strict and defeasible rules as well as *facts* (literals assumed to be true). The core element of the warrant procedure is the class of criteria for comparing two contradictory arguments. These criteria determine when an argument defeats others. While in implementations the chosen criteria are usually purely syntactic (e.g., specificity, see [20]) , the warrant procedure can be defined abstracting away the details of the relations among arguments.

The inference mechanism of DeLP generates all the arguments for and against a queried literal l. Then, the warrant procedure is applied to determine if an argument supporting l remains undefeated in a sequence of admissible attacks and defenses. If so, l is said to be *warranted* and the answer to the query is positive.

Several argumentation systems have been formulated in dialectical style [6, 12, 18–20]. The concept of dispute in most of them can be characterized as a so-called argument game. An argument game is a Śone-dimensionalŠ dispute in which each player may respond only once to each argument advanced by the opponent, and if that argument turns out to be ineffective, that player may not try a second reply to the same argument. Thus, no backtracking is allowed. This fact makes argument games into what is officially known as two-player zero-sum games, including the concepts that come with it, the most important of which is strategy.

The warrant procedure admits a natural game-theoretic interpretation. The comparison among arguments can be conceived as a contest between a *Proponent* and an *Opponent*. According to the winning strategies in the games for and against l, a gamut of truth-valuations can be obtained. Which truth value actually obtains depends on the properties of the defeating relation among arguments.

Without further constraints these games may be infinitely long, implying that the truth of some literals may remain undefined. Finite games, instead, are determined, i.e., have always a winning strategy for one of the players. But even in this case, the program can yield inconclusive answers. In more detail, four truth values may obtain. Two of them can be interpreted as positive or negative answers to queries. The other two correspond either to contradictory or ambiguous answers.

In any case, the finiteness of the games can be ensured by stipulating up front which sequences of for and against arguments are admissible. The admissibility constraints define a protocol for the games. The warrant mechanism of DeLP is based on a very natural protocol with just a few constraints.

The nature of the defeating relation among arguments, under the DeLP protocol for the games, defines which truth values obtain for any given literal. The analysis of this determination is the focus of this paper. The main result of this paper is that using this protocol, literals can be true, false or ambiguous, but never contradictory.

We proceed as follows. In Section 2 we present the DeLP formalism. In Section 3 we analyze the extension of an abstract pre-defeating relation to a *defeating* relation over the set of arguments of a given program. This analysis reveals some problems that lead us to pose some restrictions on the pre-defeating relation and suggest an alternative definition of defeaters. In Section 4 we rephrase the dialectical aspect of DeLP in game-theoretic terms. We focus on how the protocol of DeLP ensures the finiteness the games in Section 5. In Section 6 we establish the main result.

A preliminary report on this research has been published in [22].

2 The basics of DeLP

In order to discuss the formal properties of DeLP, we have to present the basics of this formalism.[1]

Each *Defeasible Logic Program* $\mathbb{P}$ is a finite set of facts, strict rules, and defeasible rules $\mathbb{P} = \langle \Pi, \Delta \rangle$, where Π denotes the set of facts and strict rules, while Δ denotes the set of defeasible rules. The set Π is the disjoint union of the sets Π_F of facts and Π_R of strict rules.

Facts and rules are defined in terms of atoms. While in the literature on DeLP, rules like $b(X) \leftarrow p(X)$ or $f(X) \prec b(X)$ are customary, they are no more than shorthand for a finite set of propositional rules. We follow the usual convention in Logic Programming (see [9]), using "schematic rules" with variables. More precisely, let At be the set of atoms that occur in a given program $\mathbb{P}$. Given a set $X \subseteq \mathsf{At}$ of atoms, $\sim X$ is the set $\{\sim x : x \in X\}$. Then, the set Lit is the set of all literals in $\mathsf{At} \cup \sim \mathsf{At}$. The complement $\bar{l}$ of a literal $l \in$ Lit is $\sim x$ if l is an atom x and x if l is a negated atom $\sim x$. This indicates the strong syntactic bent of DeLP.

Then, the main components of a program $\mathbb{P}$ are:

- Π_F *Facts*, which are literals, elements of Lit.
- Π_R *Strict Rules* of the form $l_0 \leftarrow l_1, \ldots, l_n$, where l_0 is the *head* and $\{l_i\}_{i>0}$ is the *body*. Each l_i in the body or the head is in Lit.
- Δ *Defeasible Rules* of the form $l_0 \prec l_1, \ldots, l_n$, where l_0 is the *head* and $\{l_i\}_{i>0}$ is the *body*. Again, each l_i in the body or the head is a literal.

Rules, both strict and defeasible act on facts, allowing the derivation of literals.

Definition 1 A *defeasible derivation* of l up from $X \subseteq$ Lit, $\mathcal{R} \subseteq \Pi_R$ and $\mathcal{A} \subseteq \Delta$ is a finite sequence $l_1, \ldots, l_n = l$ of literals in Lit such that each l_i is either in X or there exists a rule in $\mathcal{R} \cup \mathcal{A}$ with l_i as its head, and every literal b_j in its body is such that $b_j \in \{l_k\}_{k<i}$.

Definition 2 Given sets $X \subseteq$ Lit, $\mathcal{R} \subseteq \Pi_R$ and $\mathcal{A} \subseteq \Delta$, $C(X, \mathcal{R}, \mathcal{A})$ is the set of all literals defeasibly derivable from $X \cup \mathcal{R} \cup \mathcal{A}$. The set of *strict consequences* of $X \subseteq$

[1]We follow very closely the presentation in [7].

Lit is $C_s(X) = C(X, \Pi_R, \emptyset)$. Finally, we are going to use often $C(\mathcal{A}) = C(\Pi_F, \Pi_R, \mathcal{A})$ for sets $\mathcal{A} \subseteq \Delta$.

Definition 3 Given a set $X \subseteq \text{Lit}$, let $X^+ = X \cap \mathsf{At}$ and $X^- = \{a \in \mathsf{At} : \sim a \in X\}$. X is said to be *contradictory* if $X^+ \cap X^- \neq \emptyset$. A set of defeasible rules $\mathcal{A}$ is *contradictory* if $C(\mathcal{A})$ is contradictory.

We assume that for all programs, the set Π of facts and strict rules is not contradictory. A fundamental relation in DeLP is that of *disagreement* between literals:

Definition 4 Two literals $h, q \in \text{Lit}$ are said to *disagree* (for a given program $\mathbb{P}$) if $C_s(\Pi_F \cup \{h, q\}) = C(\Pi_F \cup \{h, q\}, \Pi_R, \emptyset)$ is contradictory. Since we are only interested in the disagreeing relation over the set $\text{Lit}_{\mathbb{P}}$ of the literals that appear in a program $\mathbb{P}$, we define a binary relation $D \subseteq \text{Lit}_{\mathbb{P}} \times \text{Lit}_{\mathbb{P}}$ to record which pairs of literals disagree. This relation is clearly symmetric.

This relation matters for the determination of attacks between arguments. In order to get to that, let us define the fundamental concept of argument. We will use $\mathcal{P}$ to denote the powerset construction.

Definition 5 An *argument* is a pair $\langle \mathcal{A}, h \rangle \in \mathcal{P}(\Delta) \times \text{Lit}$ that satisfies:

1. $h \in C(\mathcal{A})$
2. $\mathcal{A}$ is not contradictory.
3. If $h \in C(\mathcal{A}')$, then $\mathcal{A}' \not\subset \mathcal{A}$, that is, $\mathcal{A}'$ is not a proper subset of $\mathcal{A}$.

Definition 6 Let $\text{Arg}(\mathbb{P})$ be the set of all arguments of a program $\mathbb{P}$. The argument $\langle \mathcal{A}_1, h \rangle$ is a subargument of $\langle \mathcal{A}_2, h' \rangle$ iff $\mathcal{A}_1 \subseteq \mathcal{A}_2$.

The subargument relation is a preorder over $\text{Arg}(\mathbb{P})$. Furthermore, if $\langle \mathcal{A}_1, h \rangle$ is a subargument of $\langle \mathcal{A}_2, h' \rangle$ and $\langle \mathcal{A}_2, h' \rangle$ is a subargument of $\langle \mathcal{A}_1, h \rangle$, then $\mathcal{A}_1 = \mathcal{A}_2$. This means that if we identify arguments with the same first component, the relation is simply a restriction of the inclusion relation over $\mathcal{P}(\Delta)$.

Note that for the empty set, we have all the arguments of the form $\langle \emptyset, l \rangle$ with $l \in \Pi_F$.

Definition 7 For each literal h we define the binary relation R_h on $\text{Arg}(\mathbb{P})$ by $\langle \mathcal{A}_1, h_1 \rangle R_h \langle \mathcal{A}_2, h_2 \rangle$ iff there exists $\langle \mathcal{A}, h \rangle \in \text{Arg}(\mathbb{P})$ such that $\mathcal{A} \subseteq \mathcal{A}_1$ and $(h, h_2) \in D$. We say in this case that the argument $\langle \mathcal{A}_1, h_1 \rangle$ is attacked by $\langle \mathcal{A}_2, h_2 \rangle$ at h or that $\langle \mathcal{A}_2, h_2 \rangle$ *attacks* or *rebutts* $\langle \mathcal{A}_1, h_1 \rangle$ at h. We also say that $\langle \mathcal{A}_2, h_2 \rangle$ is a *counter-argument* of $\langle \mathcal{A}_1, h_1 \rangle$.

We have the following easy consequences of the definition:

Proposition 1

1. *If* $\langle \mathcal{A}_1, h_1 \rangle R_h \langle \mathcal{A}_2, h_2 \rangle$, *then for some* $\mathcal{A} \subseteq \mathcal{A}_1$, $\langle \mathcal{A}, h \rangle$ *is an argument and* $\langle \mathcal{A}_2, h_2 \rangle R_{h_2} \langle \mathcal{A}, h \rangle$ *(sub-symmetry).*

2. *If $\langle \mathcal{A}, l\rangle R_l \langle \mathcal{B}, p\rangle$, then $\langle \mathcal{B}, p\rangle R_p \langle \mathcal{A}, l\rangle$.*
3. *If $q \in C_s(\Pi_F)$, then $\langle \emptyset, q\rangle$ is an argument and it has no counter-arguments.*
4. *Furthermore, an argument $\langle \emptyset, q\rangle$ cannot be a counterargument of any argument.*

Proof

1. From the definition of R_h, we know that for some $\langle \mathcal{A}, h\rangle \in \text{Arg}(\mathbb{P})$ we have that $\mathcal{A} \subseteq \mathcal{A}_1$ and $(h, h_2) \in D$. Therefore, there exists $\mathcal{A}_2 \subseteq \mathcal{A}_2$ and $(h, h_2) \in D$ so we can claim that $\langle \mathcal{A}_2, h_2\rangle R_h \langle \mathcal{A}, h\rangle$.
2. Using the definition of R_l, this simply means that l and p disagree, and this is enough to justify that $\langle \mathcal{B}, p\rangle R_p \langle \mathcal{A}, l\rangle$, since the arguments are attacked precisely at the literals they support, so no subarguments need be considered. Recall also the minimality of the sets that form the argument.
3. It is easy to check that since $q \in C(\emptyset)$ while $\emptyset$ is not contradictory and has no proper subset, so $\langle \emptyset, q\rangle$ is an argument. Now assume it has a counterargument $\langle \mathcal{A}_2, h'\rangle$ that attacks $\langle \emptyset, q\rangle$ at h. Then there exists an argument $\langle \mathcal{A}, h\rangle$ with $\mathcal{A} \subseteq \emptyset$, so $h \in C(\emptyset)$, and $(h, h') \in D$. This means that $C_s(\Pi_F \cup \{h, h'\})$ is contradictory, but since $C(\emptyset) \subseteq C(\mathcal{A}_2)$, $\{h, h'\} \subseteq C(\mathcal{A}_2)$, so $\mathcal{A}_2$ is contradictory, and therefore $\langle \mathcal{A}_2, h'\rangle$ cannot be an argument.
4. If $\langle \mathcal{A}, h\rangle$ is an argument and $\langle \mathcal{A}, h\rangle R_p \langle \emptyset, q\rangle$, then there is an argument $\langle \mathcal{A}', p\rangle$ with $(p, q) \in D$ but we have seen in the previous proof that this contradicts the fact that $\mathcal{A}'$ is not contradictory. □

Now we have the set Arg($\mathbb{P}$) with the relations R_h over it. We want to have a method to decide, given a literal l, whether it is supported by the program $\mathbb{P}$ or not. Clearly we want all literals in $C(\emptyset)$ to be supported or warranted. Which other literals should be supported? If there is a defeasible derivation of a literal l while $\bar{l}$ is not derivable, we want l to be warranted as well.

But what about the cases in which the derivation of l yields a contradictory set of literals, or there are arguments that support the complement of l as well?

We have seen that if the relation R_h holds between two arguments, there is also some attack on the attacking argument (Proposition 1, 1). We need to be able to tell which of these two arguments (if any) 'wins' the discussion. For this, we assume the existence of a binary relation $\prec$ contained in $R = \bigcup_{h \in \text{Lit}} R_h$ (it will be of no consequence for us if $\prec$ holds in other cases, so we may as well just concentrate on subsets of R). We will call this relation *pre-defeating* . The notation for this relation is somewhat misleading since we do not assume that $\prec$ is a partial order or even a preorder. It will just be an arbitrary way of deciding whether one of two arguments, is stronger than the other. This information will be propagated to some other arguments through the following definitions.

Definition 8 $\langle \mathcal{A}_1, h_1\rangle$ is a *proper defeater* of $\langle \mathcal{A}_2, h_2\rangle$ (at literal h) if $\langle \mathcal{A}_2, h_2\rangle R_h \langle \mathcal{A}_1, h_1\rangle$ and there exists a sub-argument $\langle \mathcal{A}, h\rangle$ of $\langle \mathcal{A}_2, h_2\rangle$ such that $\langle \mathcal{A}, h\rangle \prec \langle \mathcal{A}_1, h_1\rangle$. We denote this by $\langle \mathcal{A}_2, h_2\rangle < \langle \mathcal{A}_1, h_1\rangle$.

$\langle \mathcal{A}_1, h_1\rangle$ is a *blocking defeater* of $\langle \mathcal{A}_2, h_2\rangle$ (at literal h) if $\langle \mathcal{A}_2, h_2\rangle R_h \langle \mathcal{A}_1, h_1\rangle$ and there exists a sub-argument $\langle \mathcal{A}, h\rangle$ of $\langle \mathcal{A}_2, h_2\rangle$ such that $\langle \mathcal{A}, h\rangle \not\prec \langle \mathcal{A}_1, h_1\rangle$ and $\langle \mathcal{A}_1, h_1\rangle \not\prec \langle \mathcal{A}, h\rangle$. Whenever this is the case we use the notation $\langle \mathcal{A}_2, h_2\rangle \approx \langle \mathcal{A}_1, h_1\rangle$.

A *defeater* of an argument is either a proper defeater or a blocking defeater. If $\langle \mathcal{A}_2, h_2\rangle$ is a defeater of $\langle \mathcal{A}_1, h_1\rangle$ we write $\langle \mathcal{A}_1, h_1\rangle \leq \langle \mathcal{A}_2, h_2\rangle$.

Equivalently, we may clean up the original definition by saying that $\langle \mathcal{A}_1, h_1\rangle$ is a *proper defeater* of $\langle \mathcal{A}_2, h_2\rangle$ (at literal h) if there exists a sub-argument $\langle \mathcal{A}, h\rangle$ of $\langle \mathcal{A}_2, h_2\rangle$ such that $(h, h_2) \in D$ and $\langle \mathcal{A}, h\rangle \prec \langle \mathcal{A}_1, h_1\rangle$. Similarly, $\langle \mathcal{A}_2, h_2\rangle \approx \langle \mathcal{A}_1, h_1\rangle$ if there exists a sub-argument $\langle \mathcal{A}, h\rangle$ of $\langle \mathcal{A}_2, h_2\rangle$ such that $(h, h_2) \in D$, $\langle \mathcal{A}, h\rangle \not\prec \langle \mathcal{A}_1, h_1\rangle$ and $\langle \mathcal{A}_1, h_1\rangle \not\prec \langle \mathcal{A}, h\rangle$.

From the definition, the following proposition is immediate:

Proposition 2 *The* defeating *relation $\leq$ is a subset of R. Furthermore, given two arguments such that $\langle \mathcal{A}_1, h_1\rangle R_h \langle \mathcal{A}_2, h_2\rangle$, if $\langle \mathcal{A}_2, h_2\rangle$ is not a defeater of $\langle \mathcal{A}_1, h_1\rangle$, then all the subarguments $\langle \mathcal{A}, h\rangle$ of $\langle \mathcal{A}_1, h_1\rangle$ that attack $\langle \mathcal{A}_2, h_2\rangle$ (and there must exist at least one of these) are proper defeaters of $\langle \mathcal{A}_2, h_2\rangle$.*

We must keep in mind that $\approx$ is not necessarily a symmetric relation but it is *sub-symmetric*: If $\langle \mathcal{A}_1, h_1\rangle \approx \langle \mathcal{A}_2, h_2\rangle$ then there exists $\langle \mathcal{A}, h\rangle \subseteq \langle \mathcal{A}_1, h_1\rangle$ such that $\langle \mathcal{A}_2, h_2\rangle \approx \langle \mathcal{A}, h\rangle$.

As particular cases, when the relation $\prec$ is empty, all attacking arguments are blocking defeaters, so $\leq = \approx = R$. If the pre-defeating relation is the whole of R, then all attacking arguments are proper defeaters and $\leq = < = R$.

3 Examples and problems with the definitions

Example 1 Let us consider a simple program $\mathbb{P}_1$ with $\Pi_F = \{b, c\}$; $\Pi_R = \{d \leftarrow a\}$ and $\Delta = \{a \mathrel{-\!\!\prec} b, \sim a \mathrel{-\!\!\prec} c\}$.

We have $C(\{b\}, \emptyset, \{a \mathrel{-\!\!\prec} b\}) = \{a, b\}$; $C_s(\{a\}) = \{a, d\}$; $C_s(\Pi_F) = \Pi_F$ and $C(\Delta) = \{a, b, c, d, \sim a\}$, a contradictory set.

Let $\mathcal{A}_1 = \{a \mathrel{-\!\!\prec} b\}$ and $\mathcal{A}_2 = \{\sim a \mathrel{-\!\!\prec} c\}$. Then the arguments are:

$$\mathrm{Arg}(\mathbb{P}_1) = \{\langle \emptyset, b\rangle, \langle \emptyset, c\rangle, \langle \mathcal{A}_1, a\rangle, \langle \mathcal{A}_1, d\rangle, \langle \mathcal{A}_2, \sim a\rangle\}.$$

The *disagreeing relation D* is formed by the set of all pairs of the form $(x, \bar{x})$ where $x \in \mathrm{Lit}$, together with the pairs $(a, \sim d)$ and $(\sim d, a)$. The literal $\sim d$, however, does not appear in the program, so we may as well just consider $D = \{(a, \sim a), (\sim a, a)\}$.

Note that in this example, the arguments $\langle \mathcal{A}_1, d\rangle$ and $\langle \mathcal{A}_1, a\rangle$ are subarguments of each other.

The attack relations that hold between arguments are: $\langle \mathcal{A}_1, a\rangle R_a \langle \mathcal{A}_2, \sim a\rangle$, $\langle \mathcal{A}_1, d\rangle R_a \langle \mathcal{A}_2, \sim a\rangle$ and $\langle \mathcal{A}_2, \sim a\rangle R_{\sim a} \langle \mathcal{A}_1, a\rangle$.

The attacking relations can be pictured by the following graph, where the letters on the arrows indicate the literals at which the attacks occur:

$$\begin{tikzcd}
\langle \mathcal{A}_1, a\rangle \arrow[r, shift right, "\sim a"'] & \langle \mathcal{A}_2, \sim a\rangle \arrow[l, shift right, "a"'] \arrow[r, "a"'] & \langle \mathcal{A}_1, d\rangle
\end{tikzcd}$$

Example 2 We observe in the program of the previous example that no defeating relation can compare the arguments $\langle \mathcal{A}_1, a\rangle$ and $\langle \mathcal{A}_1, d\rangle$. So if we have a pre-defeating relation $\prec_1$ such that $\langle \mathcal{A}_1, a\rangle \prec_1 \langle \mathcal{A}_2, \sim a\rangle$, it follows that the induced defeating relation has $\langle \mathcal{A}_1, a\rangle <_1 \langle \mathcal{A}_2, \sim a\rangle$ and $\langle \mathcal{A}_1, d\rangle <_1 \langle \mathcal{A}_2, \sim a\rangle$, while $\approx_1$ is empty, that is, there are no blocking defeaters for this pre-defeating relation.

Example 3 A second pre-defeating relation on the arguments of $\mathbb{P}_1$ above given by $\langle \mathcal{A}_1, d\rangle \prec_2 \langle \mathcal{A}_2, \sim a\rangle$ shows that the complexity of the definition can yield unexpected behaviors. In this case we get that all attackers are in fact blocking defeaters, that is, $R = \leq_2 = \approx_2$. In particular, $\langle \mathcal{A}_2, \sim a\rangle$ is a blocking defeater of $\langle \mathcal{A}_1, d\rangle$ and not a proper one as one may have expected.

In the literature about DeLP, the pre-defeating relation considered has been the *specificity* relation. If we want to abstract it away from that particular instance, there are some properties we may demand from it.

As we noted before, the defeating relation induced by $\prec$ is a subset of R, and this puts certain constraints on the properties one can expect from the relation. For example, it is unreasonable to expect $\prec$ to be transitive, since R is not.

The pre-defeating relation should be asymmetric, as the following example illustrates:

Example 4 Using the program $\mathbb{P}_1$ from the previous example let $\prec_3$ be equal to the relation R. Then we have that all attackers are proper defeaters. In particular, $\langle \mathcal{A}_1, a\rangle$ is a proper defeater of $\langle \mathcal{A}_2, \sim a\rangle$ and $\langle \mathcal{A}_2, \sim a\rangle$ is a proper defeater of $\langle \mathcal{A}_1, a\rangle$. This contradicts the intuition one has about what a proper defeater should be: an argument that is unequivocally better than the other.

While the asymmetry of the pre-defeating relation is necessary, it is not sufficient to guarantee that two arguments are not proper defeaters of each other:

Example 5 Consider the program $\mathbb{P}_2 : \Pi_F = \{a\}$, $\Delta = \{\sim y \mathrel{-\!\!\prec} a,\ x \mathrel{-\!\!\prec} \sim y,\ \sim x \mathrel{-\!\!\prec} a,\ y \mathrel{-\!\!\prec} \sim x\}$ with $\mathcal{A} = \{\sim y \mathrel{-\!\!\prec} a, x \mathrel{-\!\!\prec} \sim y\}$, $\mathcal{B} = \{\sim x \mathrel{-\!\!\prec} a, y \mathrel{-\!\!\prec} \sim x\}$, $\mathcal{C} = \{\sim y \mathrel{-\!\!\prec} a\}$, $\mathcal{D} = \{\sim x \mathrel{-\!\!\prec} a\}$ and arguments $\langle \emptyset, a\rangle$, $\langle \mathcal{A}, x\rangle$, $\langle \mathcal{B}, y\rangle$, $\langle \mathcal{C}, \sim y\rangle$, $\langle \mathcal{D}, \sim x\rangle$.

By letting $\langle \mathcal{D}, \sim x\rangle \prec \langle \mathcal{A}, x\rangle$ and $\langle \mathcal{C}, \sim y\rangle \prec \langle \mathcal{B}, y\rangle$, we get that $\langle \mathcal{A}, x\rangle$ is a proper defeater of $\langle \mathcal{B}, y\rangle$ and vice-versa.

Example 6 Consider the program $\mathbb{P}_3$ where $\Pi_F = \{b, c\}$, $\Pi_R = \{d \leftarrow a\}$ and $\Delta = \{a \mathrel{-\!\!\prec} b, \sim d \mathrel{-\!\!\prec} c\}$.

The difference with program $\mathbb{P}_1$ is subtle but has many consequences. The disagreeing relation is now $D = \{(d, \sim d), (\sim d, d), (a, \sim d), (\sim d, a)\}$. We have $\mathrm{Arg}(\mathbb{P}_2) = \{\langle \emptyset, b\rangle, \langle \emptyset, c\rangle, \langle \mathcal{A}, a\rangle, \langle \mathcal{A}, d\rangle, \langle \mathcal{B}, \sim d\rangle\}$, where $\mathcal{A} = \{a \mathrel{-\!\!\prec} b\}$ and $\mathcal{B} = \{\sim d \mathrel{-\!\!\prec} c\}$.

The attack relations are given in the following graph:

$$\begin{tikzcd}
\langle \mathcal{A}, d\rangle \arrow[r, shift right, "\sim d"'] & \langle \mathcal{B}, \sim d\rangle \arrow[l, shift right, "a{,}d"'] \arrow[r, shift left, "a{,}d"] & \langle \mathcal{A}, a\rangle \arrow[l, shift left, "\sim d"]
\end{tikzcd}$$

If we have as pre-defeating relation $\langle \mathcal{A}, d\rangle \prec \langle \mathcal{B}, \sim d\rangle$, according to the Definition 8, $\langle \mathcal{B}, \sim d\rangle$ is a proper defeater of $\langle \mathcal{A}, d\rangle$, but since $\langle \mathcal{A}, d\rangle$ has $\langle \mathcal{A}, a\rangle$ as a subargument, $\langle \mathcal{A}, a\rangle \not\prec \langle \mathcal{B}, \sim d\rangle$ and $\langle \mathcal{A}, a\rangle \not\succ \langle \mathcal{B}, \sim d\rangle$, this is also a blocking attack.

If one considers that an argument can only be attacked when it uses *defeasible* rules, it is a natural idea to regard only these when comparing the relative strength of two arguments. This idea is captured through the following definition:

Definition 9 Two arguments $\langle\mathcal{A}, h\rangle$ and $\langle\mathcal{B}, q\rangle$ are *equi-defeasible* iff $\mathcal{A} = \mathcal{B}$. In other words, if the defeasible rules used in the derivations of the literals h and q are the same.

We may now require that the pre-defeating relation be compatible with the equivalence relation of equi-defeasibility. In other words:

Definition 10 A pre-defeating relation $\prec$ is compatible with the equi-defeasibility relation if for all arguments such that $\langle\mathcal{A}, h\rangle \prec \langle\mathcal{B}, q\rangle$ and for all literals h', q' such that $\langle\mathcal{A}, h'\rangle$ and $\langle\mathcal{B}, q'\rangle$ are arguments, $\langle\mathcal{A}, h'\rangle \prec \langle\mathcal{B}, q'\rangle$ also holds.

Example 7 The pre-defeating relation $\prec_2$ of Example 3 is not compatible with equi-defeasibility. We put instead a relation $\prec_4$ over the arguments of $\mathbb{P}_1$ that is compatible. So we have that $\langle\mathcal{A}_1, d\rangle \prec_4 \langle\mathcal{A}_2, \sim a\rangle$ implies that $\langle\mathcal{A}_1, a\rangle \prec_4 \langle\mathcal{A}_2, \sim a\rangle$, and $\langle\mathcal{A}_2, \sim a\rangle$ is a proper defeater of $\langle\mathcal{A}_1, d\rangle$ as one originally expected. Observe also that we may summarize $\prec_4$ by just indicating that $\mathcal{A}_1 \prec_4 \mathcal{A}_2$.

Example 8 Taking a compatible pre-defeating relation such that $\mathcal{A} \prec \mathcal{B}$ in Example 6 the problem of having a defeater that is blocking and proper at the same time also disappears so that $\langle\mathcal{B}, \sim d\rangle$ is a proper defeater of $\langle\mathcal{A}, d\rangle$, and not a blocking one.

We can see, however that compatible pre-defeating relations do not preclude all the overlapping between blocking and proper defeaters.

Example 9 Let $\mathbb{P}_4$ be the program with facts set $\{a\}$, strict rules $\Pi_R = \{h \leftarrow b, h \leftarrow c, h_1 \leftarrow b, c\}$, while $\Delta = \{b \prec a, c \prec a, \sim h \prec a\}$. Here we have that the argument $\langle\{\sim h \prec a\}, \sim h\rangle$ attacks $\langle\{b \prec a, c \prec a\}, h_1\rangle$ at literal h, but we have two subarguments at which the attack can happen: $\langle\{b \prec a\}, h\rangle$ and $\langle\{c \prec a\}, h\rangle$. If we let $\langle\{b \prec a\}, b\rangle \prec \langle\{\sim h \prec a\}, \sim h\rangle$ then we obtain:

$$\begin{array}{rr}
\langle\{b \prec a\}, b\rangle < \langle\{\sim h \prec a\}, \sim h\rangle & \langle\{c \prec a\}, c\rangle \approx \langle\{\sim h \prec a\}, \sim h\rangle \\
\langle\{b \prec a\}, h\rangle < \langle\{\sim h \prec a\}, \sim h\rangle & \langle\{c \prec a\}, h\rangle \approx \langle\{\sim h \prec a\}, \sim h\rangle \\
\langle\{b \prec a, c \prec a\}, h_1\rangle < \langle\{\sim h \prec a\}, \sim h\rangle & \langle\{b \prec a, c \prec a\}, h_1\rangle \approx \langle\{\sim h \prec a\}, \sim h\rangle
\end{array}$$

So in this case $\langle\{\sim h \prec a\}, \sim h\rangle$ is both a proper and a blocking defeater for the same argument and at the same literal.

It may be desirable to have notions of proper and blocking defeaters that do not overlap. A blocking defeater should be an attack for which there is no information on whether it is stronger or weaker than the attacked argument. In an attempt to capture this intuition we re-define:

Definition 11 An argument $\langle\mathcal{A}, h\rangle$ is a *defeater* of $\langle\mathcal{B}, q\rangle$ if there exists an argument $\langle\mathcal{C}, z\rangle \sqsubseteq \langle\mathcal{B}, q\rangle$ such that $(z, h) \in D$ and for all $\langle\mathcal{D}, x\rangle \sqsubseteq \langle\mathcal{A}, h\rangle$, $\langle\mathcal{D}, x\rangle \not\prec \langle\mathcal{B}, q\rangle$.

An argument $\langle \mathcal{A}, h \rangle$ is a *proper defeater* of $\langle \mathcal{B}, q \rangle$ if it is a defeater of $\langle \mathcal{B}, q \rangle$ and there exists $\langle \mathcal{C}, z \rangle \subseteq \langle \mathcal{B}, q \rangle$ such that $\langle \mathcal{C}, z \rangle \prec \langle \mathcal{A}, h \rangle$.

An argument $\langle \mathcal{A}, h \rangle$ is a *blocking defeater* of $\langle \mathcal{B}, q \rangle$ if it is a defeater of $\langle \mathcal{B}, q \rangle$ and for all $\langle \mathcal{C}, z \rangle \subseteq \langle \mathcal{B}, q \rangle$, $\langle \mathcal{C}, z \rangle \not\prec \langle \mathcal{A}, h \rangle$.

The following proposition follows immediately from the definition above:

Proposition 3 *Under Definition* 11*, the* proper defeater *relation is asymmetric and no argument can be both a proper and blocking defeater of another argument.*

Example 10 Taking the program $\mathbb{P}_2$ from Example 5 and using Definition 11, the arguments $\langle \mathcal{A}, x \rangle$ and $\langle \mathcal{B}, y \rangle$ do not defeat each other. On the other hand, we still have that $\langle \mathcal{A}, x \rangle$ is a proper defeater of $\langle \mathcal{D}, \sim x \rangle$ and $\langle \mathcal{B}, y \rangle$ is a proper defeater of $\langle \mathcal{C}, \sim y \rangle$.

Example 11 If we examine $\mathbb{P}_4$ from Example 9 using Definition 11, we get that $\langle \{\sim h \prec a\}, \sim h \rangle$ is a proper defeater of $\langle \{b \prec a, c \prec a\}, h_1 \rangle$ and not a blocking one.

Under the new definition, the relation $\approx$ is not necessarily sub-symmetric, but the following is true.

Proposition 4 *If $\langle \mathcal{A}, h \rangle$ is a blocking defeater of $\langle \mathcal{B}, q \rangle$ as in Definition* 11*, then there is a subargument $\langle \mathcal{C}, z \rangle$ of $\langle \mathcal{B}, q \rangle$ such that $\langle \mathcal{C}, z \rangle$ is a defeater of $\langle \mathcal{A}, h \rangle$.*

Proof Since $\langle \mathcal{A}, h \rangle$ is a blocking defeater of $\langle \mathcal{B}, q \rangle$, there exists a subargument $\langle \mathcal{C}, z \rangle$ of $\langle \mathcal{B}, q \rangle$ such that $(z, h) \in D$, This is the argument that we want to prove is a defeater of $\langle \mathcal{A}, h \rangle$.

For the first condition, we take $\langle \mathcal{A}, h \rangle$ itself as the subargument of $\langle \mathcal{A}, h \rangle$ and we know already that $(h, z) \in D$. Now if $\langle \mathcal{D}, x \rangle \subseteq \langle \mathcal{C}, z \rangle$, then we also have that $\langle \mathcal{D}, x \rangle \subseteq \langle \mathcal{B}, q \rangle$, so, since $\langle \mathcal{A}, h \rangle$ is a *blocking* defeater, $\langle \mathcal{D}, x \rangle \not\prec \langle \mathcal{A}, h \rangle$. □

One drawback of Definition 11 is its computational cost, which is greater than that of Definition 8. One would need to weigh the advantages the suggested definition provides against its cost before adopting it. The results in the following sections are independent from which definition of proper and blocking defeaters are used. The original definitions are used unless otherwise stated.

4 Warrant games

Given a literal l we want to find out what the program $\mathbb{P}$ has to say about it. What if there are more than one argument supporting l? What if one of them is undefeated and the other is defeated? We will answer these questions through a slightly generalized version of the mechanism of warrant of DeLP [7], using the language of game theory, which turns out to be quite natural for framing the dialectical process that leads to the warrant of a literal.

First, we will introduce the definition of extensive games with perfect information. Then we will define *warrant games*, a class of games tailored to our needs, and finally we will show how to use them for answering queries to the program.

Definition 12 (following [15]) An *extensive game with perfect information* $G = \langle N, H, P, (U_i)_{i \in N} \rangle$ consists of:

- A set N, the set of *players*.
- A set H of sequences (finite or infinite) that satisfies the following three properties:
  - The empty sequence $\emptyset$ is in H.
  - If $(a_k)_{k=1,\ldots,K} \in H$ (where K may be infinite) and $L < K$ then $(a_k)_{k=1,\ldots,L} \in H$.
  - If an infinite sequence $(a_k)_{k=1}^{\infty}$ satisfies $(a_k)_{k=1,\ldots,L} \in H$ for every positive integer L, then $(a_k)_{k=1}^{\infty} \in H$.

  The members of H are called *histories*. Each component a_k of a history is an *action* taken by a player. A history $(a_k)_{k=1,\ldots,K} \in H$ is *terminal* if it is infinite or there is no a_{K+1} such that $(a_k)_{k=1,\ldots,K+1} \in H$. The set of terminal histories is denoted with Z.
- A function $P : H \setminus Z \to N$, that indicates for each history in H which one of the players takes an action after the history.
- Utility functions $U_i : Z \to \mathbb{R}$ for $i \in N$ that give for each terminal history and each player, the *payoff* of that player after that history.

The set H can be seen as a tree with root $\emptyset$, with its nodes labeled by the function P, and the leaves labeled by the functions U_i, $i \in N$. We identify the elements a_k with edges of the tree. Therefore, each particular branch from the root is a history, in which the edges are the consecutive actions chosen by the players. We call the game *finite* if H is finite. After any nonterminal history y player $P(y)$ chooses an action from the set $A(y) = \{a : y \text{ followed by } a \text{ is in } H\}$.

A *warrant game* for a literal l is an extensive game with perfect information with two players. We will call these two players *Proponent* and *Opponent*. We define the game as follows:

- $P(\emptyset) = Proponent$.
- The actions that the proponent *Proponent* can take at the root of the tree are all the arguments of the form $\langle \mathcal{A}, l \rangle$.
- The actions after a nonterminal history y are the arguments $\langle \mathcal{A}', q \rangle$ such that $\langle \mathcal{A}, p \rangle \preceq \langle \mathcal{A}', q \rangle$, where $\langle \mathcal{A}, p \rangle$ is the last component in y. In this case, $P(y) = Proponent$ if y has even length and $P(y) = Opponent$ if the length is odd.
- The utility for the *Proponent* assumes the value 1(win) at a history $y \in Z$ if the length of y is odd, and -1 otherwise. The utility for the *Opponent* is -1 times the utility of the *Proponent*.

If we have that the defeating relation $\leq$ is simply the union of all the attack relations $R = \bigcup_{h \in \mathrm{Lit}} R_h$, and a literal l has some argument which can be attacked, then we will have an infinite tree (see Proposition 1).

Example 12 Let $\mathbb{P}_5$ be the program consisting of $\Pi_F = \{c, d\}$; $\Delta = \{b \prec c,\ a \prec b,\ \sim b \prec d,\ \sim a \prec d\}$. Letting $\mathcal{A}_1 = \{b \prec c, a \prec b\}$, $\mathcal{A}_2 = \{\sim b \prec d\}$, $\mathcal{A}_3 = \{b \prec c\}$ and $\mathcal{A}_4 = \{\sim a \prec d\}$, the arguments are $\langle \mathcal{A}_1, a \rangle$, $\langle \mathcal{A}_2, \sim b \rangle$, $\langle \mathcal{A}_3, b \rangle$ and $\langle \mathcal{A}_4, \sim a \rangle$, while the attacking relations can be read in the graph:

$$\begin{tikzcd}
\langle \mathcal{A}_3, b \rangle \arrow[r, shift right, "\sim b"'] & \langle \mathcal{A}_2, \sim b \rangle \arrow[l, shift right, "b"'] \arrow[r, "b"'] & \langle \mathcal{A}_1, a \rangle \arrow[r, shift right, "\sim a"'] & \langle \mathcal{A}_4, \sim a \rangle \arrow[l, shift right, "a"']
\end{tikzcd}$$

If we assume a pre-defeating relation such that $\langle \mathcal{A}_3, b \rangle \prec_1 \langle \mathcal{A}_2, \sim b \rangle$ then we obtain the defeating relation:

$$\begin{array}{ll}
\langle \mathcal{A}_1, a \rangle <_1 \langle \mathcal{A}_2, \sim b \rangle & \langle \mathcal{A}_1, a \rangle \approx_1 \langle \mathcal{A}_4, \sim a \rangle \\
\langle \mathcal{A}_3, b \rangle <_1 \langle \mathcal{A}_2, \sim b \rangle & \langle \mathcal{A}_4, \sim a \rangle \approx_1 \langle \mathcal{A}_1, a \rangle
\end{array}$$

We build the warrant game for the literal a in the program $\mathbb{P}_5$:

$$\begin{tikzcd}
 & \emptyset^P \arrow[d, dash] & & \\
 & \langle \mathcal{A}_1, a \rangle^O \arrow[dl, dash] \arrow[dr, dash] & & \\
\langle \mathcal{A}_2, \sim b \rangle & & \langle \mathcal{A}_4, \sim a \rangle^P \arrow[d, dash] & \\
(-1,1) & & \langle \mathcal{A}_1, a \rangle^O \arrow[dl, dash] \arrow[dr, dash] & \\
 & \langle \mathcal{A}_2, \sim b \rangle & & \langle \mathcal{A}_4, \sim a \rangle^P \\
 & (-1,1) & & \vdots
\end{tikzcd}$$

Notice that in the diagram we indicate the nodes by a single argument. The histories can be reconstructed by tracing the path in the tree up to the root. The superscript P or O on each node indicates the player who moves next. Below the terminal histories we have written the payoffs for the players *Proponent* and *Opponent*, respectively.

If we consider instead the relation $\prec_2 = \emptyset$ or the $\prec_3 = R$, we would have the infinite game tree:

$$\begin{tikzcd}
 & \emptyset^P \arrow[d, no head] & & \\
 & \langle \mathcal{A}_1, a\rangle^O \arrow[dl, no head] \arrow[dr, no head] & & \\
\langle \mathcal{A}_2, \sim b\rangle^P \arrow[d, no head] & & \langle \mathcal{A}_4, \sim a\rangle^P \arrow[d, no head] & \\
\langle \mathcal{A}_3, b\rangle^O \arrow[d, no head] & & \langle \mathcal{A}_1, a\rangle^O \arrow[dl, no head] \arrow[dr, no head] & \\
\langle \mathcal{A}_2, \sim b\rangle^P \arrow[d, dotted, no head] & \langle \mathcal{A}_2, \sim b\rangle^P \arrow[d, dotted, no head] & & \langle \mathcal{A}_4, \sim a\rangle^P \arrow[d, dotted, no head] \\
{} & {} & & {}
\end{tikzcd}$$

A plan for a given player, in which they have a response for every possible contingency of the game is called a *strategy*:

Definition 13 [15] A *strategy* for a player $i \in N$ in an extensive game with perfect information $\langle N, H, P, U_n\rangle$ is a function that assigns an action in $A(y)$ to each nonterminal history $y \in H \setminus Z$ for which $P(y) = i$.

Since players are rational, they will seek to get the highest utility. In order to do that, each one will seek the best possible strategy. The joint strategy profiles of all players yield a single history. In the case of warrant games, since each terminal history pays either 1 or -1, we can in principle define a *winning strategy* for a player:

Definition 14 A *winning strategy* for one of the players in a warrant game is a strategy that yields a terminal history $z \in Z$ such that their utility is 1, no matter what the other player's actions are.

It turns out that in some warrant games (for example those without terminal histories), none of the players has a winning strategy.

Now we pose a *query* to $\mathbb{P}$. The query is a literal l. Then we analyze two associated games. In the first place, we look at the warrant game for the literal l, and then the warrant game for the complement literal $\bar{l}$ in which the *Proponent* and *Opponent* change their roles. That is, the *Opponent* starts the game by choosing an argument for $\bar{l}$.

This approach seems to differ from the formalism presented in [7], where the existence of a winning strategy for the proponent of a literal l is enough to yield the answer "yes" to a query . On the other hand, in that paper, to yield the answer "no" to l, the status of $\bar{l}$ is analyzed, which clearly suggests the road we have taken here. Furthermore, we will show (Proposition 7 and Theorem 1) that under the further constraints imposed by DeLP, the mechanisms are equivalent.

The following table summarizes the possible outcomes (meaning which of the players has a winning strategy) of both games and how they jointly yield an answer to the query:

Warrant game for l	Warrant game for $\bar{l}$	Answer to the query
Proponent	*Proponent*	YES
Proponent	*Opponent*	undecided
Proponent	None	yes
Opponent	*Proponent*	undecided
Opponent	*Opponent*	NO
Opponent	None	no
None	*Proponent*	yes
None	*Opponent*	no
None	None	undecided

Thus we have a system in which a query for a literal l can lead to different outcomes which can be displayed as a partially ordered set:

$$\begin{tikzcd}
 & & PP \arrow[dl, no head] \arrow[dr, no head] & & \\
 & PN \arrow[dl, no head] \arrow[dr, no head] & & NP \arrow[dl, no head] \arrow[dr, no head] & \\
PO \arrow[dr, no head] & & NN \arrow[dl, no head] \arrow[dr, no head] & & OP \arrow[dl, no head] \\
 & NO \arrow[dr, no head] & & ON \arrow[dl, no head] & \\
 & & OO & &
\end{tikzcd}$$

where each pair of letters indicates first who has a winning strategy in the game for l and then who has a winning strategy in the game for $\bar{l}$. P corresponds to the *Proponent*, O to the *Opponent* and N, in turn, indicates that none of them has a winning strategy.

This taxonomy of cases bears a resemblance to Antoniou et al.'s ([1, 8]) classification of proof outcomes for a proposition and its negation in Defeasible Logic. Nevertheless, that approach differs in many ways from DeLP, mostly because a single derivation is considered to draw a conclusion, while in DeLP several "proofs" (i.e., arguments) are played out one against the other to yield warrant.

We need an interpretation for each of these outcomes. We have marked in the table with YES and NO the cases in which the arguments are clearly settled for or against the literal l. If for l the *Proponent* has a winning strategy and neither they nor the *Opponent* have one for the warrant game on $\bar{l}$, we want to give a positive answer for the literal l, but not as strong one as we would for the case in which the *Proponent* has a winning strategy for both games. This provides a wider framework than the one presented in [7], to encompass argumentation systems without the full restrictions imposed by DeLP.

Example 13 We look now at the warrant game initiated by the *Opponent* for the literal $\sim a$ in the conditions we established in Example 12. Using $\leq_1 = <_1 \cup \approx_1$, the game is

$$\begin{tikzcd}
 & \emptyset^O \arrow[d, dash] & & \\
 & \langle \mathcal{A}_4, \sim a\rangle^P \arrow[d, dash] & & \\
 & \langle \mathcal{A}_1, a\rangle^O \arrow[dl, dash] \arrow[dr, dash] & & \\
\langle \mathcal{A}_2, \sim b\rangle & & \langle \mathcal{A}_4, \sim a\rangle^P \arrow[d, dash] & \\
(-1,1) & & \langle \mathcal{A}_1, a\rangle^O \arrow[dl, dash] \arrow[dr, dash] & \\
 & \langle \mathcal{A}_2, \sim b\rangle & & \langle \mathcal{A}_4, \sim a\rangle^P \\
 & (-1,1) & & \vdots
\end{tikzcd}$$

so the outcome for the query a is OO, yielding a negative answer. This checks with the proposed pre-defeating relation, which implied that the argument $\langle \mathcal{A}_1, a\rangle$ is defeated.

On the other hand, using the relation $\prec_2 = \emptyset$, the game we get for $\sim a$ is

$$\begin{tikzcd}
 & \emptyset^O \arrow[d, dash] & & \\
 & \langle \mathcal{A}_4, \sim a\rangle^P \arrow[d, dash] & & \\
 & \langle \mathcal{A}_1, a\rangle^O \arrow[dl, dash] \arrow[dr, dash] & & \\
\langle \mathcal{A}_2, \sim b\rangle^P \arrow[d, dash] & & \langle \mathcal{A}_4, \sim a\rangle^P \arrow[d, dash] & \\
\langle \mathcal{A}_3, b\rangle^O \arrow[d, dash] & & \langle \mathcal{A}_1, a\rangle^O \arrow[dl, dash] \arrow[dr, dash] & \\
\langle \mathcal{A}_2, \sim b\rangle^P & \langle \mathcal{A}_2, \sim b\rangle^P & & \langle \mathcal{A}_4, \sim a\rangle^P \\
\vdots & \vdots & & \vdots
\end{tikzcd}$$

So here we get the result NN, and the answer "Undecided" for the query a, which reflects the fact that we have not chosen any of the arguments to be stronger than the others.

We have the following trivial result:

Proposition 5 *The literals that are facts of a program always get the outcome PP and therefore the answer YES.*

Proof Since the class of facts is not contradictory, given a fact l, we know by Lemma 1, 1 that $\langle \emptyset, l \rangle$ is an argument and it has no counter-arguments. Therefore, the game for l ends after the *Proponent* chooses $\langle \emptyset, l \rangle$ and since there is no counterargument, they win. Alternatively, for $\bar{l}$, there can be no arguments so the *Opponent* has no valid moves and the *Proponent* wins again. Therefore, the corresponding element in the lattice is PP, which yields a YES answer. □

It should be clear that the query for a literal l and the one for $\bar{l}$ are directly related by a reversal of the roles of the *Proponent* and *Opponent*.

5 Finite warrant games

If we restrict the definition of warrant games in a way that makes them finite, then in each game one of the players has a winning strategy. The reason for this is that finite games are always *determined* [13]. Our set of possible results (or truth-values) for a given query gets reduced to four possibilities: PP, PO, OP and OO.

$$\begin{tikzcd}
 & PP \arrow[dl, dash] \arrow[dr, dash] & \\
PO \arrow[dr, dash] & & OP \arrow[dl, dash] \\
 & OO &
\end{tikzcd}$$

The top and bottom of the lattice above yield, respectively the YES and NO answers to the query. The middle possibilities yield *Undecided*, but with different meanings. If we obtain PO, both the literal l and its negation are warranted and we are facing a (defeasible) contradiction, while if the answer is OP, neither the literal nor its negation can be convincingly supported. We may see this as an informational gradient in the lattice. From left to right we move from OP (too little information) to OO and PP where we have the right amount of information to justify an answer, to PO where we have too much information.

It is enough to have a single attacked argument for the game to become infinite, in view of Proposition 1, 1. A direct way of avoiding this is to forbid the repetition of arguments. This alone ensures the finiteness of the warrant games, since the number of arguments for a given program is finite. But the following example shows it is also a bad idea to allow subarguments of an already used argument to come back into play.

Example 14 Consider the program $\mathbb{P}_1$ discussed before. We assume the relation $\prec$ to be R so all attacking arguments are proper defeaters. If we don't allow in a warrant game the repetition of an argument previously played, then the query for a makes us consider the games:

$$\begin{tikzcd}
\emptyset^P \arrow[d, dash] & \emptyset^O \arrow[d, dash] \\
\langle \mathcal{A}_1, a\rangle^O \arrow[d, dash] & \langle \mathcal{A}_2, \sim a\rangle^P \arrow[d, dash] \\
\langle \mathcal{A}_2, \sim a\rangle_{(-1,1)} & \langle \mathcal{A}_1, a\rangle_{(1,-1)}
\end{tikzcd}$$

So this query yields OP, undecided. If we query for the literal d instead, we get the games:

$$\begin{tikzcd}
\emptyset^P \arrow[d, dash] & \emptyset^O_{(1,-1)} \\
\langle \mathcal{A}_1, d\rangle^O \arrow[d, dash] & \\
\langle \mathcal{A}_2, \sim a\rangle^P \arrow[d, dash] & \\
\langle \mathcal{A}_1, a\rangle_{(1,-1)} &
\end{tikzcd}$$

That is, the query for d yields and answer YES, while in the program we have the strict rule $d \leftarrow a$ and we have seen that the query for a was inconclusive.

We seek then to disallow histories in the warrant games where subarguments are reintroduced. We define first:

Definition 15 A history y in a warrant game is said to be *s-acyclic* if there is no subsequence $\langle \mathcal{A}_0, h_0\rangle$,$\langle \mathcal{A}_1, h_1\rangle$,...,$\langle \mathcal{A}_{k+1}, h_{k+1}\rangle$ in y such that $\langle \mathcal{A}_{k+1}, h_{k+1}\rangle$ is a subargument of $\langle \mathcal{A}_0, h_0\rangle$.

There are other ways in which the information carried by an argument could be reinforced in a warrant game. One is by allowing the accrual of support [21]. That is, suppose that an argument $\langle \mathcal{A}_j, h_j\rangle$ has been blocked in a sequence y by an argument $\langle \mathcal{A}_{j+1}, h_{j+1}\rangle$. If, in turn, we allow $\langle \mathcal{A}_{j+1}, h_{j+1}\rangle$ to be blocked in y by $\langle \mathcal{A}_{j+2}, h_{j+2}\rangle$, it means that the literal supported by $\langle \mathcal{A}_j, h_j\rangle$ accrues further support from $\langle \mathcal{A}_{j+2}, h_{j+2} \rangle$ although $\prec$ is not actually involved in the attack. This amounts to counting how many arguments defend and attack a given literal, and while it would be an acceptable criterion, it is one that DeLP carefully chooses to avoid. We can disallow this possibility by means of:

Definition 16 *(n2b condition)*: Given $\langle \mathcal{A}_j, h_j \rangle$, $\langle \mathcal{A}_{j+1}, h_{j+1} \rangle$ and $\langle \mathcal{A}_{j+2}, h_{j+2} \rangle$ in a history y, if $\langle \mathcal{A}_j, h_j \rangle \approx \langle \mathcal{A}_{j+1}, h_{j+1} \rangle$, then it is not the case that $\langle \mathcal{A}_{j+1}, h_{j+1} \rangle \approx \langle \mathcal{A}_{j+2}, h_{j+2} \rangle$. In other words, a blocking defeater can not be followed by another blocking defeater.[2]

Another undesirable possibility that may arise in any play of a warrant game is that one of the players support some disagreeing literals. More precisely, that in a history $y = \langle \mathcal{A}_0, h_0 \rangle, \ldots, \langle \mathcal{A}_K, h_K \rangle$, either $\bigcup_{j \geq 0} \mathcal{A}_{2j}$ or $\bigcup_{j \geq 0} \mathcal{A}_{2j+1}$ are contradictory. This means that the information used to support a literal is further used to support its negation. To avoid this possibility we can demand the following property of histories:

Definition 17 A history y is said *concordant* if the sets $\bigcup_{j \geq 0} \mathcal{A}_{2j}$ and $\bigcup_{j \geq 0} \mathcal{A}_{2j+1}$ are not contradictory.

Given that some sequences can be disallowed, those that are allowed are captured by a *protocol* that restricts the admissible actions that may be taken in a warrant game. DeLP assumes, in particular, the following protocol:

Definition 18 In a warrant game for a literal l, a history is $y = \langle \mathcal{A}_0, h_0 \rangle, \ldots, \langle \mathcal{A}_K, h_K \rangle$, where $\langle \mathcal{A}_j, h_j \rangle \leq \langle \mathcal{A}_{j+1}, h_{j+1} \rangle$, for $j = 0, \ldots, K - 1$. We say that y is *DeLP-admissible* if and only if:

- it satisfies the *n2b* condition.
- it is *concordant*,
- it is *s-acyclic*.

While it is clear that each of the three conditions are intended to avoid certain behaviors, the choice of conditions is by no means the only possible one. One could ask for consistency of the *Proponent* but not of the *Opponent*, or viceversa, or we could allow a blocking argument to be followed by only one blocking argument, etc. The reasons why DeLP-admissibility are described by these conditions are purely pragmatic, since in most examples these conditions are enough to get the intuitively sound answers and embody some reasonable constraints on a discussion between two rational agents [7, 11].

[2]We have seen that an argument can be both a proper and a blocking defeater of another. This is not a problem when one checks for the *n2b* condition, since we only need to determine whether the defeaters are blocking or not.

Example 15 We take a look now at how the protocol above applies to the program and query from Examples 12 and 13. Using the relation $\leq_1$ or $\leq_3$ we get the games:

$$\begin{tikzcd}
 & \emptyset^P \arrow[d, dash] & \\
 & \langle \mathcal{A}_1, a \rangle^O \arrow[dl, dash] \arrow[dr, dash] & \\
\underset{(-1,1)}{\langle \mathcal{A}_2, \sim b \rangle} & & \underset{(-1,1)}{\langle \mathcal{A}_4, \sim a \rangle}
\end{tikzcd}
\qquad
\begin{tikzcd}
\emptyset^O \arrow[d, dash] \\
\langle \mathcal{A}_4, \sim a \rangle^P \arrow[d, dash] \\
\langle \mathcal{A}_1, a \rangle^O \arrow[d, dash] \\
\underset{(-1,1)}{\langle \mathcal{A}_2, \sim b \rangle}
\end{tikzcd}$$

and OO as the answer.

Using the relation $\leq_2$:

$$\begin{tikzcd}
 & \emptyset^P \arrow[d, dash] & \\
 & \langle \mathcal{A}_1, a \rangle^O \arrow[dl, dash] \arrow[dr, dash] & \\
\underset{(-1,1)}{\langle \mathcal{A}_2, \sim b \rangle} & & \underset{(-1,1)}{\langle \mathcal{A}_4, \sim a \rangle}
\end{tikzcd}
\qquad
\begin{tikzcd}
\emptyset^O \arrow[d, dash] \\
\langle \mathcal{A}_4, \sim a \rangle^P \arrow[d, dash] \\
\underset{(1,-1)}{\langle \mathcal{A}_1, a \rangle}
\end{tikzcd}$$

We get OP (Undecided) as the answer.

Proposition 6 *A warrant game in which each history is DeLP-admissible is finite.*

Proof Immediate, since every history is DeLP-admissible, and therefore an s-acyclic sequence. Given that Arg($\mathbb{P}$) is finite, the length of each history y is finite. □

In [7, Section 5], truth values are determined by means of "dialectical trees". These are essentially the subtrees obtained from a warrant game starting with a one-argument history. The tree nodes are then marked with D and U (for *Defeated* and *Undefeated*, respectively) as follows:

– All leaves are marked U.
– A nonterminal history is marked U if all its children are marked D. Otherwise it is marked D.

A literal l is said to be *warranted* if there exists an argument $\langle \mathcal{A}, l \rangle$ such that in its marked dialectical tree, the root argument is marked with U. We show now that this leads to the same outcome as our warrant games.

García and Simari's procedure of marking the dialectical tree is equivalent to finding the backward induction solution of the game: if $\langle \mathcal{A}, l \rangle$ is marked U then, the

Proponent has a winning strategy in the game that starts at $\langle \mathcal{A}, l \rangle$. Otherwise, the *Opponent* has a winning strategy in this game:

Proposition 7 *A literal l is warranted by a program $\mathbb{P}$ if and only if the* Proponent *has a winning strategy in the game for l.*

Proof Assume that the literal l is warranted by the program $\mathbb{P}$. This means that there exists an argument $\langle \mathcal{A}, l \rangle$ such that in its subtree, marked as a dialectical tree, it is marked as undefeated. We use this to define a strategy σ for *Proponent* in the warrant game for l.

In the first move, *Proponent* chooses the argument $\langle \mathcal{A}, l \rangle$, which is undefeated by hypothesis. This means that it is a leaf or all its children (which correspond to the actions that can be chosen by the *Opponent*) are defeated. In the first case, we are done. In the second one, we take a look at what happens next in the warrant game. All the alternatives the *Opponent* are marked as defeated, so for each one of these, the *Proponent* has at least one choice that leads them to a history marked as undefeated, from which we can iterate the previous reasoning until we reach a leaf. Any of these choices can be added to the strategy σ. For the histories with first move other than $\langle \mathcal{A}, l \rangle$, we can complete σ with arbitrary actions, since these are irrelevant.

Observe now that following the strategy σ constructed above, the *Proponent* reaches a terminal history in an odd number of steps: the path followed in the tree of the warrant game starts with the empty history, then goes to the root of the dialectical tree marked undefeated, and then goes through alternatively marked defeated and undefeated histories. So it is a winning strategy for *Proponent*.

Assume next that the literal l is not warranted by the program $\mathbb{P}$. This is, for every argument of the form $\langle \mathcal{A}, l \rangle$, the corresponding marked dialectical tree has $\langle \mathcal{A}, l \rangle$ marked as defeated. We can now build a winning strategy τ for the *Opponent*. Since all the arguments $\langle \mathcal{A}, l \rangle$ are defeated, they must have at least one undefeated child each. Let τ be a strategy that chooses one of those. Since they are undefeated, they must be leaves, and the *Proponent* has no choice, or they must have all of its children marked as defeated. We can see that τ can thus be defined so that a leaf will be reached in an even number of steps, so it is a winning strategy for *Opponent*. □

A concern that may arise is whether the conditions in the definition of DeLP admissibility are not redundant. The following result, for instance, hints at a certain overlap between s-acyclicity and concordance:

Proposition 8 *If a history in a warrant game is concordant, then none of the players has used a subargument of an argument used by the other player.*

Proof Assume towards contradiction that a history $y = \langle \mathcal{A}_0, h_0 \rangle, \ldots, \langle \mathcal{A}_K, h_K \rangle$ is such that there exist $\langle \mathcal{A}_j, h_j \rangle$ and $\langle \mathcal{A}_k, h_k \rangle$ in y, with $j < k$, where $k - j$ is odd, satisfying $\mathcal{A}_k \subseteq \mathcal{A}_j$. This means that one of the players has played a sub-argument of an argument used by their rival.

We also know that there must exist a literal $d(h_k) \in C(\mathcal{A}_{k-1})$ such that for $\langle \mathcal{A}_{k-1}, h_{k-1} \rangle$ in y, $\langle \mathcal{A}_{k-1}, h_{k-1} \rangle \ R_{d(h_k)} \ \langle \mathcal{A}_k, h_k \rangle$. This literal disagrees with h_k.

Without loss of generality assume that k is odd and j even. Then, since $\mathcal{A}_k \subseteq \mathcal{A}_j$, $h_k \in C(\mathcal{A}_j) \subseteq \bigcup_{i\geq 0} \mathcal{A}_{2i}$

On the other hand, since $k-1$ is even, $d(h_k) \in C(\mathcal{A}_{k-1}) \subseteq C(\bigcup_{i\geq 0} \mathcal{A}_{2i})$, so this set is contradictory. That is, y is not concordant. □

Making moves consisting of subarguments of arguments used previously by rivals was already forbidden by the s-acyclicity condition, but we will use the proposition in the following section.

Proposition 9 *The conditions that define DeLP-admissibility, i.e., s-acyclicity, concordance and n2b, are independent and consistent.*

Proof To prove the independence, it is enough to show an example where two of the conditions hold and the other one fails. Let us see this for each case:

– *s-acyclicity and n2b do not imply concordance*. Consider the program $\mathbb{P}_4$ with $\Pi_F = \{x, y, z\}$; $\Pi_R = \{l \leftarrow \sim h\}$ and $\Delta = \{\sim h \prec x,\ l \prec y,\ h \prec l,\ \sim l \prec z\}$. Then we have arguments $\langle \mathcal{A}_1, l\rangle$, $\langle \mathcal{A}_2, h\rangle$ and $\langle \mathcal{A}_3, \sim l\rangle$, where $\mathcal{A}_1 = \{\sim h \prec x\}$, $\mathcal{A}_2 = \{l \prec y, h \prec l\}$ and $\mathcal{A}_3 = \{\sim l \prec z\}$. Let $\prec = R$ so that all attacking arguments are proper (and never blocking) defeaters, so the n2b condition will be trivially satisfied. The sequence $\langle \mathcal{A}_1, l\rangle$, $\langle \mathcal{A}_2, h\rangle$, $\langle \mathcal{A}_3, \sim l\rangle$ is s-acyclic, since none of these arguments is subargument of one of the others. . On the other hand, l, and $\sim l$ are both consequences of $\Pi_F \cup \Pi_R \cup \mathcal{A}_1 \cup \mathcal{A}_3$, i.e., the sequence is not concordant.
– *concordance and n2b do not imply s-acyclicity*. Consider a program $\mathbb{P}_5$ with $\Pi_F = \{x, y\}$; $\Pi_R = \{l \leftarrow \sim h, \sim l \leftarrow s, \sim s \leftarrow \sim h\}$ and $\Delta = \{\sim h \prec x, s \prec y\}$. If we let $\prec = R$, and define $\mathcal{A}_1 = \{\sim h \prec x\}$ and $\mathcal{A}_2 = \{s \prec y\}$, the sequence $\langle \mathcal{A}_1, l\rangle$, $\langle \mathcal{A}_2, \sim l\rangle$, $\langle \mathcal{A}_1, \sim s\rangle$ is obviously not s-acyclic but is concordant and satisfies n2b.
– *s-acyclicity and concordance do not imply the n2b condition*. Consider $\mathbb{P}_6$ with $\Pi_F = \{b, p, t\}$; $\Pi_R = \emptyset$ and $\Delta = \{d \prec t, \sim d \prec b, \sim d \prec p\}$. We have arguments $\langle \mathcal{A}_1, \sim d\rangle$, $\langle \mathcal{A}_2, d\rangle$ and $\langle \mathcal{A}_3, \sim d\rangle$, where $\mathcal{A}_1 = \{\sim d \prec b\}$, $\mathcal{A}_2 = \{d \prec t\}$ and $\mathcal{A}_3 = \{\sim d \prec p\}$. Since $\langle \mathcal{A}_1, \sim d\rangle R_{\sim d} \langle \mathcal{A}_2, d\rangle$ and $\langle \mathcal{A}_2, d\rangle R_d \langle \mathcal{A}_3, \sim d\rangle$, if $\prec = \emptyset$, all the arguments in the sequence are blocking, but the sequence is s-acyclic and concordant.

The consistency of the conditions can be seen in Example 15. □

6 Admissibility protocols and truth values

Now we turn our attention to the relation between the warrant games for l and $\bar{l}$ under the DeLP protocol introduced in the previous section. We prove next that not all the truth values are attainable. This has been in fact a long-standing conjecture, implicitly incorporated in the mechanism of DeLP.

Theorem 1 *If all the admissible histories for a warrant game satisfy s-acyclicity and the concordance condition, the truth value PO cannot be obtained.*

Proof Assume towards contradiction that *Proponent* has a winning strategy σ in the game for l, while *Opponent* has a winning strategy τ for the game for $\bar{l}$. Let $\langle \mathcal{A}, l\rangle$

be the first move prescribed by σ and $\langle \mathcal{B}, \bar{l} \rangle$ the one by τ. Both of these must exist under our hypothesis. We define a strategy σ' for the *Proponent* in the game for $\bar{l}$ as follows:

$$\sigma'(\langle \mathcal{B}, \bar{l} \rangle, \langle \mathcal{A}, l \rangle, \langle \mathcal{B}_1, g_1 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \ldots \langle \mathcal{B}_k, g_k \rangle) =$$

$$\sigma(\langle \mathcal{A}, l \rangle, \langle \mathcal{B}_1, g_1 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \ldots \langle \mathcal{B}_k, g_k \rangle) = \langle \mathcal{A}_k, h_k \rangle$$

where the arguments $\mathcal{A}_i$, $1 \leq i \leq k$ are the ones played by the *Proponent* and the $\mathcal{B}_j$, $1 \leq j \leq k$ are played by the *Opponent*.

The strategy σ' is well defined for the histories of the form $\langle \mathcal{B}, \bar{l} \rangle, \langle \mathcal{A}, l \rangle, \ldots \langle \mathcal{B}_k, g_k \rangle$ that appear in the game for $\bar{l}$ because then $\langle \mathcal{B}, \bar{l} \rangle, \langle \mathcal{A}, l \rangle, \ldots \langle \mathcal{B}_k, g_k \rangle$ is an admissible history and so is $\langle \mathcal{A}, l \rangle, \ldots \langle \mathcal{B}_k, g_k \rangle$, and all such admissible histories must appear in the warrant game for l.

We extend σ' to a strategy for *Proponent* in any way: we don't care what happens in other branches of the game tree since we won't consider them in the reasoning of this proof.

Now let $\langle \mathcal{B}, \bar{l} \rangle, \langle \mathcal{A}, l \rangle, \langle \mathcal{B}_1, g_1 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \ldots \langle \mathcal{B}_k, g_k \rangle$ be the terminal history that results from playing σ' against τ. It must end with $\langle \mathcal{B}_k, g_k \rangle$ because τ is a winning strategy for *Opponent*. As a consequence, $\langle \mathcal{A}, l \rangle, \langle \mathcal{B}_1, g_1 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \ldots \langle \mathcal{B}_k, g_k \rangle$ is a terminal history in the game for l. If it wasn't terminal, there would exist $\langle \mathcal{A}_k, h_k \rangle = \sigma(\langle \mathcal{A}, l \rangle, \ldots \langle \mathcal{B}_k, g_k \rangle)$ such that the history $\langle \mathcal{B}, \bar{l} \rangle, \langle \mathcal{A}, l \rangle, \ldots \langle \mathcal{B}_k, g_k \rangle \langle \mathcal{A}_k, h_k \rangle$ is not admissible. But this can only happen if $\langle \mathcal{A}_k, h_k \rangle$ is a subargument of $\langle \mathcal{B}, \bar{l} \rangle$ and that was shown to be impossible in Proposition 8.

So we have the terminal history $\langle \mathcal{A}, l \rangle, \langle \mathcal{B}_1, g_1 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \ldots \langle \mathcal{B}_k, g_k \rangle$ in which the *Opponent* wins, and it is obtained in the game for l using the strategy σ, which was a winning strategy for *Proponent*, a contradiction. □

In our interpretation of truth values this means that in both cases the information about l (and $\bar{l}$) is never excessive. That is, a literal can only be true, false or ambiguous, but never self-contradictory.

This is a consequence from the fact that both players are using the same defeating relation, and reminds us that they are just a useful construct to determine one agent's reasoning.

Another important lesson to be drawn from this result is that the truth values depend heavily on the admissibility protocol. To see this, assume that the protocol adjudicates a win to a player if they can put forward an argument. Then, for any of the examples we have considered in this paper we would have that PO is the truth value of the query.

For the special case in which the pre-defeating relation is empty, we have the following result, which provides a computational shortcut:

Proposition 10 *If the relation $\prec$ is the empty set, then the first player has a winning strategy for a warrant game for a given literal l if and only if there exists at least one argument for the literal which has no attacking arguments.*

Proof Since all attacking arguments are blocking defeaters, the branches of the tree game will have length at most two. The first player has a winning strategy if and only if there is a branch of length one, and this can only happen if an argument for the literal l has no attackers. □

Now that we know that warrant games under the DeLP protocol only yield one of three truth values, we may streamline the procedure for determining the truth value of each literal. Given a query for a literal l, if there exists a winning strategy for the *Proponent* in warrant game for l, then the *Proponent* will also have a winning strategy in the game for $\bar{l}$ and no further calculation is needed. If the *Opponent* is the one who has a winning strategy in the warrant game for l, we need to analyze the warrant game for $\bar{l}$. This fully justifies the procedure described in [7].

7 Conclusions and future work

We have re-casted the basic definitions of DeLP in a simple mathematical language so that we can analyze its underlying assumptions anew. We hope that the proposed formal framework can shed some light on how the warrant mechanism works and in particular, on the protocol that defines admissibility of an argumentation line. We have borrowed a page from game theory to present the dialectical process in what we believe is a natural way. By initially dropping all restrictions in the dialectical process we uncovered more possibilities for the outcome of a proposed query. We can see these outcomes as truth values, yielding more information on the nature of what a given DeLP program concludes about each literal.

We checked that the facts of the program get a positive answer before turning to the conditions that make our *warrant games* finite. These conditions can come either from the *defeating* relation itself or from an imposed *protocol* on the way the games are constructed. Once we have assumptions that guarantee the games considered are finite, we have fewer truth values.

The inference procedure associated with finding winning strategies has a natural "semantical" counterpart. That is, the pair of winners, one for each of the two games can be immediately associated to a truth value as described in the table in section 4. This depends on the admissibility protocol and the defeating relation adopted.

Game semantics for DeLP have already been introduced in [2] and [3]. The main differences with our approach are that, on one hand, they do not apply standard notions of game theory, while on the other hand they restrict themselves to a single game, that may yield only three truth values, one of which only applies when the program has nothing to say about a literal.

The protocol of DeLP allows a variant of the "strategy stealing" procedure that makes the winner in the game for l the winner in the game for $\bar{l}$ and viceversa [19].

DeLP is usually seen as a representation of *skeptical* reasoning [19]. This is so, since as we have seen, positive literals with truth value PP are warranted. Therefore, contrary to what happens in more *credulous* formalisms, ambiguously valued literals do not become warranted. Nevertheless, since the inception of Dung's [6] abstract framework of argumentation, skepticism has been associated with the *grounded semantics* of an argument system. In this sense, we will see that DeLP is not skeptical, due to the properties of the warrant protocol.

In order to build this case, let us consider $\mathrm{Arg}(\mathbb{P})$ with the defeating relation $\leq$ defined on it. Then, a function $\mathbf{F} : \mathcal{P}(\mathrm{Arg}(\mathbb{P})) \to \mathcal{P}(\mathrm{Arg}(P))$ can be defined, such that for any $S \subseteq \mathrm{Arg}(P)$, $\mathbf{F}(S) = \{\alpha \in \mathrm{Arg}(\mathbb{P}) :$ for every β such that $\alpha \leq \beta$, there exists $\gamma \in S$, $\beta \leq \gamma\}$. A *complete extension* of $\mathrm{Arg}(\mathbb{P})$ is a fixed point of $\mathbf{F}$. The *grounded extension* is the *least* of those fixed points. It obtains iterating $\mathbf{F}$ up from $\emptyset$. That is, as the limit of the sequence $\mathbf{F}(\emptyset), \mathbf{F}(\mathbf{F}(\emptyset)), \ldots$.

If DeLP were a skeptical formalism in terms of the grounded extension, we would have that a literal l is warranted if and only if the argument $\langle \mathcal{A}, l\rangle$ that yields a win for the *Proponent* in the warrant game for l belongs to the grounded extension of $\langle Arg(P), \leq\rangle$. The following example shows that this is not necessarily the case:

Example 16 Consider a program in which $\Pi_F = \{a, c\}$, $\Pi_R = \{k \leftarrow l\}$ and $\Delta = \{l \prec a, \sim l \prec c, \sim k \prec \sim l\}$. Consider four arguments: $\mathcal{A}_1 = \{\sim l \prec c\}$ for $\sim l$; $\mathcal{A}_2 = \{l \prec a\}$ for l and k, and $\mathcal{A}_3 = \{\sim l \prec c, \sim k \prec \sim l\}$ for $\sim k$. Suppose furthermore that the pre-defeating relation is as follows: $\langle \mathcal{A}_2, l\rangle \prec \langle \mathcal{A}_1, \sim l\rangle$; $\langle \mathcal{A}_3, \sim k\rangle \prec \langle \mathcal{A}_2, l\rangle$; $\langle \mathcal{A}_2, k\rangle \prec \langle \mathcal{A}_1, \sim l\rangle$; $\langle \mathcal{A}_3, \sim k\rangle \prec \langle \mathcal{A}_2, k\rangle$. Consequently, the defeating relation is as follows: $\langle \mathcal{A}_2, l\rangle < \langle \mathcal{A}_1, \sim l\rangle$; $\langle \mathcal{A}_3, \sim k\rangle < \langle \mathcal{A}_2, l\rangle$; $\langle \mathcal{A}_2, k\rangle < \langle \mathcal{A}_1, \sim l\rangle$; $\langle \mathcal{A}_3, \sim k\rangle < \langle \mathcal{A}_2, k\rangle$. Notice that all the defeaters are proper.

It is easy to see that the literal $\sim k$ is not warranted, since in the game starting with $\langle \mathcal{A}_3, \sim k\rangle$, the *Opponent* can respond with either $\langle \mathcal{A}_2, \sim k\rangle$ or $\langle \mathcal{A}_2, \sim l\rangle$ to which the *Proponent* cannot respond with $\langle \mathcal{A}_1, \sim l\rangle$ since $\mathcal{A}_1 \subseteq \mathcal{A}_3$.

On the other hand, $\mathbf{F}(\emptyset) = \{\langle\emptyset, a\rangle, \langle\emptyset, c\rangle, \langle \mathcal{A}_1, \sim l\rangle\}$ while the grounded extension is $\mathbf{F}(\mathbf{F}(\emptyset)) = \{\langle\emptyset, a\rangle, \langle\emptyset, c\rangle, \langle \mathcal{A}_1, \sim l\rangle, \langle \mathcal{A}_3, \sim k\rangle\}$.

This framework of analysis can be extended to other argumentative systems. In a system like DeLP, where arguments support certain logical formulas[3] any defeating relation among these arguments may be applied to yield games for a formula and its negation. The defeating relation and the properties of the protocol determine the actual partition of the class of formulas. In a more general setting, as shown in [6], when arguments are abstract entities there is no "negation" involved. But then, we can still partition the class of arguments in terms of a single game for an argument. If there is a winning strategy for it, it is deemed true.

It would be interesting to analyze settings in which infinite games could be considered, to make full use of the possible truth values in the most general framework we have given. Another interesting line of study is to look for conditions the pre-defeating relation should satisfy in order give a defeating relation that yields games with interesting properties (such as finiteness, concordance or s-acyclicity) without resorting to a somewhat arbitrary protocol.

References

1. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. ACM Trans. Comput. Log. **2**(2), 255–287 (2001)
2. Cecchi, L., Simari, G.: Sobre la relación entre la semántica gs y el razonamiento rebatible. In: X CACiC, pp. 1883–1894. Universidad Nacional de La Matanza (2004)

[3]The reason why DeLP is concerned only with literals is the one why LP is: to facilitate the implementation of working interpreters (see [10]).

3. Cecchi, L., Fillottrani, P., Simari, G.: On the complexity of delp through game semantics. In: Dix, J., Hunter, A. (eds.) Proc. 11th Intl. Workshop on Nonmonotonic Reasoning (NMR 2006), Windermere, UK. IfI Technical Report Series, pp. 386–394. Clausthal University (2006)
4. Chesñevar, C.I., Maguitman, A.G., Loui, R.P.: Logical models of argument. ACM Comput. Surv. **32**(4), 337–383 (2000)
5. Chesñevar, C.I., Dix, J., Stolzenburg, F., Simari, G.R.: Relating defeasible and normal logic programming through transformation properties. Theor. Comput. Sci. **290**(1), 499–529 (2003)
6. Dung, P.M.: On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming, and n-person games. Artif. Intell. **77**, 321–357 (1995)
7. García, A., Simari, G.: Defeasible logic programming: an argumentative approach. Theory and Practice of Logic Programming **4**(1), 95–138 (2004)
8. Governatori, G., Maher, M.J., Antoniou, G., Billington, D.: Argumentation semantics for defeasible logic. J. Log. Comput. **14**(5), 675–702 (2004)
9. Lifschitz, V.: Foundations of logic programming. In: Brewka, G. (ed.) Principles of Knowledge Representation, pp. 69–127. Center for the Study of Language and Information, Stanford, CA, USA (1996)
10. Lloyd, J.W.: Foundations of Logic Programming, 2nd edn. Springer (1987)
11. Lorenzen, P., Lorenz, K.: Dialogische Logik. Wissenschaftliche Buchgesellschaft, Darmstadt (1978)
12. Loui, R.P.: Process and policy: resource-bounded non-demonstrative reasoning. Comput. Intell. **14**, 1–38 (1998)
13. Mycielski, J.: Games with perfect information. In: Aumann, R.J., Hart, S. (eds.) Handbook of Game Theory with Economic Applications, vol. 1, Chapter 3, pp. 41–70. Elsevier (1992)
14. Nute, D.: Defeasible logic. In: Gabbay, D., Hogger, C.J., Robinson, J.A. (eds.) Handbook of Logic in Artificial Intelligence and Logic Programming. Vol. 3: Nonmonotonic Reasoning and Uncertain Reasoning, pp. 353–395. Oxford University Press, Oxford (1994)
15. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. MIT Press, Cambridge, MA (1994)
16. Pollock, J.L.: Defeasible reasoning. Cogn. Sci. **11**(4), 481–518 (1987)
17. Pollock, J.L.: Cognitive Carpentry: A Blueprint for how to Build a Person. MIT Press, Cambridge, MA (1995)
18. Prakken, H., Sartor, G.: A dialectical model of assessing conflicting arguments in legal reasoning. Artif. Intell. Law **4**, 331–368 (1996)
19. Prakken, H., Vreeswijk, G.: Handbook of Philosophical Logic, vol. 4. Chapter: Logics for Defeasible Argumentation, 2nd edn., pp. 219–318. Kluwer, Dordrecht (2002)
20. Simari, G., Loui, R.P.: A mathematical treatment of defeasible reasoning and its implementation. Artif. Intell. **53**(2–3), 125–157 (1992)
21. Verheij, B.: Rules, Reasons, Arguments. Formal Studies of Argumentation and Defeat. PhD thesis, University of Maastricht (1996)
22. Viglizzo, I.D., Tohmé, F.A., Simari, G.R.: An alternative foundation for delp: defeating relations and truth values. In: Hartmann, S., Kern-Isberner, G. (eds.) FoIKS. Lecture Notes in Computer Science, vol. 4932, pp. 42–57. Springer (2008)
23. Vreeswijk, G.: Abstract argumentation systems. Artif. Intell. **90**(1–2), 225–279 (1997)